

RP04/5/6

RP04/5/6 MLT DR LGC
CZRJDD0

AH 9197D MC
COPYRIGHT 76-79
FICHE 1 OF 2

NOV 1979
digital
MADE IN USA

RP04/5/6

RP04/5/6 MLT DR LGC
CZRJDD0

AH 9197D MC

NOV 1979

COPYRIGHT 76 79

digital

FICHE 2 OF 2

MADE IN USA

.REM @

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

PRODUCT CODE: AC-9195D-MC
PRODUCT NAME: CZRJDDO RP04/5/6 MLT-DR LGC
DATE CREATED: MAY, 1979
MAINTAINER: DIAGNOSTIC ENGINEERING

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

COPYRIGHT (C) 1976,1979 DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

REVISION HISTORY

-
1. CHANGE TITLE FROM CZRJDCOTO CZRJDDO
 2. MODIFY STARTING ADDRESSES TO SAFEGUARD DRIVES IN MULTI-PROCESSOR ENVIRONMENT. SEE PARAGRAPH 2.4 BELOW.
 3. MODIFY CODE IN "CLEAR DRIVE PARAMETER BLOCK" SUBROUTINE (CLRDPB) TO PREVENT CLEARING LOCATION \$RPDT.
 4. MODIFY CODE IN "DRIVE INITIALIZATION" ROUTINE (DRVINT) TO STORE DRIVE TYPE IN APPROPRIATE DRIVE PARAMETER BLOCK.
-

CONTENTS

-
1. ABSTRACT
 2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 MEDIA
 - 2.3 PRELIMINARY PROGRAMS
 - 2.4 PROGRAMMABLE DRIVES
 3. OPERATING THE PROGRAM
 - 3.1 LOADING THE PROGRAM
 - 3.2 STARTING THE PROGRAM
 - 3.3 RESTARTING THE PROGRAM
 - 3.4 PROGRAM CONTROL
 - 3.5 PASS/TEST TERMINATION
 - 3.5.1 PASS TERMINATION
 - 3.5.2 TEST TERMINATION
 - 3.6 RUN TIME
 - 3.6.1 DATA TRANSFER MODE
 - 3.6.2 SEEK VERIFICATION MODE
 - 3.7 UNIBUS & VECTOR ADDRESSES
 - 3.8 DUAL PORT OPERATION
 4. CONTROLLING THE PROGRAM
 - 4.1 DATE & OPERATOR IDENTIFICATION
 - 4.2 PARAMETERS
 - 4.2.1 PROGRAM CONTROL PARAMETERS
 - 4.2.2 PERIPHERAL DEVICE ADDRESSES
 - 4.2.3 PARAMETERS FOR THE FIRST OPERATION
 - 4.3 SWITCH REGISTER SETTINGS
 - 4.4 KEYBOARD COMMANDS
 - 4.4.1 'T' COMMAND
 - 4.4.2 'D' COMMAND
 - 4.4.3 'S' COMMAND
 - 4.4.4 'W' COMMAND
 - 4.4.5 'R' COMMAND

4.4.6 GENERAL COMMAND INFORMATION

5. PERFORMANCE SUMMARY TYPEOUT

- 5.1 PERFORMANCE SUMMARY TYPEOUT EXPLANATION
- 5.2 HARD/SOFT ERROR DEFINITIONS
 - 5.2.1 HARD ERRORS
 - 5.2.2 SOFT ERRORS

6. DATA CHECKING & ERROR RECOVERY

- 6.1 DATA BUFFER COMPARISON
- 6.2 VERIFICATION OF DATA WRITTEN
- 6.3 SECTOR REFORMATTING
- 6.4 BAD TRACK/SECTOR FLAGGING

7. ERROR MESSAGES

- 7.1 ERROR DESCRIPTION LINES
- 7.2 DETAIL ERROR LINES

8. PROGRAM DESCRIPTION

- 8.1 HOW THE PROGRAM OPERATES
- 8.2 DUAL PORT OPERATION
- 8.3 HOW VARIABLES ARE SELECTED FOR EACH OPERATION
- 8.4 DATA PATTERNS

9. PROGRAM LISTING

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. PROGRAMMABLE OPERATION IS INHIBITED WITH STARTING ADDRESS 200-SEE SEC 2.4. THEREFORE, USE STARTING ADDRESS 204 OR 220 FOR DUAL PORT TESTING.

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)

2.4 PROGRAMMABLE DRIVES (DUAL PORT ENABLED)

THIS REV INCORPORATES A SAFEGUARD TO PREVENT INADVERTENT CORRUPTION OF DISK PACKS IN PROGRAMMABLE DRIVES. THIS IS A POTENTIAL HAZARD IN RUNNING THIS PROGRAM IN A MULTIPROCESSOR SYSTEM. FOR THE STANDARD STARTING ADDRESS OF 200 THE PROGRAM HAS BEEN MODIFIED TO PREVENT INITIALIZING DRIVES FOUND TO BE PROGRAMMABLE. THIS MODIFICATION APPLIES ONLY TO THE FIELD ENVIRONMENT (XXDP CHAIN, STANDALONE) WHERE LOCATION 42 DOES NOT EQUAL LOCATION 46. FOR THE MANUFACTURING ENVIRONMENT (WHERE LOCATION 42 EQUALS LOCATION 46) PROGRAMMABLE

DRIVES WILL NOT BE INHIBITED. IF THE OPERATOR DESIRES TO RUN THIS PROGRAM USING PROGRAMMABLE DRIVES IN A FIELD ENVIRONMENT USE STARTING ADDRESS 220, WHERE 220 IS THE SAME AS 200 WITHOUT INHIBITING PROGRAMMABLE DRIVES. SEE SECTION 3.2,3.3 FOR A SUMMARY OF ALL STARTING ADDRESSES.

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. PROGRAMMABLE DRIVES ARE INHIBITED STANDALONE AND XXDP CHAIN-SEE SEC 2.4.

3.3 START THE PROGRAM AT LOCATION 204(8) IF THE RH11 OR THE RH70 IS NOT AT ADDRESS 176700. (NO INHIBITIONS)

STARTING THE PROGRAM AT 220 IS THE SAME AS 200 BUT PROGRAMMABLE DRIVES ARE NOT INHIBITED.

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 1×10^9 BITS READ OR NO MORE THAN 1 SEEK ERROR IN 1×10^6 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 1.875×10^8 WORDS (3×10^9 BITS).
- B. IF PARAMETER 'ENDET' IS 0, END OF PASS OCCURS WHEN THE DRIVE

HAS PERFORMED 3×10^6 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 PEACH 1.875×10^8 WORDS)
TO
- 8 DRIVES - APPROXIMATELY 11 HRS (FOR ALL DRIVES TO REACH 1.875×10^8 WORDS)

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

- 1 DRIVE - 1.7 HRS (1.875×10^8 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 (3×10^6 SEEKS)
TO
- 8 DRIVES - APPROXIMATELY 40 HRS (3×10^6 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.)

UNIBUS VECTOR

UNIT	ADDRESS	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.
- D. PROGRAMMABLE (DUAL PORT) OPERATION IS INHIBITED WITH STARTING ADDRESS 200-SEE SEC 2.4. THEREFORE, USE STARTING ADDRESS 204 OR 220 FOR DUAL PORT TESTING.

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 SUCCESSIVE 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

NAME	BASE	DEFAULT VALUE	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
WCSEL	8	000000	0 OR 1	THE NUMBER OF DATA COMPARISON IF PARAMETER = 0, THE DATA TRANSFER WORD COUNT IS RANDOMLY SELECTED BETWEEN 4 (10) AND THE VALUE IN 'MAXDL'.
ENDET	8	000001	0 OR 1	IF PARAMETER = 1, THE DATA TRANSFER WORD COUNT WILL BE THE VALUE IN 'MAXDL'
FORMAT	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.
				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 SPECIFY 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 RPO4/5/6 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

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

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

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.
W0<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 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 PERFORM A 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
-1200 MICRO-INCHES

+800 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 CAPABILITY, 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', 'MCRC')

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.

CZRJDDO,MLT-DR LGC MACY11 30A(1052) 03-JUL-79 10:41 PAGE 24^{J 2}
CZRJDD.P11 06-JUN-79 08:32

SEQ 0022

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 SIGNALLED; 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 SIGNALLED BY THE RH11
THE OPERATION COMPLETED NORMALLY; HOWEVER, THE PROGRAM FOUND EITHER ERROR BITS IN THE RPO4 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 REGISTER 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 SIGNALLED EITHER 'SKI' OR 'OCYL' ERROR BITS.

EM51 PROGRAM DETECTED POSITIONING ERROR

A HEADER COMPARE ERROR OCCURRED ('MCE'); 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)
WRDAT - 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 RPO4/5/6 REGISTERS. THE CONTENTS OF THIS WORD IDENTIFY HOW THE ERROR WAS PROCESSED BY THE RPO4/5/6 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 'RPECT' 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: YYY WRDS READ: ZZZ

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: WWW 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 SIGNALLED 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 ASSIGNMENT, 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 INSTRUCTION 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 NOT 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 TRANSFERRED 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 SIGNALLED BY THE RH11 - EM44
ECC LOGIC FAILURE - EM45
BUS ADDRESS OR WORD COUNT NOT CONSISTENT - EM46

8.2 DUAL PORT OPERATION

PROGRAMMABLE DRIVES (DUAL PORT) OPERATION IS INHIBITED WITH STARTING ADDRESS 200-SEE SEC 2.4.
THEREFORE, USE STARTING ADDRESS 204 OR 220 FOR DUAL PORT TESTING.

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 5, THEN TRACK ADDRESS SELECTION WILL EXCLUDE TRACKS 6 - 17 FROM THE SELECTION AND SELECT AN ADDRESS FROM AMONG ADDRESSES 18, 0, 1, 2, 3, 4, 5.
- B. THE BUFFER SIZE IS RANDOM SELECTED BETWEEN 4 (10) - AND THE VALUE IN 'MAXDL'. THE SIZE SELECTED IS WEIGHTED TO ENSURE THAT AT LEAST 4 WORDS ARE WRITTEN IN THE DATA AREA OF THE LAST SECTOR. THIS IS NECESSARY AS THE PROGRAM REQUIRES 4 LOCATIONS IN THE DATA PORTION OF THE SECTOR TO BE ABLE TO MATCH THE DATA TO A PATTERN FOR DATA COMPARISON PURPOSES.
- C. THE DATA WRITTEN IS RANDOMLY SELECTED AMONG THE 15 STANDARD PATTERNS. THE KEYWORDS IN THE HEADER (WHEN PERFORMING A WRITE HEADER & DATA ORDER) ARE ZERO FILLED. THE PROGRAM EXPECTS TO FIND THAT THE KEYWORDS ARE ZERO.
- D. THE ORDERS ARE SELECTED RANDOMLY. WRITE CHECK DATA AND WRITE CHECK HEADER & DATA ORDERS ARE PERFORMED ONLY IF THE PREVIOUS ORDER WAS THE APPROPRIATE DATA ORDER. IF THE 'FORMAT' PARAMETER IS ZERO, THE PROGRAM WILL NOT SELECT WRITE HEADER & DATA (AND WRITE CHECK HEADER & DATA) ORDERS. WHEN THE PROGRAM SELECTS A WRITE HEADER & DATA ORDER, THE BUFFER SIZE IS FORCED TO 260 (10); THE PROGRAM WILL NOT PERFORM A MULTI-SECTOR FORMAT WRITE OPERATION.
- E. THE FIRST ORDER PERFORMED AFTER A UNIT IS ASSIGNED WITH A 'T', 'W', OR 'R' COMMAND IS NOT RANDOMLY SELECTED. THE PARAMETERS FOR THE FIRST OPERATION ARE THE MINIMUM OR STARTING VALUES OF THE VARIABLES.

8.4 DATA PATTERNS

THE PROGRAM SELECTS ONE OF THE FOLLOWING DATA PATTERNS TO WRITE WHEN A WRITE ORDER IS SELECTED. THE ENTIRE BUFFER IS FILLED WITH THE SELECTED PATTERN. WHEN DATA IS READ FROM THE DISK, THE PROGRAM COMPARES DATA ON A SECTOR BASIS: FROM THE FIRST 4 DATA WORDS OF EACH SECTOR, THE PROGRAM MATCHES THE DATA TO ONE OF THE FOLLOWING PATTERNS. TO 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

@

1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828

```
.TITLE CZRJDDO, RP04/5/6 MLT-DR LGC
:*COPYRIGHT (C) 1975,1979
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
:*
.SBTTL OPERATIONAL SWITCH SETTINGS
:*
```


Line	SWITCH	USE
1829	15	HALT ON ERROR
1830	13	INHIBIT ERROR TYPEOUTS
1831	10	BELL ON ERROR
1832	7	DISPLAY ALL DATA COMPARE ERRORS
1833	6	DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
1834	5	A. PARTIAL REGISTER DISPLAY IF ERROR
1835	5	B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
1836	4	A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS
1837	4	B. DO NOT DROP DRIVE WHEN NORMAL END OF TEST REACHED
1838	3	A. DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR 'WCF' ERROR
1839	3	B. DISPLAY SECTOR IF 'DCK' ERR UNCORRECTABLE AFTER
1840	3	28TH RETRY
1841	3	C. IF DATA COMPARE ERROR & SW7 SET, DISPLAY
1842	3	REMAINDER OF BUFFER
1843	2	A. DON'T TYPE SUBSYSTEM STATUS WHEN PROGRAM STARTED
1844	2	B. DON'T TYPE PERFORMANCE SUMMARY
1845	1	INHIBIT DATA COMPARSION AFTER READ ORDERS
1846	1	INHIBIT DATA COMPARSION AFTER READ ORDERS
1847	0	READ ONLY MODE
1848	0	READ ONLY MODE
1849		.SBTTL BASIC DEFINITIONS
1850		
1851		;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
1852	001100	STACK= 1100
1853		.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
1854		.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
1855		
1856		;*MISCELLANEOUS DEFINITIONS
1857	000011	HT= 11 ;;CODE FOR HORIZONTAL TAB
1858	000012	LF= 12 ;;CODE FOR LINE FEED
1859	000015	CR= 15 ;;CODE FOR CARRIAGE RETURN
1860	000200	CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
1861	177776	PS= 177776 ;;PROCESSOR STATUS WORD
1862		.EQUIV PS,PSW
1863	177774	STKLMT= 177774 ;;STACK LIMIT REGISTER
1864	177772	PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
1865	177570	DSWR= 177570 ;;HARDWARE SWITCH REGISTER
1866	177570	DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
1867		
1868		;*GENERAL PURPOSE REGISTER DEFINITIONS
1869	000000	R0= %0 ;;GENERAL REGISTER
1870	000001	R1= %1 ;;GENERAL REGISTER
1871	000002	R2= %2 ;;GENERAL REGISTER
1872	000003	R3= %3 ;;GENERAL REGISTER
1873	000004	R4= %4 ;;GENERAL REGISTER
1874	000005	R5= %5 ;;GENERAL REGISTER
1875	000006	R6= %6 ;;GENERAL REGISTER
1876	000007	R7= %7 ;;GENERAL REGISTER
1877	000006	SP= %6 ;;STACK POINTER
1878	000007	PC= %7 ;;PROGRAM COUNTER
1879		
1880		;*PRIORITY LEVEL DEFINITIONS
1881	000000	PRO= 0 ;;PRIORITY LEVEL 0
1882	000040	PR1= 40 ;;PRIORITY LEVEL 1
1883	000100	PR2= 100 ;;PRIORITY LEVEL 2
1884	000140	PR3= 140 ;;PRIORITY LEVEL 3

1885	000200	PR4=	200	::PRIORITY LEVEL 4
1886	000240	PR5=	240	::PRIORITY LEVEL 5
1887	000300	PR6=	300	::PRIORITY LEVEL 6
1888	000340	PR7=	340	::PRIORITY LEVEL 7

;'SWITCH REGISTER' SWITCH DEFINITIONS

1890		SW15=	100000
1891	100000	SW14=	40000
1892	040000	SW13=	20000
1893	020000	SW12=	10000
1894	010000	SW11=	4000
1895	004000	SW10=	2000
1896	002000	SW09=	1000
1897	001000	SW08=	400
1898	000400	SW07=	200
1899	000200	SW06=	100
1900	000100	SW05=	40
1901	000040	SW04=	20
1902	000020	SW03=	10
1903	000010	SW02=	4
1904	000004	SW01=	2
1905	000002	SW00=	1
1906	000001	.EQUIV	SW09,SW9
1907		.EQUIV	SW08,SW8
1908		.EQUIV	SW07,SW7
1909		.EQUIV	SW06,SW6
1910		.EQUIV	SW05,SW5
1911		.EQUIV	SW04,SW4
1912		.EQUIV	SW03,SW3
1913		.EQUIV	SW02,SW2
1914		.EQUIV	SW01,SW1
1915		.EQUIV	SW00,SW0

;'DATA BIT DEFINITIONS (BIT00 TO BIT15)

1918		BIT15=	100000
1919	100000	BIT14=	40000
1920	040000	BIT13=	20000
1921	020000	BIT12=	10000
1922	010000	BIT11=	4000
1923	004000	BIT10=	2000
1924	002000	BIT09=	1000
1925	001000	BIT08=	400
1926	000400	BIT07=	200
1927	000200	BIT06=	100
1928	000100	BIT05=	40
1929	000040	BIT04=	20
1930	000020	BIT03=	10
1931	000010	BIT02=	4
1932	000004	BIT01=	2
1933	000002	BIT00=	1
1934	000001	.EQUIV	BIT09,BIT9
1935		.EQUIV	BIT08,BIT8
1936		.EQUIV	BIT07,BIT7
1937		.EQUIV	BIT06,BIT6
1938		.EQUIV	BIT05,BIT5
1939		.EQUIV	BIT04,BIT4
1940			


```
1941 .EQUIV BIT03,BIT3
1942 .EQUIV BIT02,BIT2
1943 .EQUIV BIT01,BIT1
1944 .EQUIV BIT00,BIT0
1945
1946 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
1947 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
1948 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
1949 000014 TBITVEC=14 ;: "T" BIT
1950 000014 TRIVEC= 14 ;:TRACE TRAP
1951 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
1952 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
1953 000024 PWRVEC= 24 ;:POWER FAIL
1954 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
1955 000034 TRAPVEC=34 ;:"TRAP" TRAP
1956 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
1957 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
1958 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
1959
1960 ;:*****
1961 .SBTTL RH11 REGISTERS !
1962
1963 ;:*****
1964
1965 ;CONTROL AND STATUS REGISTER 1 (RPCS1)
1966
1967
1968 000100 IE= 100 ;:INTERRUPT ENABLE (BIT #6)
1969 000200 RDY= 200 ;:READY (BIT #7)
1970 000400 A16= 400 ;:HIGH ORDER BUS ADDRESS BIT (BIT #8)
1971 001000 A17= 1000 ;:HIGH ORDER BUS ADDRESS BIT (BIT #9)
1972 002000 PSEL= 2000 ;:PORT SELECT (BIT #10)
1973 020000 MCPE= 20000 ;:MASSBUS PARITY ERROR (BIT #13)
1974 040000 TRE= 40000 ;:TRANSFER ERROR (BIT #14)
1975 ;SC= 100000 ;:SPECIAL CONDITION (BIT #15)
1976
1977 ;WORD COUNT REGISTER (RPWC)
1978 ;(EACH BIT IS CALLED BY BIT NUMBER)
1979
1980 ;BUS ADDRESS REGISTER (RPBA)
1981 ;(EACH BIT IS CALLED BY BIT NUMBER)
1982
1983 ;CONTROL AND STATUS REGISTER 2 (RPCS2)
1984
1985 000001 US1= 1 ;:UNIT SELECT (BIT #0)
1986 000002 US2= 2 ;:UNIT SELECT (BIT #1)
1987 000004 US4= 4 ;:UNIT SELECT (BIT #2)
1988 000010 BAI= 10 ;:BUS ADDRESS INCREMENT INHIBIT (BIT #3)
1989 000020 PAT= 20 ;:MASSBUS PARITY TEST (BIT #4)
1990 000040 CLR= 40 ;:CLEAR (BIT #5)
1991 000100 IR= 100 ;:INPUT READY (BIT #6)
1992 000200 OR= 200 ;:OUTPUT READY (BIT #7)
1993 000400 MPE= 400 ;:MASS BUS PARITY ERROR (BIT #8)
1994 001000 MXF= 1000 ;:MISSED TRANSFER ERROR (BIT #9)
1995 002000 PGE= 2000 ;:PROGRAM ERROR (BIT #10)
1996 004000 NEM= 4000 ;:NON EXISTENT MEMORY (BIT #11)
```



```
1997      010000      NED= 10000      ;NON EXISTENT DRIVE (BIT #12)
1998      020000      UPE= 20000      ;UNIBUS PARITY ERROR (BIT #13)
1999      040000      WCE= 40000      ;WRITE CHECK ERROR (BIT #14)
2000      100000      DLT= 100000     ;DATA LATE (BIT #15)
2001
2002      ;DATA BUFFER REGISTER (RPDB)
2003      ;(EACH BIT IS CALLED BY BIT NUMBER)
2004
2005
2006      ;:*****
2007
2008      .SBTTL RP04/5/6 REGISTERS
2009
2010      ;:*****
2011
2012      ;CONTROL AND STATUS 1 REGISTER. (#00)
2013
2014      000001      GO= 1           ;GO BIT (BIT #0)
2015      000002      F1= 2           ;FUNCTION CODE BIT #1
2016      000004      F2= 4           ;FUNCTION CODE BIT #2
2017      000010      F3= 10          ;FUNCTION CODE BIT #3
2018      000020      F4= 20          ;FUNCTION CODE BIT #4
2019      000040      F5= 40          ;FUNCTION CODE BIT #5
2020      004000      DVA= 4000       ;DEVICE AVAILABLE (BIT #11) *
2021
2022      ;DRIVE STATUS REGISTER (RPDS1) (#01)
2023
2024      ;DF5= 1      DRIVE FORWARD 5"/SEC. (BIT #0)
2025      000002      DFF20= 2      ;DRIVE FORWARD 20"/SEC. (BIT #1)
2026      000004      DIGB= 4      ;DRIVE TO INNER GUARD BAND (BIT #2)
2027      000010      GRV= 10      ;GO REVERSE (BIT #3)
2028      000020      DL64= 20      ;DIFFERENCE LESS THAN 64 (BIT #4)
2029      000040      DE1= 40      ;DIFFERENCE EQUALS 1 (BIT #5)
2030      000100      VV= 100      ;VOLUME VALID (BIT #6)
2031      000200      DRY= 200      ;DRIVE READY (BIT #7)
2032      000400      DPR= 400      ;DRIVE PRESENT (BIT #8)
2033      001000      PGM= 1000     ;PROGRAMABLE (BIT #9)
2034      002000      LST= 2000     ;LAST SECTOR TRANSFERRED (BIT #10)
2035      004000      WRL= 4000     ;WRITE LOCK (BIT #11)
2036      010000      MOL= 10000    ;MEDIUM ON-LINE (BIT #12)
2037      020000      PIP= 20000    ;POSITIONING OPERATION IN PROGRESS (BIT #13)
2038      040000      ERR= 40000    ;COMPOSITE ERROR (BIT #14)
2039      100000      ATA= 100000   ;ATTENTION ACTIVE (BIT #15)
2040
2041      ;ERROR REGISTER #01 (RPER1) (#02)
2042
2043      000001      ILF= 1         ;ILLEGAL FUNCTION (BIT #0)
2044      000002      ILR= 2         ;ILLEGAL REGISTER (BIT #1)
2045      000004      RMR= 4         ;REGISTER MODIFICATION REFUSED (BIT #2)
2046      000010      PAR= 10        ;PARITY ERROR (BIT #3) *
2047      000020      FER= 20        ;FORMAT ERROR (BIT #4)
2048      000040      WCF= 40        ;WRITE CLOCK FAIL (BIT #5)
2049      000100      ECH= 100       ;ECC HARD ERROR (BIT #6)
2050      000200      HCE= 200       ;HEADER COMPARE ERROR (BIT #7)
2051      000400      HCRC= 400      ;HEADER CRC ERROR (BIT #8)
2052      001000      AOE= 1000     ;ADDRESS OVERFLOW ERROR (BIT #9)
```


2053	002000	IAE= 2000	;INVALID ADDRESS ERROR (BIT #10)
2054	004000	WLE= 4000	;WRITE LOCK ERROR (BIT #11)
2055	010000	DTE= 10000	;DRIVE TIMING ERROR (BIT #12)
2056	020000	OPI= 20000	;OPERATION INCOMPLETE (BIT #13)
2057	040000	UNS= 40000	;DRIVE UNSAFE (BIT #14)
2058	100000	DCK= 100000	;DATA CHECK ERROR (BIT 15)
2059			
2060		;MAINTAINABILITY REGISTER (RPMR)(#03)	
2061			
2062	000001	DMD= 1	;DIAGINOSTIC MODE (BIT #0)
2063	000002	MCLK= 2	;MAINTAINABILITY CLOCK (BIT #1)
2064	000004	MINX= 4	;MAINTAINABILITY INDEX (BIT #2)
2065	000010	MSTCK= 10	;MAINTAINABILITY SECTOR CLOCK (BIT #3)
2066	000020	MRD= 20	;MAINTAINABILITY READ (BIT #4)
2067	000040	MWR= 40	;MAINTAINABILITY WRITE (BIT #5)
2068	000200	DTSY= 200	;MAINTAINABILITY SYNC DETECTED (BIT #7)
2069			
2070		;ATTENTION SUMMARY PSEUDO-REGISTER (RPAS) (#04)	
2071			
2072	000001	AT0= 1	;DEVICE 0 (BIT #0)
2073	000002	AT1= 2	;DEVICE 1 (BIT #1)
2074	000004	AT2= 4	;DEVICE 2 (BIT #2)
2075	000010	AT3= 10	;DEVICE 3 (BIT #3)
2076	000020	AT4= 20	;DEVICE 4 (BIT #4)
2077	000040	AT5= 40	;DEVICE 5 (BIT #5)
2078	000100	AT6= 100	;DEVICE 6 (BIT #6)
2079	000200	AT7= 200	;DEVICE 7 (BIT #7)
2080			
2081		;DESIRED SECTOR/TRACK ADDRESS REGISTER (RPDA) (#05)	
2082		;(EACH BIT IS CALLED BY BIT NUMBER)	
2083			
2084		;DRIVE TYPE REGISTER (RPDT) (#06)	
2085			
2086	000001	DT00= 1	;DRIVE TYPE NUMBER BIT 1
2087	000002	DT01= 2	;DRIVE TYPE NUMBER BIT 2
2088	000004	DT02= 4	;DRIVE TYPE NUMBER BIT 3
2089	000010	DT03= 10	;DRIVE TYPE NUMBER BIT 4
2090	000020	DT04= 20	;DRIVE TYPE NUMBER BIT 5
2091	000040	DT05= 40	;DRIVE TYPE NUMBER BIT 6
2092	000100	DT06= 100	;DRIVE TYPE NUMBER BIT 7
2093	000200	DT07= 200	;DRIVE TYPE NUMBER BIT 8
2094	000400	DT08= 400	;DRIVE TYPE NUMBER BIT 9
2095	004000	DRQ= 4000	;DRIVE REQUEST REQUIRED (BIT #11)
2096	020000	MOH= 20000	;MOVING HEAD (BIT #13)
2097	040000	TAP= 40000	;TAPE DRIVE (BIT #14)
2098	100000	NBA= 100000	;NOT BLOCK ADDRESSED (BIT #15)
2099			
2100		;LOOK-AHEAD REGISTER (RPLA) (#07)	
2101			
2102	000001	EXT1= 1	;EXTENSION 1 (BIT #0)
2103	000002	EXT2= 2	;EXTENSION 2 (BIT #1)
2104	000004	EXT4= 4	;EXTENSION 3 (BIT #2)
2105	000010	EXT10= 10	;EXTENSION 4 (BIT #3)
2106	000020	EXT20= 20	;EXTENSION 5 (BIT #4)
2107	000040	EXT40= 40	;EXTENSION 6 (BIT #5)
2108	000100	SC1= 100	;SECTOR COUNT FIELD 0 (BIT #6)

2109	000200	SC2= 200	; SECTOR COUNT FIELD 1 (BIT #7)
2110		; SC4= 400	; SECTOR COUNT FIELD 2 (BIT #8)
2111	001000	SC10= 1000	; SECTOR COUNT FIELD 3 (BIT #9)
2112	002000	SC20= 2000	; SECTOR COUNT FIELD 4 (BIT #10)
2113	004000	TRK1= 4000	; TRACK FIELD 1 (BIT #11)
2114	010000	TRK2= 10000	; TRACK FIELD 2 (BIT #12)
2115	020000	TRK4= 20000	; TRACK FIELD 3 (BIT #13)
2116	040000	TRK10= 40000	; TRACK FIELD 4 (BIT #14)
2117	100000	TRK20= 100000	; TRACK FIELD 5 (BIT #15)
2118			
2119		; RP04 ERROR REGISTER #2 (RPER2) (#10)	
2120			
2121	000001	WCU= 1	; WRITE CURRENT UNSAFE (BIT #0)
2122	000002	CSF= 2	; CURRENT SINK FAILURE (BIT #1)
2123	000004	WSU= 4	; WRITE SELECT UNSAFE (BIT #2)
2124	000010	CSU= 10	; CURRENT SWITCH UNSAFE (BIT #3)
2125	000020	MSE= 20	; MOTOR SEQUENCE ERROR (BIT #4)
2126	000040	TDF= 40	; TRANSITIONS DETECTOR FAILURE (BIT #5)
2127	000100	TUF= 100	; TRANSITIONS UNSAFE (BIT #6)
2128	000200	FEN= 200	; FAILSAFE ENABLED (BIT #7)
2129	000400	WRU= 400	; WRITE READY UNSAFE (BIT #8)
2130	001000	MHS= 1000	; MULTIPLE HEAD SELECT (BIT #9)
2131	002000	NHS= 2000	; NO HEAD SELECTION (BIT #10)
2132	004000	IXE= 4000	; INDEX ERROR (BIT #11)
2133	010000	VU30= 10000	; 30VOLT UNSAFE (BIT #12)
2134	020000	PLU= 20000	; PLO UNSAFE (BIT #13)
2135	100000	ACU= 100000	; AC UNSAFE (BIT #15)
2136			
2137		; RP05/6 ERROR REGISTER #02 (RPER2) (#10)	
2138			
2139	000001	WCU= 1	; WRITE CURRENT UNSAFE (BIT #0)
2140	000002	CSF= 2	; CURRENT SINK FAILURE (BIT #1)
2141	000004	WSU= 4	; WRITE SELECT UNSAFE (BIT #2)
2142	000010	CSU= 10	; CURRENT SWITCH UNSAFE (BIT #3)
2143	000020	RAW= 20	; READ AND WRITE (BIT #4)
2144	000040	TDF= 40	; TRANSITIONS DETECTOR FAILURE (BIT #5)
2145	000100	TUF= 100	; TRANSITIONS UNSAFE (BIT #6)
2146	000200	ABS= 200	; ABNORMAL STOP (BIT #7)
2147	000400	WRU= 400	; WRITE READY UNSAFE (BIT #8)
2148	001000	MHS= 1000	; MULTIPLE HEAD SELECT (BIT #9)
2149	002000	NHS= 2000	; NO HEAD SELECTION (BIT #10)
2150	004000	IXE= 4000	; INDEX ERROR (BIT #11)
2151	020000	PLU= 20000	; PLO UNSAFE (BIT #12)
2152			
2153		; OFFSET REGISTER (RPOF) (#11)	
2154			
2155	000001	OF25= 1	; OFFSET 25 MICRO INCHES (BIT #0)
2156	000002	OF50= 2	; OFFSET 50 MICRO INCHES (BIT #1)
2157	000004	OF100= 4	; OFFSET 100 MICRO INCHES (BIT #2)
2158	000010	OF200= 10	; OFFSET 200 MICRO INCHES (BIT #3)
2159	000020	OF400= 20	; OFFSET 400 MICRO INCHES (BIT #4)
2160	000040	OF800= 40	; OFFSET 800 MICRO INCHES (BIT #5)
2161	000200	OFREV= 200	; OFFSET NEGATIVE (REVERSE) (BIT #5)
2162	002000	HCI= 2000	; HEADER COMPARE INHIBIT (BIT #10)
2163	004000	ECl= 4000	; ERROR CORRECTION CODE INHIBIT (BIT #11)
2164	010000	FMT22= 10000	; FORMAT BIT (BIT #12)


```
2165  
2166 ;DESIRED CYLINDER ADDRESS (RPCA) (#12)  
2167 ;(EACH BIT IS CALLED BY BIT NUMBER)  
2168  
2169 ;CURRENT CYLINDER ADDRESS (RPCC) (#13)  
2170 ;(EACH BIT IS CALLED BY BIT NUMBER)  
2171  
2172 ;SERIAL NUMBER REGISTER (RPSN) (#14)  
2173 ;(EACH IS CALLED BY BIT NUMBER)  
2174  
2175 ;RP04 ERROR REGISTER #03 (RPER3) (#15)  
2176  
2177 000001 PSU= 1 ;PACK SPEED UNSAFE (BIT #0)  
2178 000002 VUF= 2 ;VELOCITY UNSAFE (BIT #1)  
2179 000010 UWR= 10 ;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)  
2180 000040 ACL= 40 ;AC LOW (BIT #5)  
2181 000100 DCL= 100 ;DC LOW (BIT #6)  
2182 040000 SKI= 40000 ;SEEK INCOMPLETE (BIT #14)  
2183 100000 GCYL= 100000 ;OFF CYLINDER (BIT #15)  
2184  
2185 ;RP05/6 ERROR REGISTER #03 (RPER3) (#15)  
2186  
2187 000001 DCU= 1 ;DC UNSAFE (BIT #0)  
2188 000002 WAO= 2 ;WRITE AND OFFSET (BIT #1)  
2189 000040 ACL= 40 ;AC LOW (BIT #5)  
2190 000100 DCL= 100 ;DC LOW (BIT #6)  
2191 020000 OPE= 20000 ;OPERATOR PLUG ERROR (BIT #13)  
2192 040000 SKI= 40000 ;SEEK INCOMPLETE (BIT #14)  
2193 100000 OCYL= 100000 ;OFF CYLINDER ERROR (BIT #15)  
2194  
2195 ;ECC POSITION REGISTER (RPEC1) (#16)  
2196 ;(EACH BIT IS CALLED BY BIT NUMBER)  
2197  
2198 ;ECC PATTERN REGISTER (RPEC2) (#17)  
2199 ;(EACH BIT IS CALLED BY BIT NUMBER)  
2200  
2201 ;:*****  
2202  
2203 .SBTTL RP04/5/6 DRIVER COMMANDS  
2204  
2205 ;:*****  
2206  
2207 000101 RNOP = 101 ;NO OPERATION  
2208 000103 UNLOAD = 103 ;UNLOAD  
2209 000105 SEEK = 105 ;SEEK  
2210 000107 RECAL = 107 ;RECALIBRATE  
2211 000111 DRVCLR = 111 ;DRIVE CLEAR  
2212 000113 RELSE = 113 ;RELEASE  
2213 000115 OFFSET = 115 ;OFFSET  
2214 000117 RTC = 117 ;RETURN TO CENTER LINE  
2215 000121 READIN = 121 ;READ IN PRESET  
2216 000123 ACK = 123 ;PACK ACKNOWLEDGE  
2217 000131 SEARCH = 131 ;SEARCH  
2218 000141 GETREG = 141 ;GET REGISTERS  
2219 000143 SETFMT = 143 ;SET FORMAT (& ECI OR HCI)  
2220 000145 SELDRV = 145 ;SELECT DRIVE
```



```
2221      000151      WCKD      =      151      ;WRITE CHECK DATA
2222      000153      WCKHD     =      153      ;WRITE CHECK HEADER & DATA
2223      000161      WRTDAT    =      161      ;WRITE DATA
2224      000163      WRTHD     =      163      ;WRITE HEADER & DATA
2225      000171      RDDAT     =      171      ;READ DATA
2226      000173      RDHD      =      173      ;READ HEADER & DATA
2227
2228      .SBTTL TRAP CATCHER
2229
2230      000000      .=0
2231      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
2232      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
2233      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
2234      .=174
2235      000174      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
2236      000176      000000      SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER
2237      .SBTTL STARTING ADDRESS(ES)
2238      000200      000137      004140      JMP      @#START!    ;;JUMP TO STARTING ADDRESS OF PROGRAM
2239      000204      000137      004122      JMP      @#START    ;CHANGE THE R#11 UNIBUS ADDRESS
```


CZRJDDO, RP04/5/6 MLT-DR LGC MACY11 30A(1052) 03-JUL-79 10:41 F 4 PAGE 46
CZRJDD.P11 06-JUN-79 08:32 STARTING ADDRESS(ES)

SEQ 0044

2240

;AFTER INITIAL START

CZRJDDO, RP04/5/6 MLT-DR LGC MACY11 30A(1052) 03-JUL-79 10:41 ^{6 4} PAGE 47
CZRJDD.P11 06-JUN-79 08:32 STARTING ADDRESS(ES)

SEQ 0045

2241

;DO NOT INHIBIT PROGRAMMABLE DRIVES


```
2242          000220          .=220
2243 000220 000137 004106   JMP   @#START3      ; SAME AS 200 WITHOUT INHIBITING PROGRAMMABLE DRIVES
2244
2245          .SBTTL  ACT11 HOOKS
2246
2247          ;:*****
2248          ;HOOKS REQUIRED BY ACT11
2249          000224          $SVPC=.          ; SAVE PC
2250          000046          .=46
2251 000046 005446          $ENDAD          ;; 1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
2252          000052          .=52
2253 000052 040000          .WORD 40000      ;; 2)SET LOC.52 TO 40000
2254          000224          .= $SVPC          ;; RESTORE PC
```


Address	Tag	Value	Tag	Description
2255			.SBTTL	COMMON TAGS
2256				
2257				*****
2258				;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
2259				;*USED IN THE PROGRAM.
2260				
2261		001100		
2262	001100		\$CMTAG:	.=1100
2263	001100	000000	\$PASS:	.WORD 0
2264	001102	000	\$TSTNM:	.BYTE 0
2265	001103	000	\$ERFLG:	.BYTE 0
2266	001104	000000	\$ICNT:	.WORD 0
2267	001106	000000	\$LPADR:	.WORD 0
2268	001110	000000	\$LPERR:	.WORD 0
2269	001112	000000	\$ERTTL:	.WORD 0
2270	001114	000	\$ITEMB:	.BYTE 0
2271	001115	001	\$ERMAX:	.BYTE 1
2272	001116	000000	\$ERRPC:	.WORD 0
2273	001120	000000	\$GDADR:	.WORD 0
2274	001122	000000	\$BDADR:	.WORD 0
2275	001124	000000	\$GDDAT:	.WORD 0
2276	001126	000000	\$BDDAT:	.WORD 0
2277	001130	000000		.WORD 0
2278	001132	000000		.WORD 0
2279	001134	000	\$AUTOB:	.BYTE 0
2280	001135	000	\$INTAG:	.BYTE 0
2281	001136	000000		.WORD 0
2282	001140	177570	\$SWR:	.WORD DSWR
2283	001142	177570	\$DISPLAY:	.WORD DDISP
2284	001144	177560	\$TKS:	177560
2285	001146	177562	\$TKB:	177562
2286	001150	177564	\$TPS:	177564
2287	001152	177566	\$TPB:	177566
2288	001154	000	\$NULL:	.BYTE 0
2289	001155	002	\$FILLS:	.BYTE 2
2290	001156	012	\$FILLC:	.BYTE 12
2291	001157	000	\$TPFLG:	.BYTE 0
2292	001160	177607	\$BELL:	.ASCIZ <207><377><377>
2293	001164	077	\$QUES:	.ASCII /?/
2294	001165	015	\$CRLF:	.ASCII <15>
2295	001166	000012	\$LF:	.ASCIZ <12>
2296				*****
2297		000015	CR	= 15
2298		000012	LF	= 12
2299	001170	176700	\$RPADR:	.WORD 176700
2300	001172	000254	\$RPVEC:	.WORD 254
2301	001174	172540	\$LKCSR:	.WORD 172540
2302	001176	172542	\$LKCSB:	.WORD 172542
2303	001200	000104	\$LPVEC:	.WORD 104
2304	001202	177546	\$LKS:	.WORD 177546
2305	001204	000100	\$LLVEC:	.WORD 100
2306	001206	177777	PCLOCK:	.WORD -1
2307	001210	177777	CLKFLG:	.WORD -1
2308	001212	000074	HZ:	.WORD 74
2309	001214	000000	STATIN:	.WORD 0
2310	001216	000000	PACK:	.WORD 0

2311	001220	000000	000000	000000	DATE: .WORD	0,0,0,0,0	;OPERATOR ENTERED DATE
2312	001226	000000	000000				
2313	001232	000000	000000	000000	OPERID: .WORD	0,0,0,0	;OPERATOR ID
2314	001240	000000					
2315	001242	000000			DRIVE: .WORD	0	;DRIVE # STORAGE: ERRORS 1-5 & 10
2316	001244	000000			ATTN: .WORD	0	;ATTN REG STORAGE: ERRORS 1-5 & 10
2317	001246	000000			UNIT: .WORD	0	;DRIVE # STORAGE FOR PRINTOUT
2318	001250	000000			MASK: .WORD	0	;ERROR RETRY REGISTER MASK
2319	001252	000	000		RETRY: .BYTE	0,0	;ERROR RETRY LIMIT IN THE LOWER BYTE
2320							;RETRY COUNT IN THE UPPER BYTE
2321	001254	000003			FAIRNS: .WORD	3	;MAXIMUM TIME IN QUEUE VALUE
2322	001256	000000			LSTAD: .WORD	0	;STORE LAST MEMORY ADDRESS HERE
2323	001260	000000			CHGADR: .WORD	0	;CHANGE RH11 UNIBUS ADDRESS FLAG
2324	001262	000000			CFLAG: .WORD	0	; 'CONTROL C' FLAG
2325	001264	000000			BADSEC: .WORD	0	;BAD SECTOR/TRACK FLAG
2326	001266	000000			HOUR: .WORD	0	;HOUR COUNT STORED HERE (MAXIMUM - 999.)
2327	001270	000000			MINUTE: .WORD	0	;MINUTE'S COUNT STORED HERE
2328	001272	000000			SECOND: .WORD	0	;SECOND'S COUNT STORED HERE
2329	001274	000000			SIXTEE: .WORD	0	;TIMER ROUTINE COUNTER (FOR ONE SECOND)
2330	001276	177777			ZROIND: .WORD	-1	;ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
2331	001300	000			FRSTER: .BYTE	0	;DATA COMPARE ERROR FLAG
2332							;IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
2333							;IF < 0, MISCOMPARSION FOUND
2334	001301	000			.BYTE	0	;MISCOMPARSION OR CAN'T MATCH PATTERN FLAG
2335							;IF < 0, ERROR IN BUFFER
2336	001302	000000			SAVER1: .WORD	0	;SAVE R1 HERE
2337	001304	000000			SAVER5: .WORD	0	;SAVE R5 HERE
2338	001306	000000			ERCTR: .WORD	0	;NUMBER OF ERRORS
2339	001310	000000			LIMIT: .WORD	0	;DISPLAY LIMIT
2340	001312	000000			CMCNT: .WORD	0	;WORD COUNT
2341	001314	000000			CMCYL: .WORD	0	;CYLINDER ADDRESS
2342	001316	000			CMSEC: .BYTE	0	;SECTOR ADDRESS
2343	001317	000			CMTRK: .BYTE	0	;TRACK ADDRESS
2344	001320	000000			ECBIT: .WORD	0	;ERROR BURST BIT OFFSET
2345	001322	000000			ECSEC: .WORD	0	;ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
2346	001324	000000			ECMSK0: .WORD	0	;CORRECTION MASK FOR FIRST ERROR WORD
2347	001326	000000			ECMSK1: .WORD	0	;CORRECTION MASK FOR SECOND ERROR WORD
2348	001330	000000			ECWRD: .WORD	0	;LOCATION OF FIRST ERROR WORD
2349	001332	000000			ECGD: .WORD	0	;GOOD DATA, FIRST WORD
2350	001334	000000			ECBAD0: .WORD	0	;BAD DATA, FIRST WORD
2351	001336	000000			ECWRD1: .WORD	0	;LOCATION OF SECOND ERROR WORD
2352	001340	000000			ECGD1: .WORD	0	;GOOD DATA, SECOND WORD
2353	001342	000000			ECBAD1: .WORD	0	;BAD DATA, SECOND WORD
2354	001344	000025			SECLMT: .WORD	21.	;SECTOR ADDRESS LIMIT
2355	001346	000022			TRKLMT: .WORD	18.	;TRACK ADDRESS LIMIT
2356	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
 ENDCN: .WORD 002740 ;1.875x10^8 WORDS (10) [3x10^9 BITS]

2366 001352 002740


```
2367 001354 005455          :MSW
2368 001356 143300          :3 X 10^6 SEEKS (LSW)
2369 001360 000055          :MSW
2370          :PROGRAM USES THESE PARAMETERS TO DETERMINE Q.V. END OF PASS
2371 001362 120274          :2.3437X10^7 WORDS (10)
2372 001364 000005          :MSW
2373 001366 134330          :3.75 X 10^5 SEEKS (10)
2374 001370 000005          :MSW
2375          :THE NUMBERS TO DETERMINE END OF PASS ARE LOADED IN HERE BY THE PROGRAM.
2376          :THE FIRST TIME THROUGH, THE QV PARAMETERS ARE USED, AFTER THAT THE
2377          :REGULAR PARAMETERS ARE USED.
2378 001372 000000          :
2379 001374 000000          :
2380 001376 000000          :
2381 001400 000000          :
2382          :
2383 001402 000001          :NUMBER OF PASSES TO END OF TEST
2384 001404 000000          :MAXIMUM DATA TRANSFER SIZE IN WORDS
2385          : (FILLED BY PROGRAM AT STARTUP OR BY OPERATOR
2386          : DURING PARAMETER ENTRY DIALOG.)
2387 001406 000144          :MAXIMUM ERRORS - 100(10)
2388 001410 000005 000000  :FIRST WORD IS THE PERFORMANCE TYPEOUT INTERVAL
2389          : (IN MINUTES). SECOND WORD IS THE INTERVAL
2390          :COUNTER
2391          :COUNTER. UPPER BYTE IS VALUE.
2392 001414 000004          :NUMBER OF COMPARE ERRORS TYPED OUT
2393 001416 000001          :IF NOT EQ 0, ALLOW WRITE HEADER & DATA ORDERS
2394          :IF EQ 0, DO NOT ALLOW WRITE HEADER & DATA ORDERS
2395 001420 000000          :IF EQ TO 0, GENERATE A RANDOM WORD COUNT
2396          : FOR THE OPERATION.
2397          :IF NOT EQ TO 0, USE THE VALUE IN 'MAXDL' FOR
2398          : THE WORD COUNT
2399 001422 000003          :READ/WRITE RATIO [RANGE 0 - 7]
2400          :0 - 0/8 (READ/WRITE)
2401          :1 - 7/1
2402          :2 - 6/2
2403          :3 - 5/3
2404          :4 - 4/4
2405          :5 - 3/5
2406          :6 - 2/6
2407          :7 - 1/7
2408 001424 000001          :IF NOT EQ 0, DO AN APPROPRIATE WRITE
2409          : CHECK AFTER EACH WRITE ORDER.
2410          :IF EQ 0, SELECT WRITE CHECK ORDERS
2411          : RANDOMLY.
2412 001426 000001          :IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES
2413          : ASSOCIATED WITH OPERATOR SPECIFIED
2414          : BAD PACK AREAS.
2415          :IF NOT EQ 0, PRINT ERROR MESSAGES RELATING TO
2416          : THESE AREAS.
2417 001430 000001          :IF NOT EQ 0, END OF PASS DETERMINED
2418          : BY THE 'WORDS READ' COUNT.
2419          :IF EQ 0, END OF PASS DETERMINED
2420          : BY THE SEEK COUNT.
2421          :
2422          :*****
```


2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478

.SBTTL VALUES FOR FIRST OPERATION

::*****

BEGPAT: .WORD 10 ;STARTING PATTERN CODE [RANGE 1 - 17 (OCTAL)]
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')
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

::*****

ORDERQ: .WORD 0,0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES PERFORMING COMMANDS
ASNLS: .WORD 0 ;A BIT SET IS AN ASSIGNED DRIVE
DUNIT: .WORD 0,0,0,0,0,0,0,0,0,0 ;ADDRESSES OF DRIVES TO BE DEASSIGNED
NEWUNT: .WORD 0,0,0,0,0,0,0,0,0,0 ;ADDRESSES OF NEWLY ASSIGNED DRIVES
AVAIL: .WORD 0,0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES WAITING FOR BUFFERS/PARAMETERS
WAIT: .WORD 0,0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES WAITING FOR BUFFERS
PARQ: .WORD 0,0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES WAITING FOR NEXT PARAMETERS
BUFTBL: .WORD 0 ;BUFFER ALLOCATION TABLE ENTRY COUNT
.WORD 0,0

2479	001624	000000	000000	.WORD	0,0	
2480	001630	000000	000000	.WORD	0,0	
2481	001634	000000	000000	.WORD	0,0	
2482	001640	000000	000000	.WORD	0,0	
2483	001644	000000	000000	.WORD	0,0	
2484	001650	000000	000000	.WORD	0,0	
2485	001654	000000	000000	.WORD	0,0	
2486	001660	000000	000000	.WORD	0,0	
2487	001664	000000	000000	.WORD	0,0	
2488	001670	000000	000000	.WORD	0,0	
2489	001674	000000	000000	.WORD	0,0	
2490	001700	000000	000000	.WORD	0,0	
2491	001704	000000	000000	.WORD	0,0	
2492	001710	000000	000000	.WORD	0,0	
2493	001714	000000	000000	.WORD	0,0	
2494	001720	000000	000000	.WORD	0,0	
2495	001724	000000	000000	.WORD	0,0	
2496	001730	000000	000000	.WORD	0,0	
2497	001734	000000	000000	.WORD	0,0	
2498						
2499	001740	042036		BLKADR: .WORD	DRIVE0	: ADDRESS OF THE BLOCK FOR DRIVE 0
2500	001742	042342		.WORD	DRIVE1	: ADDRESS OF THE BLOCK FOR DRIVE 1
2501	001744	042646		.WORD	DRIVE2	: ADDRESS OF THE BLOCK FOR DRIVE 2
2502	001746	043152		.WORD	DRIVE3	: ADDRESS OF THE BLOCK FOR DRIVE 3
2503	001750	043456		.WORD	DRIVE4	: ADDRESS OF THE BLOCK FOR DRIVE 4
2504	001752	043762		.WORD	DRIVE5	: ADDRESS OF THE BLOCK FOR DRIVE 5
2505	001754	044266		.WORD	DRIVE6	: ADDRESS OF THE BLOCK FOR DRIVE 6
2506	001756	044572		.WORD	DRIVE7	: ADDRESS OF THE BLOCK FOR DRIVE 7
2507						
2508	001760	151		COMTBL: .BYTE	WCKD	: WRITE CHECK DATA
2509	001761	153		.BYTE	WCKHD	: WRITE CHECK HEADER AND DATA
2510	001762	161		.BYTE	WRDAT	: WRITE DATA
2511	001763	163		.BYTE	WRTHD	: WRITE HEADER AND DATA
2512	001764	171		.BYTE	RDDAT	: READ DATA
2513	001765	173		.BYTE	RDHD	: READ HEADER AND DATA
2514						
2515	001766	002		OPTBL: .BYTE	2	: UNLOAD
2516	001767	004		.BYTE	4	: SEEK
2517	001770	006		.BYTE	6	: RECAL
2518	001771	010		.BYTE	10	: DRIVE CLEAR
2519	001772	012		.BYTE	12	: RELEASE
2520	001773	014		.BYTE	14	: OFFSET
2521	001774	016		.BYTE	16	: RETURN TO CENTERLINE
2522	001775	020		.BYTE	20	: READIN PRESET
2523	001776	022		.BYTE	22	: PACK ACKNOWLEDGE
2524	001777	030		.BYTE	30	: SEARCH
2525	002000	050		.BYTE	50	: WRITE CHECK DATA
2526	002001	052		.BYTE	52	: WRITE CHECK HEADER AND DATA
2527	002002	060		.BYTE	60	: WRITE DATA
2528	002003	062		.BYTE	62	: WRITE HEADER AND DATA
2529	002004	070		.BYTE	70	: READ DATA
2530	002005	072		.BYTE	72	: READ HEADER AND DATA
2531	002006	377		.BYTE	-1	: TERMINATOR
2532						
2533	002010			.EVEN		
2534						

2535	002010	047125	047514	042101	MNTBL:	.ASCIZ	/UNLOAD	/
2536	002016	000040						
2537	002020	042523	045505	020040		.ASCIZ	/SEEK	/
2538	002026	000040						
2539	002030	042522	040503	020114		.ASCIZ	/RECAL	/
2540	002036	000040						
2541	002040	051104	041526	051114		.ASCIZ	/DRVCLR	/
2542	002046	000040						
2543	002050	042522	051514	020105		.ASCIZ	/RELSE	/
2544	002056	000040						
2545	002060	043117	051506	052105		.ASCIZ	/OFFSET	/
2546	002066	000040						
2547	002070	052122	020103	020040		.ASCIZ	/RTC	/
2548	002076	000040						
2549	002100	042522	042101	047111		.ASCIZ	/READIN	/
2550	002106	000040						
2551	002110	040520	045503	020040		.ASCIZ	/PACK	/
2552	002116	000040						
2553	002120	042523	051101	044103		.ASCIZ	/SEARCH	/
2554	002126	000040						
2555	002130	041527	042113	020040		.ASCIZ	/WCKD	/
2556	002136	000040						
2557	002140	041527	044113	020104		.ASCIZ	/WCKHD	/
2558	002146	000040						
2559	002150	051127	042124	052101		.ASCIZ	/WRDAT	/
2560	002156	000040						
2561	002160	051127	044124	020104		.ASCIZ	/WRTHD	/
2562	002166	000040						
2563	002170	042122	040504	020124		.ASCIZ	/RDDAT	/
2564	002176	000040						
2565	002200	042122	042110	020040		.ASCIZ	/RDHD	/
2566	002206	000040						
2567	002210	047516	042516	020040		.ASCIZ	/NONE	/
2568	002216	000040						
2569								

2570	002220	000			OFFCOD:	.BYTE	0	:OFFSET CODE TABLE
2571	002221	010				.BYTE	10	:+200 U INCHES
2572	002222	210				.BYTE	210	:-200 U INCHES
2573	002223	020				.BYTE	20	:+400 U INCHES
2574	002224	220				.BYTE	220	:-400 U INCHES
2575	002225	030				.BYTE	30	:+600 U INCHES
2576	002226	230	000			.BYTE	230,0	:-600 U INCHES, TERMINATOR
2577	002230	020				.BYTE	20	:+400 U INCHES
2578	002231	220				.BYTE	220	:-400 U INCHES
2579	002232	040				.BYTE	40	:+800 U INCHES
2580	002233	240				.BYTE	240	:-800 U INCHES
2581	002234	060				.BYTE	60	:+1200 U INCHES
2582	002235	260	000			.BYTE	260,0	:-1200 U INCHES, TERMINATOR
2583		002240			.EVEN			
2584								
2585	002240	002274			OFMTBL:	.WORD	OFMSG0	:1ST OFFSET MESSAGE
2586	002242	002327				.WORD	OFMSG1	:2ND OFFSET MESSAGE
2587	002244	002363				.WORD	OFMSG2	:3RD OFFSET MESSAGE
2588	002246	002417				.WORD	OFMSG3	:4TH OFFSET MESSAGE
2589	002250	002453				.WORD	OFMSG4	:5TH OFFSET MESSAGE
2590	002252	002507				.WORD	OFMSG5	:6TH OFFSET MESSAGE

2591	002254	002543				.WORD	OFMSG6		:7TH OFFSET MESSAGE
2592	002256	002274				.WORD	OFMSG0		:1ST OFFSET MESSAGE
2593	002260	002417				.WORD	OFMSG3		:4TH OFFSET MESSAGE
2594	002262	002453				.WORD	OFMSG4		:5TH OFFSET MESSAGE
2595	002264	002577				.WORD	OFMSG7		:8TH OFFSET MESSAGE
2596	002266	002633				.WORD	OFMSG8		:9TH OFFSET MESSAGE
2597	002270	002667				.WORD	OFMSG9		:10TH OFFSET MESSAGE
2598	002272	002724				.WORD	OFMSGA		:11TH OFFSET MESSAGE
2599									
2600	002274	043101	042524	020122	OFMSG0:	.ASCIZ	/AFTER RETRY. WITHOUT OFFSET/		
2601	002302	042522	051124	020131					
2602	002310	044527	044124	052517					
2603	002316	020124	043117	051506					
2604	002324	052105	000						
2605	002327	101	020124	043117	OFMSG1:	.ASCIZ	/AT OFFSET +200 MICRO-INCHES/		
2606	002334	051506	052105	025440					
2607	002342	030062	020060	044515					
2608	002350	051103	026517	047111					
2609	002356	044103	051505	000					
2610	002363	101	020124	043117	OFMSG2:	.ASCIZ	/AT OFFSET -200 MICRO-INCHES/		
2611	002370	051506	052105	026440					
2612	002376	030062	020060	044515					
2613	002404	051103	026517	047111					
2614	002412	044103	051505	000					
2615	002417	101	020124	043117	OFMSG3:	.ASCIZ	/AT OFFSET +400 MICRO-INCHES/		
2616	002424	051506	052105	025440					
2617	002432	030064	020060	044515					
2618	002440	051103	026517	047111					
2619	002446	044103	051505	000					
2620	002453	101	020124	043117	OFMSG4:	.ASCIZ	/AT OFFSET -400 MICRO-INCHES/		
2621	002460	051506	052105	026440					
2622	002466	030064	020060	044515					
2623	002474	051103	026517	047111					
2624	002502	044103	051505	000					
2625	002507	101	020124	043117	OFMSG5:	.ASCIZ	/AT OFFSET +600 MICRO-INCHES/		
2626	002514	051506	052105	025440					
2627	002522	030066	020060	044515					
2628	002530	051103	026517	047111					
2629	002536	044103	051505	000					
2630	002543	101	020124	043117	OFMSG6:	.ASCIZ	/AT OFFSET -600 MICRO-INCHES/		
2631	002550	051506	052105	026440					
2632	002556	030066	020060	044515					
2633	002564	051103	026517	047111					
2634	002572	044103	051505	000					
2635	002577	101	020124	043117	OFMSG7:	.ASCIZ	/AT OFFSET +800 MICRO-INCHES/		
2636	002604	051506	052105	025440					
2637	002612	030070	020060	044515					
2638	002620	051103	026517	047111					
2639	002626	044103	051505	000					
2640	002633	101	020124	043117	OFMSG8:	.ASCIZ	/AT OFFSET -800 MICRO-INCHES/		
2641	002640	051506	052105	026440					
2642	002646	030070	020060	044515					
2643	002654	051103	026517	047111					
2644	002662	044103	051505	000					
2645	002667	101	020124	043117	OFMSG9:	.ASCIZ	/AT OFFSET +1200 MICRO-INCHES/		
2646	002674	051506	052105	025440					

2647 002702 031061 030060 046440
2648 002710 041511 047522 044455
2649 002716 041516 042510 000123
2650 002724 052101 047440 043106
2651 002732 042523 020124 030455
2652 002740 030062 020060 044515
2653 002746 051103 026517 047111
2654 002754 044103 051505 000

OFMSGA: .ASCIZ /AT OFFSET -1200 MICRO-INCHES/

2655
2656 002762

.EVEN

2657
2658

.SBTTL DATA PATTERNS

2659
2660

2661
2662

STNDAT: .WORD 0 ;STANDARD DATA PATTERN POINTER TABLE

2663
2664 002762 000000

.WORD DATA1

2665 002764 003066

.WORD DATA1+40

2666 002766 003126

.WORD DATA1+100

2667 002770 003166

.WORD DATA1+140

2668 002772 003226

.WORD DATA1+200

2669 002774 003266

.WORD DATA1+240

2670 002776 003326

.WORD DATA1+300

2671 003000 003366

.WORD DATA1+340

2672 003002 003426

.WORD DATA1+400

2673 003004 003466

.WORD DATA1+440

2674 003006 003526

.WORD DATA1+500

2675 003010 003566

.WORD DATA1+540

2676 003012 003626

.WORD DATA1+600

2677 003014 003666

.WORD DATA1+640

2678 003016 003726

.WORD DATA1+700

2679 003020 003766

.WORD DATA10

2680 003022 003026

.WORD DATA1+642

2681 003024 003730

;ZER0ES

;ONES

2682
2683 003026 000000

DATA0: .WORD 0 ;DUMMY DATA PATTERN

2684 003030 000000

.WORD 0

2685 003032 000000

.WORD 0

2686 003034 000000

.WORD 0

2687 003036 000000

.WORD 0

2688 003040 000000

.WORD 0

2689 003042 000000

.WORD 0

2690 003044 000000

.WORD 0

2691 003046 000000

.WORD 0

2692 003050 000000

.WORD 0

2693 003052 000000

.WORD 0

2694 003054 000000

.WORD 0

2695 003056 000000

.WORD 0

2696 003060 000000

.WORD 0

2697 003062 000000

.WORD 0

2698 003064 000000

.WORD 0

2699

DATA1: .WORD 000001 ;STANDARD PATTERN 1

2700 003066 000001

.WORD 000003

2701 003070 000003

.WORD 000007

2702 003072 000007

2703	003074	000017	.WORD	000017	
2704	003076	000037	.WORD	000037	
2705	003100	000077	.WORD	000077	
2706	003102	000177	.WORD	000177	
2707	003104	000377	.WORD	000377	
2708	003106	000777	.WORD	000777	
2709	003110	001777	.WORD	001777	
2710	003112	003777	.WORD	003777	
2711	003114	007777	.WORD	007777	
2712	003116	017777	.WORD	017777	
2713	003120	037777	.WORD	037777	
2714	003122	077777	.WORD	077777	
2715	003124	177777	.WORD	177777	
2716					
2717	003126	177776	.WORD	177776	; STANDARD PATTERN 2
2718	003130	177774	.WORD	177774	
2719	003132	177770	.WORD	177770	
2720	003134	177760	.WORD	177760	
2721	003136	177740	.WORD	177740	
2722	003140	177700	.WORD	177700	
2723	003142	177600	.WORD	177600	
2724	003144	177400	.WORD	177400	
2725	003146	177000	.WORD	177000	
2726	003150	176000	.WORD	176000	
2727	003152	174000	.WORD	174000	
2728	003154	170000	.WORD	170000	
2729	003156	160000	.WORD	160000	
2730	003160	140000	.WORD	140000	
2731	003162	100000	.WORD	100000	
2732	003164	000000	.WORD	000000	
2733					
2734	003166	000000	.WORD	000000	; STANDARD PATTERN 3
2735	003170	000000	.WORD	000000	
2736	003172	000000	.WORD	000000	
2737	003174	177777	.WORD	177777	
2738	003176	177777	.WORD	177777	
2739	003200	177777	.WORD	177777	
2740	003202	000000	.WORD	000000	
2741	003204	000000	.WORD	000000	
2742	003206	177777	.WORD	177777	
2743	003210	177777	.WORD	177777	
2744	003212	000000	.WORD	000000	
2745	003214	177777	.WORD	177777	
2746	003216	000000	.WORD	000000	
2747	003220	177777	.WORD	177777	
2748	003222	000000	.WORD	000000	
2749	003224	177777	.WORD	177777	
2750					
2751	003226	000000	.WORD	000000	; STANDARD PATTERN 4
2752	003230	010421	.WORD	010421	
2753	003232	021042	.WORD	021042	
2754	003234	031463	.WORD	031463	
2755	003236	042104	.WORD	042104	
2756	003240	052525	.WORD	052525	
2757	003242	063146	.WORD	063146	
2758	003244	073567	.WORD	073567	

2759	003246	104210	.WORD	104210	
2760	003250	114631	.WORD	114631	
2761	003252	125252	.WORD	125252	
2762	003254	135673	.WORD	135673	
2763	003256	146314	.WORD	146314	
2764	003260	156735	.WORD	156735	
2765	003262	167356	.WORD	167356	
2766	003264	177777	.WORD	177777	
2767					
2768	003266	052525	.WORD	052525	; STANDARD PATTERN 5
2769	003270	052525	.WORD	052525	
2770	003272	052525	.WORD	052525	
2771	003274	125252	.WORD	125252	
2772	003276	125252	.WORD	125252	
2773	003300	125252	.WORD	125252	
2774	003302	052525	.WORD	052525	
2775	003304	052525	.WORD	052525	
2776	003306	125252	.WORD	125252	
2777	003310	125252	.WORD	125252	
2778	003312	052525	.WORD	052525	
2779	003314	125252	.WORD	125252	
2780	003316	052525	.WORD	052525	
2781	003320	125252	.WORD	125252	
2782	003322	052525	.WORD	052525	
2783	003324	125252	.WORD	125252	
2784					
2785	003326	007417	.WORD	007417	; STANDARD PATTERN 6
2786	003330	007417	.WORD	007417	
2787	003332	007417	.WORD	007417	
2788	003334	170360	.WORD	170360	
2789	003336	170360	.WORD	170360	
2790	003340	170360	.WORD	170360	
2791	003342	007417	.WORD	007417	
2792	003344	007417	.WORD	007417	
2793	003346	170360	.WORD	170360	
2794	003350	170360	.WORD	170360	
2795	003352	007417	.WORD	007417	
2796	003354	170360	.WORD	170360	
2797	003356	007417	.WORD	007417	
2798	003360	170360	.WORD	170360	
2799	003362	007417	.WORD	007417	
2800	003364	170360	.WORD	170360	
2801					
2802	003366	026455	.WORD	026455	; STANDARD PATTERN 7
2803	003370	026455	.WORD	026455	
2804	003372	026455	.WORD	026455	
2805	003374	151322	.WORD	151322	
2806	003376	151322	.WORD	151322	
2807	003400	151322	.WORD	151322	
2808	003402	026455	.WORD	026455	
2809	003404	026455	.WORD	026455	
2810	003406	151322	.WORD	151322	
2811	003410	151322	.WORD	151322	
2812	003412	026455	.WORD	026455	
2813	003414	151322	.WORD	151322	
2814	003416	026455	.WORD	026455	

2815	003420	151322	.WORD	151322	
2816	003422	026455	.WORD	026455	
2817	003424	151322	.WORD	151322	
2818					
2819	003426	165555	.WORD	165555	; STANDARD PATTERN 8
2820	003430	133333	.WORD	133333	
2821	003432	165555	.WORD	165555	
2822	003434	133333	.WORD	133333	
2823	003436	165555	.WORD	165555	
2824	003440	133333	.WORD	133333	
2825	003442	165555	.WORD	165555	
2826	003444	133333	.WORD	133333	
2827	003446	165555	.WORD	165555	
2828	003450	133333	.WORD	133333	
2829	003452	165555	.WORD	165555	
2830	003454	133333	.WORD	133333	
2831	003456	165555	.WORD	165555	
2832	003460	133333	.WORD	133333	
2833	003462	165555	.WORD	165555	
2834	003464	133333	.WORD	133333	
2835					
2836	003466	000001	.WORD	000001	; STANDARD PATTERN 9
2837	003470	000002	.WORD	000002	
2838	003472	000004	.WORD	000004	
2839	003474	000010	.WORD	000010	
2840	003476	000020	.WORD	000020	
2841	003500	000040	.WORD	000040	
2842	003502	000100	.WORD	000100	
2843	003504	000200	.WORD	000200	
2844	003506	000400	.WORD	000400	
2845	003510	001000	.WORD	001000	
2846	003512	002000	.WORD	002000	
2847	003514	004000	.WORD	004000	
2848	003516	010000	.WORD	010000	
2849	003520	020000	.WORD	020000	
2850	003522	040000	.WORD	040000	
2851	003524	100000	.WORD	100000	
2852					
2853	003526	177776	.WORD	177776	; STANDARD PATTERN 10
2854	003530	177775	.WORD	177775	
2855	003532	177773	.WORD	177773	
2856	003534	177767	.WORD	177767	
2857	003536	177757	.WORD	177757	
2858	003540	177737	.WORD	177737	
2859	003542	177677	.WORD	177677	
2860	003544	177577	.WORD	177577	
2861	003546	177377	.WORD	177377	
2862	003550	176777	.WORD	176777	
2863	003552	175777	.WORD	175777	
2864	003554	173777	.WORD	173777	
2865	003556	167777	.WORD	167777	
2866	003560	157777	.WORD	157777	
2867	003562	137777	.WORD	137777	
2868	003564	077777	.WORD	077777	
2869					
2870	003566	172666	.WORD	172666	; STANDARD PATTERN 11

2871	003570	155555	.WORD	155555	
2872	003572	172666	.WORD	172666	
2873	003574	155555	.WORD	155555	
2874	003576	172666	.WORD	172666	
2875	003600	155555	.WORD	155555	
2876	003602	172666	.WORD	172666	
2877	003604	155555	.WORD	155555	
2878	003606	172666	.WORD	172666	
2879	003610	155555	.WORD	155555	
2880	003612	172666	.WORD	172666	
2881	003614	155555	.WORD	155555	
2882	003616	172666	.WORD	172666	
2883	003620	155555	.WORD	155555	
2884	003622	172666	.WORD	172666	
2885	003624	155555	.WORD	155555	
2886					
2887	003626	077777	.WORD	077777	; STANDARD PATTERN 12
2888	003630	137777	.WORD	137777	
2889	003632	157777	.WORD	157777	
2890	003634	167777	.WORD	167777	
2891	003636	173777	.WORD	173777	
2892	003640	175777	.WORD	175777	
2893	003642	176777	.WORD	176777	
2894	003644	177377	.WORD	177377	
2895	003646	177577	.WORD	177577	
2896	003650	177677	.WORD	177677	
2897	003652	177737	.WORD	177737	
2898	003654	177757	.WORD	177757	
2899	003656	177767	.WORD	177767	
2900	003660	177773	.WORD	177773	
2901	003662	177775	.WORD	177775	
2902	003664	177776	.WORD	177776	
2903					
2904	003666	153333	.WORD	153333	; STANDARD PATTERN 13
2905	003670	066667	.WORD	066667	
2906	003672	153333	.WORD	153333	
2907	003674	066667	.WORD	066667	
2908	003676	153333	.WORD	153333	
2909	003700	066667	.WORD	066667	
2910	003702	153333	.WORD	153333	
2911	003704	066667	.WORD	066667	
2912	003706	153333	.WORD	153333	
2913	003710	066667	.WORD	066667	
2914	003712	153333	.WORD	153333	
2915	003714	066667	.WORD	066667	
2916	003716	153333	.WORD	153333	
2917	003720	066667	.WORD	066667	
2918	003722	153333	.WORD	153333	
2919	003724	066667	.WORD	066667	
2920					
2921	003726	000000	.WORD	000000	; STANDARD PATTERN 14
2922	003730	177777	.WORD	177777	
2923	003732	177777	.WORD	177777	
2924	003734	177777	.WORD	177777	
2925	003736	177777	.WORD	177777	
2926	003740	177777	.WORD	177777	

2927	003742	177777	.WORD	177777	
2928	003744	177777	.WORD	177777	
2929	003746	177777	.WORD	177777	
2930	003750	177777	.WORD	177777	
2931	003752	177777	.WORD	177777	
2932	003754	177777	.WORD	177777	
2933	003756	177777	.WORD	177777	
2934	003760	177777	.WORD	177777	
2935	003762	177777	.WORD	177777	
2936	003764	177777	.WORD	177777	
2937					
2938	003766	177777	.WORD	177777	; STANDARD PATTERN 15
2939	003770	000000	.WORD	000000	
2940	003772	000000	.WORD	000000	
2941	003774	000000	.WORD	000000	
2942	003776	000000	.WORD	000000	
2943	004000	000000	.WORD	000000	
2944	004002	000000	.WORD	000000	
2945	004004	000000	.WORD	000000	
2946	004006	000000	.WORD	000000	
2947	004010	000000	.WORD	000000	
2948	004012	000000	.WORD	000000	
2949	004014	000000	.WORD	000000	
2950	004016	000000	.WORD	000000	
2951	004020	000000	.WORD	000000	
2952	004022	000000	.WORD	000000	
2953	004024	000000	.WORD	000000	
2954					


```
2955 .SBTTL ERROR POINTER TABLE
2956
2957 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
2958 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
2959 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
2960 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
2961 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
2962
2963 ;* EM ;:POINTS TO THE ERROR MESSAGE
2964 ;* DH ;:POINTS TO THE DATA HEADER
2965 ;* DT ;:POINTS TO THE DATA
2966 ;* DF ;:POINTS TO THE DATA FORMAT
2967
2968
2969 004026 $ERRTB:
2970 ;ERROR 1
2971
2972 ;ERROR 1
2973
2974 004026 045166 EM1 ;RH11 INTERRUPT OCCURRED (RPAS = 0)
2975 004030 047626 DH1
2976 004032 050260 DT1
2977 004034 000000 0
2978
2979 ;ERROR 2
2980
2981 004036 045231 EM2 ;UNEXPECTED ATTENTION OCCURRED
2982 004040 047633 DH2
2983 004042 050264 DT2
2984 004044 000000 0
2985
2986 ;ERROR 3
2987
2988 004046 045267 EM3 ;MASSBUS PARITY ERROR (MCPE=1)
2989 004050 047710 DH3
2990 004052 050302 DT3
2991 004054 000000 0
2992
2993 ;ERROR 4
2994
2995 004056 045325 EM4 ;MASSBUS PARITY ERROR (PAR=1)
2996 004060 047736 DH4
2997 004062 050312 DT4
2998 004064 000000 0
2999
3000 ;ERROR 5
3001
3002 004066 045362 EM5 ;ADDRESS PLUG BIT CHANGED
3003 004070 047633 DH2
3004 004072 050264 DT2
3005 004074 000000 0
3006
3007 ;ERROR 6
3008
3009 004076 045416 EM6 ;RH11 DIDN'T RESPOND TO ADDRESSING
3010 004100 047775 DH6
```



```
3011 004102 050324 DT6
3012 004104 000000 0
3013
3014 ::*****
3015
3016 .SBTTL SETUP AND INITIALIZATION ROUTINE
3017
3018 : START ADDRESS = 200
3019 : ADDRESS TO CHANGE RH11 UNIBUS ADDRESS = 204
3020
3021 ::*****
3022
3023 004106 012737 000001 034264 START3: MOV #1, TSTPGM ;ENABLE PROGRAMMABLE DRIVES
3024 004114 005037 001260 CLR CHGADR ;CLEAR THE RH11 ADDRESS CHANGE FLAG
3025 004120 000413 BR START2
3026 004122 012737 177777 001260 START: MOV #-1,CHGADR ;SET RH11 ADDRESS CHANGE FLAG
3027 004130 012737 000001 034264 MOV #1, TSTPGM ;ENABLE PROGRAMMABLE DRIVES
3028 004136 000404 BR START2 ;START THE PROGRAM
3029 004140 005037 001260 START1: CLR CHGADR ;CLEAR THE RH11 ADDRESS CHANGE FLAG
3030 004144 005037 034264 CLR TSTPGM ;DISABLE PROGRAMMABLE DRIVES
3031 004150 000005 START2: RESET ;CLEAR THE BUS
3032 004152 013737 001352 001372 MOV ENDCN,ENDCON ;SET UP FOR NORMAL PASS
3033 004160 013737 001354 001374 MOV ENDCN+2,ENDCON+2
3034 004166 013737 001356 001376 MOV ENDSK,ENDSEK
3035 004174 013737 001360 001400 MOV ENDSK+2,ENDSEK+2
3036 004202
3037 SETUP:
3038 .SBTTL INITIALIZE THE COMMON TAGS
3039 004202 012706 001100 ::CLEAR THE COMMON TAGS (%CMTAG) AREA
3040 004206 005026 MOV #%CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
3041 004210 022706 001140 CLR (R6)+ ;;CLEAR MEMORY LOCATION
3042 004214 001374 CMP #SWR,R6 ;;DONE?
3043 004216 012706 001100 BNE -6 ;;LOOP BACK IF NO
3044 ::INITIALIZE A FEW VECTORS
3045 004222 012737 030626 000030 MOV #%ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
3046 004230 012737 000340 000032 MOV #340,@#EMTVEC+2 ;;LEVEL 7
3047 004236 012737 033174 000034 MOV #%TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
3048 004244 012737 000340 000036 MOV #340,@#TRAPVEC+2;LEVEL 7
3049 004252 012737 176543 032560 MOV #176543,$HINUM ;;PRIME THE RANDOM NUMBER GENERATOR
3050 004260 012737 123456 032562 MOV #123456,$LONUM ;;BOTH HIGH AND LOW WORDS
3051 ::SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
3052 ::EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
3053 004266 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
3054 004272 012737 004326 000004 MOV #64,@#ERRVEC ;;SET UP ERROR VECTOR
3055 004300 012737 177570 001140 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
3056 004306 012737 177570 001142 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
3057 004314 022777 177777 174616 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
3058 004322 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
3059 ;;AND THE HARDWARE SWR IS NOT = -1
3060 004324 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
3061 004326 012716 004334 64$: MOV #65$(SP) ;;SET UP FOR TRAP RETURN
3062 004332 000002 RTI
3063 004334 012737 000176 001140 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
3064 004342 012737 000174 001142 MOV #DISPREG,DISPLAY
3065 004350 012637 000004 66$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
3066
```



```

3067 004354 012737 000240 000032      MOV      #240,@#EMTVEC+2      ;CHANGE EMT PRIORITY TO 5
3068 004362 012737 000240 000036      MOV      #240,@#TRAPVEC+2    ;CHANGE TRAP PRIORITY TO 5
3069 004370 005227 177777                INC      #-1                  ;FIRST START ?
3070 004374 001030                BNE      1$                  ;BR IF NOT
3071 004376 023737 000042 000046      CMP      @#42,@#46          ;ACT11 AUTOMATIC MODE?
3072 004404 001004                BNE      4$                  ;BRANCH IF NO
3073 004406 012737 000001 034264      MOV      #1, TSTPGM        ;ENABLE PROGRAMMABLE DRIVES
3074 004414 000402                BR       5$
3075 004416 104401 055652                4$:    TYPE      ,TITLE      ;TYPE THE PROGRAM'S TITLE
3076 004422 005737 034264                5$:    TST      TSTPGM      ;CAN WE USE PROGRAMMABLE DRIVES?
3077 004426 001402                BEQ      6$                  ;BRANCH IF NO
3078 004430 104401 052326                TYPE      ,USE              ;TYPE MSG
3079 004434                6$:
3080 004434 005737 000042                TST      42                  ;AUTO ACCEPT OR CHAIN MODE ?
3081 004440 001006                BNE      1$                  ;BR IF EITHER
3082 004442 122737 000011 000041      CMPB     #11,41             ;LOADED FROM AN RP04/5/6 ?
3083 004450 001002                BNE      1$                  ;BR IF NOT
3084 004452 104401 055703                TYPE      ,LOADRV          ;INSTRUCT THE OPERATOR ON HOW TO TEST DRIVE 0
3085 004456 004737 030146                1$:    JSR      PC,$TKINT     ;TURN ON THE KEYBOARD INTERRUPT
3086                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
3087 004462 005737 000042                TST      @#42               ;ARE WE RUNNING UNDER XXDP/ACT?
3088 004466 001006                BNE      67$                ;BRANCH IF YES
3089 004470 023727 001140 000176      CMP      SWR,#SWREG        ;SOFTWARE SWITCH REG SELECTED?
3090 004476 001005                BNE      68$                ;BRANCH IF NO
3091 004500 104406                GTSWR
3092 004502 000403                BR       68$                ;GET SOFT-SWR SETTINGS
3093 004504 112737 000001 001134      67$:   MOVB     #1,$AUTOB        ;SET AUTO-MODE INDICATOR
3094 004512                68$:
3095 004512 005227 177777                INC      #-1                  ;FIRST START ?
3096 004516 001015                BNE      2$                  ;BR IF NOT
3097 004520 004737 055242                JSR      PC,BUSADR         ;CHECK RH11 BUS ADDRESS
3098 004524 013737 001170 033430      MOV      $RPADR,RPADR      ;RH11 ADDRESS
3099 004532 013737 001172 033432      MOV      $RPVEC,RPVEC     ;RH11 VECTOR ADDRESS
3100 004540 005737 000042                TST      @#42               ;ACT-11 AUTO OR CHAIN MODE?
3101 004544 001002                BNE      2$                  ;BRANCH IF EITHER, SKIP
3102                ;DATE & OPERATOR ID INPUT
3103 004546 004737 054756                JSR      PC,OPRDAT         ;GET THE DATE AND OPERATOR ID
3104 004552 005037 001214                2$:   CLR      STATIN          ;CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
3105 004556 012705 001440                MOV      #ORDERQ,R5        ;START OF AREA TO CLEAR
3106 004562 005025                3$:   CLR      (R5)+
3107 004564 022705 001740                CMP      #BLKADR,R5        ;LOOK FOR END OF CLEAR AREA
3108 004570 001374                BNE      3$                  ;BR IF NOT FINISHED
3109 004572 012706 001100                MOV      #STACK,SP        ;SETUP THE STACK POINTER
3110 004576 005037 177776                CLR      PS                 ;CLEAR THE PROCESSOR STATUS WORD
3111 004602 013737 001212 001274      MOV      HZ,SIXTEE         ;1/60 TH OR 1/50 TH SECOND COUNTER VALUE
3112 004610 005037 001266                CLR      HOUR              ;CLEAR THE HOUR'S COUNTER
3113 004614 005037 001270                CLR      MINUTE            ;CLEAR THE MINUTE'S COUNTER
3114 004620 005037 001272                CLR      SECOND            ;CLEAR THE SECOND'S COUNTER
3115 004624 005037 001412                CLR      INTRVL+2          ;CLEAR INTERVAL COUNTER
3116 004630 005037 001216                CLR      PACK              ;CLEAR THE 'R' OR 'W' COMMAND FLAG
3117 004634 005037 001262                CLR      CFLAG            ;CLEAR THE 'CONTROL C' FLAG
3118 004640 042737 170000 001406      BIC      #170000,MAXER     ;MAKE SURE ERROR LIMITS ARE NOT TOO HIGH
3119
3120                ;ROUTINE TO DETERMINE BUFFER AREA SIZE
3121
3122 004646 005227 177777                SIZMEM: INC      #-1        ;SEE IF TIME TO SIZE MEMORY
    
```



```
3123 004652 001005          BNE      1$          ;BR IF NOT
3124 004654 004737 055136    JSR      PC,$SIZE   ;SEE HOW MUCH MEMORY ON SYSTEM
3125 004660 013737 055232 001256    MOV      $LSTAD,LSTAD ;SAVE THE LAST ADDRESS
3126 004666 012737 000001 001616 1$:      MOV      #1,BUFTBL   ;LOAD NUMBER OF BUFFERS
3127 004674 012737 054756 001620    MOV      #ENDPGM,BUFTBL+2 ;STARTING ADDRESS OF BUFFER
3128 004702 013737 001256 001622    MOV      LSTAD,BUFTBL+4 ;LAST ADDR TO BUFFER ALLOCATION TABLE
3129 004710 162737 054756 001622    SUB      #ENDPGM,BUFTBL+4 ;SUBTRACT PROGRAM SPACE
3130 004716 000241          CLC              ;CLEAR THE 'C' BIT
3131 004720 006037 001622    ROR      BUFTBL+4   ;CONVERT TO WORD COUNT
3132 004724 162737 000144 001622    SUB      #100.,BUFTBL+4 ;SAVE ROOM FOR THE 'ABS' LOADER
3133 004732 023727 001256 100000    CMP      LSTAD,#100000 ;16K ON THE SYSTEM ?
3134 004740 103406          BLO      3$          ;BR IF YES
3135 004742 105737 000041    TSTB    41         ;SEE WHO LOADED THE PROGRAM
3136 004746 001403          BEQ      3$          ;BR IF LOADED BY PAPER TAPE
3137 004750 162737 002570 001622    SUB      #1400.,BUFTBL+4 ;SUBTRACT 'XXDP' LOADER SIZE
3138 004756 005737 001404 3$:      TST      MAXDL     ;VALUE IN 'MAXDL' ?
3139 004762 001012          BNE      4$          ;BR IF VALUE IS
3140 004764 012737 013534 001404    MOV      #5980.,MAXDL ;ASSUME FULL TRACK + 1 SEC MAXIMUM
3141 004772 023737 001404 001622    CMP      MAXDL,BUFTBL+4 ;IS THAT TOO LARGE ?
3142 005000 103403          BLO      4$          ;BR IF NOT
3143 005002 013737 001622 001404    MOV      BUFTBL+4,MAXDL ;USE MAX AVAIL MEMORY AS MAX BUFFER SIZE
3144 005010 013737 001622 053746 4$:      MOV      BUFTBL+4,PARLST+2 ;VALUE FOR THE PARAMETER TABLE
3145
3146          ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
3147
3148 005016 005737 000042    LKPAR:  TST      @#42   ;'XXDP' CHAIN MODE OR 'ACT11' OPERATION ?
3149 005022 001022          BNE      SETVEC     ;BR IF YES
3150 005024 104401 054042    TYPE    ,ASKPAR    ;ASK FOR PARMETERS
3151 005030 104411          RDLIN           ;READ THE ENTRY
3152 005032 012605          MOV      (SP)+,R5   ;ADDRESS OF ENTRY TO R5
3153 005034 122715 000131    CMPB    #'Y',(R5)  ;WAS ENTRY A 'Y' (YES)
3154 005040 001013          BNE      SETVEC     ;BR IF NOT 'Y'
3155
3156 005042 012703 053744    ENTPR:  MOV      #PARLST,R3 ;PARAMETER TABLE ADDRESS
3157 005046 004737 026370    JSR      PC,PARENT  ;GET THE PARAMETER ENTRY
3158 005052 023727 001404 000004    CMP      MAXDL,#4   ;IS THE 'MAXDL' VALUE OK ?
3159 005060 103003          BHS      SETVEC     ;BR IF IT IS
3160 005062 012737 000004 001404    MOV      #4,MAXDL   ;SET 'MAXDL' TO THE MINIMUM VALUE
3161
3162          ;DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
3163          ; THE PROGRAM WILL USE
3164
3165 005070 004737 022500    SETVEC: JSR      PC,CKCLK ;START THE CLOCK
3166 005074 004737 033446    JSR      PC,RPINIT  ;INITIALIZE THE RP04/5/6 DRIVER
3167 005100 012737 177777 033370    MOV      #-1,SAVEFG ;SET THE SAVE REGISTERS FLAG
3168 005106 062727 177777 000000    ADD      #-1,#0     ;CHECK FOR FIRST START
3169 005114 103004          BCC      11$        ;BR IF FIRST START
3170 005116 032777 000004 174014    BIT      #SW02,@SWR ;TYPEOUT THE DRIVE STATUS TABLE ?
3171 005124 001105          BNE      10$        ;BR IF NOT
3172 005126 012737 000340 177776 11$:    MOV      #PR7,PS   ;SET PRIORITY TO 7
3173 005134 005004          CLR      R4         ;DRIVE TABLE POINTER
3174 005136 104401 001165    TYPE    ,%CRLF     ;CR-LF
3175 005142 104401 052634    TYPE    ,SYSTAT    ;TYPE STATUS HEADING
3176 005146          1$:
3177 005146 010446          MOV      R4,-(SP)  ;;SAVE R4 FOR TYPEOUT
3178          ;;TYPE DRIVE NUMBER
```



```
3179 005150 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
3180 005152      002        .BYTE 2          ;;TYPE 2 DIGIT(S)
3181 005153      000        .BYTE 0          ;;SUPPRESS LEADING ZEROS
3182 005154 104401 052321    TYPE ,LIN4SP     ;;SPACES
3183 005160 105764 033302    TSTB DRVSTA(R4) ;;CHECK DRIVE'S STATUS
3184 005164 100425          BMI 4$          ;;BR IF UNSAFE
3185 005166 001027          BNE 5$          ;;BR IF ONLINE
3186 005170 105764 033312    TSTB DRVTP(R4) ;;SEE IF OFFLINE OR NONEXISTENT
3187 005174 001404          BEQ 2$          ;;BR IF NONEXISTENT
3188 005176 100006          BPL 3$          ;;BR IF OFFLINE
3189 005200 104401 052547    TYPE ,NOTRP     ;;DRIVE NOT AN RP04/5/6
3190 005204 000447          BR 9$          ;;CHECK NEXT DRIVE
3191 005206 104401 052570    2$: TYPE ,NOTPRS ;;DRIVE NOT PRESENT
3192 005212 000444          BR 9$          ;;CHECK NEXT DRIVE
3193 005214 132764 000010 033312 3$: BITB #BIT03, DRVTP(R4) ;;DRIVE PROGRAMMABLE?
3194 005222 001403          BEQ 12$        ;;BRANCH IF NO
3195 005224 104401 052403    TYPE ,NOUSE     ;;PRINT MSG
3196 005230 000410          BR 6$          ;;PRINT DRIVE TYPE
3197 005232 104401 052456    12$: TYPE ,UNTOFF ;;DRIVE OFFLINE
3198 005236 000405          BR 6$          ;;PRINT DRIVE TYPE
3199 005240 104401 052624    4$: TYPE ,NOTSAF ;;DRIVE UNSAFE
3200 005244 000402          BR 6$          ;;PRINT DRIVE TYPE
3201 005246 104401 052467    5$: TYPE ,UNTON  ;;DRIVE ONLINE
3202 005252 104401 052323    6$: TYPE ,LINSR  ;;SPACES
3203 005256 012737 052654 005322 MOV #RP04B,8$   ;;ADDRESS OF RP04 MESSAGE
3204 005264 132764 000001 033312 BITB #BIT00,DRVTP(R4) ;;RP04 ?
3205 005272 001012          BNE 7$          ;;BR IF YES
3206 005274 012737 052661 005322 MOV #RP05,8$   ;;ADDRESS OF RP05 MESSAGE
3207 005302 132764 000002 033312 BITB #BIT01,DRVTP(R4) ;;RP05 ?
3208 005310 001003          BNE 7$          ;;BR IF YES
3209 005312 012737 052666 005322 MOV #RP06,8$   ;;ADDRESS OF RP06 MESSAGE
3210 005320 104401          7$: TYPE        ;;TYPE THE DRIVE TYPE MESSAGE
3211 005322 000000          8$: .WORD 0    ;;MESSAGE ADDRESS HERE
3212 005324 104401 001165    9$: TYPE ,%CRLF ;;CR-LF
3213 005330 005204          INC R4         ;;INCREMENT DRIVE NUMBER/TABLE POINTER
3214 005332 020427 000010    CMP R4,#8.    ;;FINISHED ?
3215 005336 001303          BNE 1$        ;;BR IF NOT
3216 005340 104401 001165    10$: TYPE ,%CRLF ;;CR-LF
3217 005344 005037 177776    CLR PS        ;;SET PRIORITY BACK TO '0'
3218 005350 000137 005354    JMP MONTR     ;;CHECK FOR 'XXDP' OR 'ACT11' MONITOR
3219
3220          ;SETUP IF 'XXDP' OR 'ACT11' OPERATION
3221
3222 005354 005737 000042    MONTR: TST 42   ;;'XXDP' CHAIN MODE OR 'ACT11' AUTO ACCEPT
3223 005360 001402          BEQ 1$        ;;BR IF NEITHER
3224 005362 004737 024362    JSR PC,ASGN2 ;;ASSIGN DRIVES
3225 005366 005227 177777    1$: INC #-1   ;;FIRST START ?
3226 005372 001011          BNE 2$        ;;BR IF NOT
3227 005374 105737 000041    TSTB @#41    ;;LOADED FROM PAPER TAPE ?
3228 005400 001406          BEQ 2$        ;;BR IF YES
3229 005402 023727 001256 100000 CMP LSTAD,#100000 ;;MORE THAN 16K ON THE SYSTEM ?
3230 005410 103002          BHIS 2$      ;;BR IF YES
3231 005412 104401 056102    TYPE ,NLOAD   ;;TELL THE OPERATOR THAT THE 'XXDP' LOADER
3232          ;;WILL BE OVERWRITTEN
3233 005416 004737 030146    2$: JSR PC,$TKINT ;;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
3234 005422 104401 053631    TYPE ,INTDON ;;TYPE 'INITIALIZE COMPLETE'
```



```
3235 005426 000137 005610          JMP      MAIN1          ;START THE PROGRAM
3236
3237          ;'XXDP' OR 'ACT11' END OF TEST ROUTINE
3238
3239 005432 013700 000042          $GET42: MOV      42,R0          ;MONITOR ADDRESS
3240 005436 001002          BNE      1$              ;BR IF MONITOR
3241 005440 000137 005610          JMP      MAIN1          ;NONE, CONTINUE
3242 005444 000005          1$:      RESET          ;CLEAR EVERYTHING
3243 005446 004710          $ENDAD: JSR     PC,(R0)      ;GO TO THE MONITOR
3244 005450 000240          NOP                      ;SAVE RCOM
3245 005452 000240          NOP                      ;FOR
3246 005454 000240          NOP                      ;ACT11
3247 005456 000137 004140          $DOAGN: JMP     START1      ;START AGAIN
3248
3249          ;;*****
3250
3251          .SBTTL  MAIN PROGRAM
3252
3253          ;;*****
3254
3255 005462 012703 000010          MAIN:   MOV      #8.,R3          ;DRIVE COUNTER
3256 005466 012705 001464          MOV      #DUNIT,R5          ;ADDRESS OF 'DROP DRIVE' TABLE
3257 005472 005715          1$:     TST      (R5)          ;SEE IF ENTRY AT PRESENT POSITION
3258 005474 001011          BNE      3$              ;BR IF THERE IS ONE
3259 005476 062705 000002          2$:     ADD      #2,R5          ;INCREMENT TO NEXT TABLE POSITION
3260 005502 005303          DEC      R3              ;DECREMENT DRIVE COUNTER
3261 005504 001372          BNE      1$              ;BR IF MORE TO CHECK
3262 005506 005737 001462          TST      ASNLST          ;ANY DRIVES ACTIVE ?
3263 005512 001036          BNE      MAIN1          ;BR IF YES
3264 005514 000137 005432          JMP      $GET42          ;CHECK FOR MONITOR RETURN
3265 005520 012701 001530          3$:     MOV      #AVAIL,R1      ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
3266 005524 005711          4$:     TST      (R1)          ;SEE IF AT END OF TABLE
3267 005526 001405          BEQ      5$              ;BR IF AT END: GO CHECK 'WAIT' TABLE
3268 005530 021115          CMP      (R1),(R5)        ;IS DRIVE IN 'AVAIL' THE ONE TO BE DROPPED
3269 005532 001414          BEQ      7$              ;BR IF YES
3270 005534 062701 000002          ADD      #2,R1          ;INCREMENT 'AVAIL' TABLE ADDRESS
3271 005540 000771          BR       4$              ;CONTINUE LOOKING
3272 005542 012701 001552          5$:     MOV      #WAIT,R1      ;MOVE THE ADDRESS OF THE BUFFER WAIT TABLE
3273 005546 005711          6$:     TST      (R1)          ;AT THE END OF THE 'WAIT' TABLE ?
3274 005550 001752          BEQ      2$              ;BR IF YES: SEE IF ANY MORE 'DROP' REQUESTS
3275 005552 021115          CMP      (R1),(R5)        ;DRIVE IN THE 'WAIT' TABLE ?
3276 005554 001403          BEQ      7$              ;BR IF IT IS
3277 005556 062701 000002          ADD      #2,R1          ;INCREMENT 'WAIT' TABLE ADDRESS
3278 005562 000771          BR       6$              ;CONTINUE LOOK THROUGH THE 'WAIT' TABLE
3279 005564 011100          7$:     MOV      (R1),R0        ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
3280 005566 104401 053140          TYPE     ,DEASSG          ;TYPE 'DRIVE DEASSIGNED'
3281 005572 004737 022764          JSR     PC,TYPEST        ;TYPE THE DRIVE'S PERFORMANCE SUMMARY
3282 005576 005015          CLR*    (R5)            ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
3283 005600 005011          CLR     (R1)            ;REMOVE THE DRIVE FROM THE 'AVAIL' OR 'WAIT' TABLE
3284 005602 004737 017616          JSR     PC,CMPRES        ;COMPRESS THE RESPECTIVE TABLE
3285 005606 000733          BR       2$              ;SEE IF ANY MORE DRIVES
3286
3287          ;LOOK FOR DRIVES TO BE ASSIGNED
3288
3289 005610 012703 000010          MAIN1:  MOV      #8.,R3          ;DRIVE COUNT
3290 005614 005002          CLR     R2              ;'AVAIL' INDEX
```



```
3291 005616 005004 CLR R4 ;ASSIGN LIST INDEX
3292 005620 005005 CLR R5 ;NEW DRIVE INDEX
3293 005622 005765 001506 1$: TST NEWUNT(R5) ;NEW DRIVE IN THIS POSITION
3294 005626 001006 BNE 3$ ;BR IF THERE IS
3295 005630 062705 000002 2$: ADD #2,R5 ;INCREMENT R5
3296 005634 005204 INC R4 ;INCREMENT ASSIGN INDEX
3297 005636 005303 DEC R3 ;DECREMENT DRIVE COUNT
3298 005640 001370 BNE 1$ ;BR IF MORE DRIVES
3299 005642 000432 BR MAIN2 ;START OPERATIONS FOR THE AVAILABLE DRIVES
3300 005644 104401 052450 3$: TYPE ,UNMSG ;'DRIVE'
3301 005650 010446 MOV R4,-(SP) ;;SAVE R4 FOR TYPEOUT
3302 ;;TYPE DRIVE NUMBER
3303 005652 104403 TYPOS ;;GO TYPE--OCTAL ASCII
3304 005654 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
3305 005655 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
3306 005656 104401 053210 TYPE ,ASGND ;'ASSIGNED'
3307 005662 005762 001530 4$: TST AVAIL(R2) ;AT END OF AVAILABLE TABLE
3308 005666 001403 BEQ 5$ ;BR IF YES
3309 005670 062702 000002 ADD #2,R2 ;INCREMENT AVAILABLE TABLE INDEX
3310 005674 000772 BR 4$ ;CONTINUE LOOKING FOR END OF TABLE
3311 005676 016562 001506 001530 5$: MOV NEWUNT(R5),AVAIL(R2) ;MOVE ADDR OF DRIVE INTO AVAIL LST
3312 005704 005065 001506 CLR NEWUNT(R5) ;TAKE DRIVE OUT OF NEW DRIVE TABLE
3313 005710 156437 033416 001462 BISB ATABIT(R4),ASNLS ;SET DRIVE ASSIGNED INDICATOR
3314 005716 016200 001530 MOV AVAIL(R2),R0 ;PUT STARTING ADDRESS OF BLOCK IN R0
3315 005722 062702 000002 ADD #2,R2 ;INCREMENT AVAILABLE TABLE POINTER
3316 005726 000740 BR 2$ ;LOOK FOR MORE DRIVES
3317
3318 ;GET PARAMETERS, BUFFER SPACE, AND START ORDERS FOR DRIVES IN
3319 ; THE 'AVAILABLE' QUEUE
3320
3321 005730 005737 001552 MAIN2: TST WAIT ;OUTSTANDING BUFFER REQUESTS
3322 005734 001113 BNE MAIN3 ;BR IF THERE ARE
3323 005736 005002 CLR R2 ;CLEAR DRIVE TABLE POINTER
3324 005740 005762 001530 1$: TST AVAIL(R2) ;ANY DRIVES WAITING FOR PARAMETERS
3325 005744 001551 BEQ IDLE ;BRANCH IF NONE
3326 005746 016200 001530 MOV AVAIL(R2),R0 ;CONTROL BLOCK ADDR IN R0
3327 005752 005760 000104 TST $NEXT(R0) ;PARAMETERS BEEN SELECTED ?
3328 005756 001021 BNE 6$ ;BR IF THEY HAVE
3329 005760 105760 000026 TSTB $PACK(R0) ;'R' OR 'W' COMMAND FOR THE DRIVE ?
3330 005764 001403 BEQ 2$ ;BR IF NOT
3331 005766 004737 017634 JSR PC,WRTPK ;GET DATA PACK PARAMETERS
3332 005772 000415 BR 7$ ;GET THE BUFFER
3333 005774 012701 001574 2$: MOV #PARQ,R1 ;ADDRESS OF THE PARAMETER QUEUE
3334 006000 020011 3$: CMP R0,(R1) ;IS CURRENT DRIVE IN THE QUEUE ?
3335 006002 001403 BEQ 4$ ;BR IF IT IS
3336 006004 005721 TST (R1)+ ;AT END OF THE QUEUE
3337 006006 001403 BEQ 5$ ;BR IF AT END
3338 006010 000773 BR 3$ ;CONTINUE LOOKING
3339 006012 004737 017616 4$: JSR PC,COMPRES ;COMPRESS THE TABLE
3340 006016 004737 016534 5$: JSR PC,SELPAR ;SELECT THE PARAMETERS
3341 006022 004737 017410 6$: JSR PC,GETPAR ;LOAD NEW PARAMETERS
3342 006026 005046 7$: CLR -(SP) ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
3343 006030 004737 015760 JSR PC,GETBUF ;GET BUFFER
3344 006034 012660 000006 MOV (SP)+,$BUF(R0) ;MOVE BUFFER ADDR TO DPB
3345 006040 001424 BEQ 8$ ;BR IF '0' ADDR (NO BUFFER)
3346 006042 004737 016330 JSR PC,FILBUF ;FILL THE BUFFER
```



```
3347 006046 005060 000072 CLR $FAIR(R0) ;CLEAR THE 'FAIRNESS' COUNT
3348 006052 004737 016456 JSR PC,GODRIV ;PUT CURRENT DPB IN DRIVER
3349 006056 012705 001440 MOV #ORDERQ,R5 ;ADDRESS OF ORDER QUEUE IN R5
3350 006062 005725 TST (R5)+ ;END OF QUEUE ?
3351 006064 001376 BNE -2 ;BR IF NOT
3352 006066 010045 MOV RO,-(R5) ;PUT BLOCK ADDRESS INTO QUEUE
3353 006070 105760 000026 TSTB $PACK(R0) ;'R' OR 'W' COMMAND FOR DRIVE ?
3354 006074 001025 BNE 10$ ;BR IF EITHER
3355 006076 012705 001574 MOV #PARQ,R5 ;PUT BLOCK INTO THE PARAMETER QUEUE
3356 006102 005725 TST (R5)+ ;FIND THE END OF THE QUEUE
3357 006104 001376 BNE -2 ;BR IF NOT AT END OF QUEUE
3358 006106 010045 MOV RO,-(R5) ;PUT BLOCK ADDRESS INTO THE QUEUE
3359 006110 000417 BR 10$ ;CONTINUE LOOKING
3360 006112 026037 000072 001254 8$: CMP $FAIR(R0),FAIRNS ;ENTRY BEEN IN THE QUEUE LONG ENOUGH ?
3361 006120 001405 BEQ 9$ ;BR IF YES
3362 006122 005260 000072 INC $FAIR(R0) ;INCREMENT THE ENTRY COUNT
3363 006126 062702 000002 ADD #2,R2 ;INCREMENT THE POINTER
3364 006132 000702 BR 1$ ;LOOK FOR SOME MORE DRIVES
3365 006134 012705 001552 9$: MOV #WAIT,R5 ;'WAIT' QUEUE ADDRESS
3366 006140 005725 TST (R5)+ ;LOOK FOR AN OPENING
3367 006142 001376 BNE -2 ;BR IF NONE YET
3368 006144 016245 001530 MOV AVAIL(R2),-(R5) ;MOVE DRIVE'S BLOCK ADDRESS TO QUEUE
3369 006150 012701 001530 10$: MOV #AVAIL,R1 ;'AVAILABLE' TABLE ADDRESS
3370 006154 060201 ADD R2,R1 ;FORM ADDRESS OF LAST ENTRY
3371 006156 004737 017616 JSR PC,COMPRES ;COMPRESS THE TABLE
3372 006162 000666 BR 1$ ;CONTINUE LOOKING
3373
3374 ;GET BUFFER ASSIGNMENTS FOR DRIVES IN THE 'BUFFER WAIT' QUEUE
3375
3376 006164 013700 001552 MAIN3: MOV WAIT,R0 ;MOVE THE 'WAIT' ENTRY TO R0
3377 006170 005046 CLR -(SP) ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
3378 006172 004737 015760 JSR PC,GETBUF ;TRY TO GET A BUFFER
3379 006176 012660 000006 MOV (SP)+,$BUF(R0) ;MOVE THE BUFFER ADDR TO THE DPB
3380 006202 001002 BNE 1$ ;BR IF A BUFFER WAS ASSIGNED
3381 006204 000137 006270 JMP IDLE ;NO BUFFER AVAILABLE YET
3382 006210 004737 016330 1$: JSR PC,FILBUF ;FILL THE BUFFER
3383 006214 004737 016456 JSR PC,GODRIV ;PUT THE ENTRY IN THE DRIVER
3384 006220 005060 000072 CLR $FAIR(R0) ;CLEAR THE 'FAIRNESS' COUNT
3385 006224 012705 001440 MOV #ORDERQ,R5 ;ADDRESS OF ORDER QUEUE IN R5
3386 006230 005725 TST (R5)+ ;AT END OF THE QUEUE
3387 006232 001376 BNE -2 ;BR IF NOT
3388 006234 010045 MOV RO,-(R5) ;PUT BLOCK ADDRESS IN QUEUE
3389 006236 105760 000026 TSTB $PACK(R0) ;'R' OR 'W' COMMAND FOR DRIVE ?
3390 006242 001005 BNE 2$ ;BR IF YES, DON'T PUT BLOCK INTO 'PARQ'
3391 006244 012705 001574 MOV #PARQ,R5 ;FIND THE END OF THE PARAMETER QUEUE
3392 006250 005725 TST (R5)+ ;OPEN SLOT IN THE QUEUE ?
3393 006252 001376 BNE -2 ;BR IF NOT
3394 006254 010045 MOV RO,-(R5) ;PUT BLOCK ADDRESS INTO THE QUEUE
3395 006256 012701 001552 2$: MOV #WAIT,R1 ;ADDRESS OF TABLE TO TO COMPRESS
3396 006262 004737 017616 JSR PC,COMPRES ;COMPRESS THE WAIT TABLE
3397 006266 000620 BR MAIN2 ;LOOK FOR MORE ENTRIES
3398
3399 ;WAIT FOR AN ORDER TO FINISH
3400
3401 006270 012701 001440 IDLE: MOV #ORDERQ,R1 ;ADDRESS OF THE ORDER QUEUE IN R1
3402 006274 012100 1$: MOV (R1)+,R0 ;PUT BLOCK ADDRESS INTO R0
```



```
3403 006276 001433          BEQ  IDLE1          ;BR IF END OF QUEUE
3404 006300 005760 000016    TST  $STATUS(RO)   ;SEE IF DRIVE FINISHED
3405 006304 001773          BEQ  1$            ;BR IF DRIVE NOT FINISHED
3406 006306 162701 000002    SUB  #2,R1         ;CORRECT THE QUEUE POINTER
3407 006312 010146          MOV  R1,-(SP)      ;SAVE THE QUEUE ADDRESS
3408 006314 004737 015614    JSR  PC,STATIS    ;ACCUMULATE STATISTICS FOR DRIVE IN RO
3409 006320 000240          NOP               ;DEBUGGING AID
3410 006322 004737 006626    JSR  PC,PROCES    ;PROCESS END OF ORDER
3411 006326 005037 001264    CLR  BADSEC       ;CLEAR THE BAD TRK/SEC ERROR INDICATOR
3412 006332 004737 026622    JSR  PC,ABNRML    ;SEE IF ANY DRIVES HAVE TOO MANY ERRORS
3413 006336 004737 026650    JSR  PC,EOP       ;SEE IF ANY DRIVE HAS XFERED 3X10^9 BITS
3414 006342 012601          MOV  (SP)+,R1     ;RESTORE THE ORDER TABLE INDEX
3415 006344 012705 001530    MOV  #AVAIL,R5    ;FIND THE END OF THE 'AVAILABLE' TABLE
3416 006350 005725          2$: TST  (R5)+       ;END OF THE TABLE ?
3417 006352 001376          BNE  2$           ;BR IF NOT AT END OF LIST
3418 006354 011145          MOV  (R1),-(R5)   ;MOVE THE BLOCK ADDRESS INTO THE TABLE
3419 006356 004737 017616    JSR  PC,COMPRES   ;COMPRESS THE ORDER QUEUE
3420 006362 004737 016114    JSR  PC,RELBUF    ;RESTORE BUFFER
3421 006366 005737 001262    IDLE1: TST  CFLAG   ;'CONTROL C' FLAG ENTERED ?
3422 006372 001403          BEQ  1$           ;BR IF IT WAS
3423 006374 004737 023776    JSR  PC,KSR       ;SERVICE THE KEYBOARD
3424 006400 000733          BR   IDLE         ;SYSTEM WAS BUSY
3425 006402 032777 000004 172530 1$: BIT  #SW02,@SWR   ;TYPE PERFORMANCE SUMMARY
3426 006410 001007          BNE  2$           ;BR IF NOT
3427 006412 005737 001214    TST  STATIN      ;TIME TO TYPE THE PERFORMANCE SUMMARY ?
3428 006416 001404          BEQ  2$           ;BR IF NCT
3429 006420 005037 001214    CLR  STATIN      ;CLEAR THE INDICATOR
3430 006424 004737 022702    JSR  PC,STATPR    ;TYPE THE SUMMARY
3431 006430 005737 001574          2$: TST  PARQ      ;ENTRY IN THE PARAMETER QUEUE ?
3432 006434 001410          BEQ  3$           ;BR IF NOT
3433 006436 013700 001574    MOV  PARQ,RO     ;PUT THE BLOCK ADDRESS INTO RO
3434 006442 004737 016534    JSR  PC,SELPAR    ;GET THE PARAMETERS FOR NEXT OPERATION
3435 006446 012701 001574    MOV  #PARQ,R1    ;SETUP TO COMPRESS THE TABLE
3436 006452 004737 017616    JSR  PC,COMPRES   ;COMPRESS THE PARAMETER QUEUE
3437 006456 000137 005462          3$: JMP  MAIN      ;CONTINUE THE LOOP
3438
3439          ;SETUP TO REFORMAT AN ERROR SECTOR
3440
3441 006462 032777 000001 172450 REFMT: BIT  #SW0,@SWR   ;READ ONLY SWITCH SET ?
3442 006470 001055          BNE  REFMTX      ;BR IF IT IS
3443 006472 032777 000200 172440    BIT  #SW7,@SWR   ;SWITCH 7 SET ?
3444 006500 001051          BNE  REFMTX      ;BR IF IT IS
3445 006502 005737 001416    TST  FORMAT      ;WRITE HEADER & DATA ORDERS ALLOWED ?
3446 006506 001446          BEQ  REFMTX      ;BR IF NOT
3447 006510 016060 000272 000100    MOV  $RPCC(RO),$NCYL(RO) ;USE PRESENT CYLINDER
3448 006516 004737 022402    JSR  PC,READDR    ;GET CORRECTED SECTOR-TRACK ADDRESSES
3449 006522 112660 000077    MOV  (SP)+,$NTRK(RO) ;TRACK ADDR TO DPB
3450 006526 112660 000076    MOV  (SP)+,$NSEC(RO) ;SECTOR ADDR TO DPB
3451 006532 012760 000404 000102    MOV  #260,$NWRDL(RO) ;WORD COUNT FOR FORMAT
3452 006540 023727 001404 000404    CMP  MAXDL,#260. ;CAN A FULL SECTOR BE WRITTEN ?
3453 006546 103003          BHS  1$          ;BR IF IT CAN
3454 006550 013760 001404 000102    MOV  MAXDL,$NWRDL(RO) ;PUT TRANSFER SIZE INTO THE DPB
3455 006556 112760 000003 000074 1$: MOV  #3,$NCODE(RO) ;COMMAND CODE
3456 006564 004737 017362    JSR  PC,GETPAT    ;GET A PATTERN
3457 006570 110560 000075    MOV  R5,$NPATC(RO) ;PATTERN CODE TO CONTROL BLOCK
3458 006574 012760 177777 000104    MOV  #-1,$NEXT(RO) ;SET PARAMETERS SELECTED INDICATOR
```


3459 006602 012701 001574
3460 006606 005711
3461 006610 001405

2\$: MOV #PARQ,R1 ;SET UP TO SEE IF BLOCK IN THE PARAMETER QUEUE
 TST (R1) ;SEE IF AT END OF TABLE
 BEQ REFMTX ;BR IF AT END

3462 006612 020021
3463 006614 001374
3464 006616 005041
3465 006620 004737 017616
3466 006624 000207
3467
3468
3469

CMP R0,(R1)+ ;SEE IF BLOCK AT PRESENT POSITION
BNE 2\$;BR IF NOT
CLR -(R1) ;CLEAR THE ENTRY
JSR PC,CMPRES ;COMPRESS THE TABLE
REFMTX: RTS PC ;RETURN
;PROCESS THE ORDER TERMINATION


```
3470 006626 111037 001246 PROCES: MOVB (RO),UNIT ;DRIVE NUMBER FOR ANY ERROR MESSAGES
3471 006632 005760 000016 TST $STATUS(RO) ;SEE IF DRIVER SIGNALLED AN ERROR
3472 006636 100427 BMI ERPROC ;BR IF ERROR
3473 006640 032760 100000 000234 BIT #BIT15,$RPCS1(RO) ;SEE IF 'SC' SET
3474 006646 001410 BEQ 1$ ;BR IF NOT SET
3475 006650 032760 040000 000234 BIT #BIT14,$RPCS1(RO) ;SEE IF 'TRE' SET
3476 006656 001017 BNE ERPROC ;BR IF SET
3477 006660 032760 040000 000246 BIT #BIT14,$RPDS1(RO) ;SEE IF 'ERR' SET
3478 006666 001013 BNE ERPROC ;BR IF SET
3479 006670 004737 013056 1$: JSR PC,CKERR ;NO ERROR, CHECK ERROR BITS ANYWAY
3480 006674 004737 013156 JSR PC,CKBUS ;NO ERROR, CHECK BUS ADDR & WC
3481 006700 032777 000002 172232 BIT #SW01,@SWR ;DATA COMPARE ?
3482 006706 001002 BNE 2$ ;BR IF NOT
3483 006710 004737 013242 JSR PC,CMPAR ;NO ERROR, COMPARE DATA
3484 006714 000207 2$: RTS PC ;RETURN
3485
3486 ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
3487
3488 006716 032760 000200 000016 ERPROC: BIT #BIT07,$STATUS(RO) ;DONE BIT SET ?
3489 006724 001402 BEQ ERPRC1 ;BR IF ORDER DIDN'T COMPLETE NORMALLY
3490 006726 000137 007352 JMP DONE ;PROCESS ERROR WITH 'DONE' BIT SET
3491
3492 ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
3493
3494 006732 032760 010000 000016 ERPRC1: BIT #BIT12,$STATUS(RO) ;SEE IF DRIVE WAS UNSAFE
3495 006740 001025 BNE PUNSAF ;BR IF YES
3496 006742 032760 004000 000016 BIT #BIT11,$STATUS(RO) ;PARITY ERROR OCCURRED
3497 006750 001055 BNE UCPAR ;BR IF IT DID
3498 006752 032760 002000 000016 BIT #BIT10,$STATUS(RO) ;FATAL PARITY ERROR?
3499 006760 001056 BNE FALPAR ;BR IF THERE IS ONE
3500 006762 032760 001000 000016 BIT #BIT09,$STATUS(RO) ;TIMEOUT?
3501 006770 001076 BNE SWTIM ;BR IF YES
3502 006772 032760 040002 000016 BIT #BIT14:BIT01,$STATUS(RO) ;DRIVE WENT OFFLINE ?
3503 007000 001111 BNE OFLIN ;BR IF IT DID
3504 007002 032760 000004 000016 BIT #BIT2,$STATUS(RO) ;PORT REQUEST TIME OUT ?
3505 007010 001141 BNE PRTIM ;BR IF IT DID
3506 007012 000207 RTS PC ;ERROR. RETURN
3507
3508 ;DRIVE IS PERSISTENTLY UNSAFE
3509
3510 007014 104401 001165 PUNSAF: TYPE , $CRLF ;CR-LF
3511 007020 104401 045556 TYPE ,EM12 ;'DRIVE UNSAFE' MESSAGE
3512 007024 104401 053163 TYPE ,DRNUM ;DRIVE NUMBER
3513 007030 013746 001246 MOV UNIT,-(SP) ;SAVE UNIT FOR TYPEOUT
3514 ;TYPE DRIVE NUMBER
3515 007034 104403 TYPOS ;GO TYPE--OCTAL ASCII
3516 007036 002 .BYTE 2 ;TYPE 2 DIGIT(S)
3517 007037 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
3518 007040 104401 001165 TYPE , $CRLF ;CR-LF
3519 007044 004737 020266 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3520 007050 104414 045556 DISPLY ,EM12 ;PERSISTENT DEVICE UNSAFE MESSAGE
3521 007054 004737 020332 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3522 007060 004737 020740 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3523 007064 004737 021414 JSR PC,LINE4 ;PRINT LINE 4 OF THE ERROR MESSAGE
3524 007070 004737 023512 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3525 007074 004737 022050 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
```



```
3526 007100 000137 026544          JMP      DROP          ;DROP THE DRIVE
3527
3528          ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
3529
3530 007104 104401 001165      UCPAR:  TYPE      ,%CRLF          ;CR-LF
3531 007110 104401 045460          TYPE      ,EM10          ;'UNCORRECTABLE PARITY ERROR' MESSAGE
3532 007114 000404          BR        FALPR1        ;FINISH PROCESSING THE ERROR
3533
3534          ;'FATAL' MASSBUS PARITY ERROR OCCURRED
3535
3536 007116 104401 001165      FALPAR:  TYPE      ,%CRLF          ;CR-LF
3537 007122 104401 045523          TYPE      ,EM11          ;'FATAL PARITY ERROR' MESSAGE
3538 007126 104401 053163      FALPR1:  TYPE      ,DRNUM        ;DRIVE NUMBER
3539 007132 013746 001246          MOV      UNIT,-(SP)    ;:SAVE UNIT FOR TYPEOUT
3540          ;:TYPE DRIVE NUMBER
3541 007136 104403          TYPOS          ;:GO TYPE--OCTAL ASCII
3542 007140          002          .BYTE      2          ;:TYPE 2 DIGIT(S)
3543 007141          000          .BYTE      0          ;:SUPPRESS LEADING ZEROS
3544 007142 104401 001165          TYPE      ,%CRLF          ;CR-LF
3545 007146 004737 023512          JSR      PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
3546 007152 032777 100000 171760      BIT      #SW15,@SWR    ;HALT ON ERROR ?
3547 007160 001401          BEQ      1$          ;BR IF NOT
3548 007162 000000          HALT          ;ERROR HALT
3549 007164 000207      1$:      RTS      PC
3550
3551          ;SOFTWARE TIMEOUT OCCURRED
3552
3553 007166 004737 020266      SWTIM:  JSR      PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3554 007172 104414 045607          DISPLY   ,EM13          ;PRINT THE TIME OUT MESSAGE
3555 007176 004737 020332          JSR      PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3556 007202 004737 020740          JSR      PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3557 007206 004737 021414          JSR      PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3558 007212 004737 023512          JSR      PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
3559 007216 004737 022050          JSR      PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3560 007222 000207          RTS      PC          ;RETURN
3561
3562          ;DRIVE WENT OFFLINE
3563
3564 007224 104401 001165      OFLIN:  TYPE      ,%CRLF          ;CR-LF
3565 007230 104401 045661          TYPE      ,EM14          ;'DRIVE WENT OFFLINE' MESSAGE
3566 007234 104401 053163          TYPE      ,DRNUM        ;DRIVE NUMBER
3567 007240 013746 001246          MOV      UNIT,-(SP)    ;:SAVE UNIT FOR TYPEOUT
3568          ;:TYPE DRIVE NUMBER
3569 007244 104403          TYPOS          ;:GO TYPE--OCTAL ASCII
3570 007246          002          .BYTE      2          ;:TYPE 2 DIGIT(S)
3571 007247          000          .BYTE      0          ;:SUPPRESS LEADING ZEROS
3572 007250 104401 001165          TYPE      ,%CRLF          ;CR-LF
3573 007254 004737 020266          JSR      PC,LINE1        ;PRINT LINE 1 OF THE ERROR MESSAGE
3574 007260 104414 045661          DISPLY   ,EM14          ;PRINT OFFLINE MESSAGE
3575 007264 004737 020332          JSR      PC,LINE2        ;PRINT LINE 2 OF THE ERROR MESSAGE
3576 007270 004737 020740          JSR      PC,LINE3        ;PRINT LINE 3 OF THE ERROR MESSAGE
3577 007274 004737 021414          JSR      PC,LINE4        ;PRINT LINE 4 OF THE ERROR MESSAGE
3578 007300 004737 023512          JSR      PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
3579 007304 004737 022050          JSR      PC,LINE7        ;PRINT LINE 7 OF THE ERROR MESSAGE
3580 007310 000137 026544          JMP      DROP          ;DROP THE DRIVE
3581
```



```
3582 ;PORT REQUEST TIMEOUT ERROR
3583
3584 007314 004737 020266 PRTIM: JSR PC,LINE1 ;TYPE LINE 1 OF THE ERROR MESSAGE
3585 007320 104414 045704 DISPLY ,EM15 ;PRINT PORT TIME OUT MESSAGE
3586 007324 004737 020332 JSR PC,LINE2 ;TYPE LINE 2 OF THE ERROR MESSAGE
3587 007330 004737 020740 JSR PC,LINE3 ;TYPE LINE 3 OF THE ERROR MESSAGE
3588 007334 004737 021414 JSR PC,LINE4 ;TYPE LINE 4 OF THE ERROR MESSAGE
3589 007340 004737 023512 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3590 007344 004737 022050 JSR PC,LINE7 ;TYPE LINE 7 OF THE ERROR MESSAGE
3591 007350 000207 RTS PC ;RETURN
3592
3593 ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
3594
3595 007352 032760 000030 000016 DONE: BIT #BIT04,BIT03,$STATUS(RO) ;UNSAFE OCCURRED
3596 007360 001402 BEQ .+6 ;BR IF NOT
3597 007362 000137 012554 JMP UNSAF ;REPORT UNSAFE
3598 007366 032760 040000 000244 BIT #BIT14,$RPCS2(RO) ;IS 'WCE' SET ?
3599 007374 001006 BNE 1$ ;BR IF SET
3600 007376 032760 040000 000246 BIT #BIT14,$RPDS1(RO) ;CHECK 'ERR'
3601 007404 001002 BNE 1$ ;BR IF SET
3602 007406 000137 012320 JMP TRFER ;PROCESS 'TRE'
3603 007412 032760 000400 000250 1$: BIT #BIT08,$RPER1(RO) ;'HCRC' SET?
3604 007420 001402 BEQ .+6 ;BR IF NOT
3605 007422 000137 010776 JMP HRCER ;PROCESS 'HCRC'
3606 007426 032760 000020 000250 BIT #BIT04,$RPER1(RO) ;'FMT' SET?
3607 007434 001402 BEQ .+6 ;BR IF NOT SET
3608 007436 000137 011160 JMP CKFMT ;CHECK FORMAT ERROR
3609 007442 032760 00C200 000250 BIT #BIT07,$RPER1(RO) ;'HCE' SET?
3610 007450 001402 BEQ .+6 ;BR IF NOT SET
3611 007452 000137 011354 JMP CKHCE ;CHECK 'HCE' ERROR
3612 007456 032760 020000 000250 BIT #BIT13,$RPER1(RO) ;'OPI' SET?
3613 007464 001402 BEQ .+6 ;BR IF NOT SET
3614 007466 000137 011654 JMP OPIER ;REPORT 'OPI'
3615 007472 032760 000010 000250 BIT #BIT3,$RPER1(RO) ;'PAR' SET?
3616 007500 001402 BEQ .+6 ;BR IF NOT SET
3617 007502 000137 012006 JMP PARER ;REPORT 'PAR'
3618 007506 032760 000040 000250 BIT #BIT5,$RPER1(RO) ;'WCF' SET?
3619 007514 001402 BEQ .+6 ;BR IF NOT SET
3620 007516 000137 012456 JMP WCFER ;REPORT 'WCF'
3621 007522 032760 002000 000250 BIT #BIT10,$RPER1(RO) ;'IAE' SET?
3622 007530 001402 BEQ .+6 ;BR IF NOT SET
3623 007532 000137 012100 JMP IAEER ;REPORT 'IAE'
3624 007536 032760 004000 000250 BIT #BIT11,$RPER1(RO) ;'WLE' SET?
3625 007544 001402 BEQ .+6 ;BR IF NOT SET
3626 007546 000137 012132 JMP WLEER ;REPORT 'WLE'
3627 007552 032760 001000 000250 BIT #BIT9,$RPER1(RO) ;'AOE' SET?
3628 007560 001405 BEQ 2$ ;BR IF NOT SET
3629 007562 032760 002000 000246 BIT #BIT10,$RPDS1(RO) ;'LST' SET?
3630 007570 001401 BEQ 2$ ;BR IF NOT SET
3631 007572 000207 RTS PC ;'AOE' & 'LST' SET, EXIT
3632 007574 032760 010000 000250 2$: BIT #BIT12,$RPER1(RO) ;SEE IF 'DTE' SET
3633 007602 001402 BEQ .+6 ;BR IF NOT
3634 007604 000137 011764 JMP DTEER ;REPORT 'DTE' ERROR
3635 007610 032760 040000 000244 BIT #BIT14,$RPCS2(RO) ;SEE IF 'WCK' SET
3636 007616 001402 BEQ .+6 ;BR IF NOT SET
3637 007620 000137 010450 JMP WCKER ;REPORT 'WCK'
```



```
3638 007624 005760 000250          TST    $RPER1(R0)      ;SEE IF 'DCK' SET
3639 007630 100002                    BPL    .+6             ;BR IF NOT
3640 007632 000137 007656          JMP    DCKER          ;PROCESS 'DCK'
3641 007636 032760 140000 000276  BIT    #BIT15!BIT14,$RPER3(R0);'SKI' OR 'OCYL' SET
3642 007644 001402                    BEQ    .+6             ;BR IF NOT SET
3643 007646 000137 012420          JMP    SKIER          ;REPORT ERROR
3644 007652 000137 011126          JMP    DRIVER         ;REPORT DRIVE ERROR
3645
3646                                ;PROCESS DATA ('DCK') CHECK ERROR
3647
3648 007656 022760 010042 000300  DCKER: CMP    #10042,$RPEC1(R0) ;VALID POSITION COUNT ?
3649 007664 101406                    BLOS   1$             ;BR IF NOT VALID
3650 007666 005760 000300          TST    $RPEC1(R0)    ;POSITION COUNT 0 ?
3651 007672 001403                    BEQ    1$             ;BR IF 0'S
3652 007674 005760 000302          TST    $RPEC2(R0)    ;VALUE IN PATTERN REGISTER ?
3653 007700 001026                    BNE    4$             ;BR IF YES
3654 007702 004737 020266          1$:  JSR    PC,LINE1   ;TYPE FIRST LINE OF ERROR MESSAGE
3655 007706 104414 047311          DISPLY ,EM45         ;TYPE 'ECC LOGIC ERROR'
3656 007712 004737 020332          JSR    PC,LINE2     ;TYPE LINE 2 OF ERROR MESSAGE
3657 007716 004737 023512          JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
3658 007722 012737 000003 001252  MOV    #3,RETRY      ;RETRY COUNT
3659 007730 004737 015466          JSR    PC,$RETRY    ;RETRY THE ORDER
3660 007734 000403                    BR     2$             ;RETRY WAS NOT SUCCESSFUL
3661 007736 004737 021766          JSR    PC,LINE6C    ;TYPE LINE 6C OF ERROR MESSAGE
3662 007742 000402                    BR     3$             ;FINISH THE ERROR REPORT
3663 007744 004737 021774          2$:  JSR    PC,LINE6D  ;TYPE LINE 6D OF ERROR MESSAGE
3664 007750 004737 022050          3$:  JSR    PC,LINE7   ;TYPE LINE 7 OF ERROR MESSAGE
3665 007754 000402                    BR     5$             ;EXIT
3666 007756 004737 020130          4$:  JSR    PC,SPOTCK  ;SEE IF ERROR AT A BAD SPOT ON THE PACK
3667 007762 000207                    RTS    PC             ;IT IS, DON'T REPORT IT
3668 007764 126027 000024 000001  CMPB   $CODE(R0),#1  ;IS ORDER A WRITE CHECK ?
3669 007772 101002                    BHI    6$             ;BR IF NOT
3670 007774 000137 010450          JMP    WCKER         ;REPORT ERROR UNDER WRITE CHECK PROCESSING
3671 010000 004737 020266          6$:  JSR    PC,LINE1   ;PRINT LINE 1 OF ERROR MESSAGE
3672 010004 104414 045761          DISPLY ,EM21         ;DATA CHECK ERROR
3673 010010 004737 020332          DCKER1: JSR    PC,LINE2   ;PRINT LINE 2 OF ERROR MESSAGE
3674 010014 004737 020740          JSR    PC,LINE3     ;PRINT LINE 3 OF ERROR MESSAGE
3675 010020 004737 021414          JSR    PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
3676 010024 004737 015130          JSR    PC,PRTBAD    ;SEE IF BAD SECTOR TO BE PRINTED
3677 010030 012737 110100 001250  MOV    #BIT15!BIT12!BIT06,MASK ;LOAD ERROR MASK
3678 010036 032760 010100 000250  BIT    #BIT12!BIT06,$RPER1(R0);CHECK 'DTE' & 'ECH'
3679 010044 001003                    BNE    1$             ;BR IF SET
3680 010046 004737 021740          JSR    PC,LINE6     ;PRINT LINE 6 OF ERROR MESSAGE
3681 010052 000541                    BR     8$             ;FINISH THE ERROR REPORT
3682 010054 012737 000020 001252  1$:  MOV    #16.,RETRY   ;RETRY COUNT
3683 010062 005001                    CLR    R1             ;R1 IS OFFSET CODE POINTER
3684 010064 032760 000002 000262  BIT    #BIT01,$RPDT(R0) ;IS DRIVE AN RP06 ?
3685 010072 001002                    BNE    16$            ;BR IF IT IS
3686 010074 012701 000007          MOV    #7,R1         ;INCREMENT PAST RP06 OFFSET CODES
3687 010100 004737 016114          16$: JSR    PC,RELBUF   ;RELEASE THE BUFFER
3688 010104 004737 022402          JSR    PC,READDR    ;GET THE ADDRESS OF THE ERROR SECTOR
3689 010110 112660 000011          MOVB   (SP)+,$TRK(R0) ;TRACK ADDRESS OF ERROR SECTOR
3690 010114 112660 000010          MOVB   (SP)+,$SEC(R0) ;SECTOR ADDRESS OF ERROR SECTOR
3691 010120 016060 000272 000012  MOV    $RPCC(R0),$CYL(R0) ;PRESENT CYLINDER
3692 010126 026060 000022 000020  CMP    $SSEC(R0),$WRDL(R0) ;SEE IF TRANSFER LENGTH LESS THAN 1 SECTOR
3693 010134 103010                    BHIS   15$            ;BR IF IT IS; USE PRESENT TRANSFER LENGTH
```



```

3694 010136 016060 000022 000020      MOV      $SSEC(R0), $WRDL(R0) ;CHANGE TRANSFER SIZE TO 1 SECTOR
3695 010144 016060 000020 000004      MOV      $WRDL(R0), $WRDM(R0) ;SETUP WORD COUNT FOR OPERATION
3696 010152 005460 000004              NEG      $WRDM(R0) ;CHANGE COUNT TO 2'S COMP
3697 010156 005046              15$:    CLR      -(SP) ;SPACE FOR NEW BUFFER ADDRESS
3698 010160 004737 015760              JSR      PC, GETBUF ;GET A BUFFER
3699 010164 012660 000006              MOV      (SP)+, $BUF(R0) ;NEW BUFFER ADDRESS TO DPB
3700 010170 004737 016456              2$:    JSR      PC, GODRIV ;RETRY
3701 010174 005760 000016              3$:    TST      $STATUS(R0) ;TEST FOR DONE
3702 010200 001775              BEQ      3$ ;BR IF NOT DONE
3703 010202 100075              BPL      10$ ;BR IF NOT ERROR
3704 010204 032760 000200 000016      BIT      #BIT7, $STATUS(R0) ;SEE IF ORDER TERMINIATED NORMALLY
3705 010212 001006              BNE      14$ ;BR IF NOT
3706 010214 004737 023512              JSR      PC, INCTOT ;INCREMENT TOTAL ERROR COUNT
3707 010220 104414 051476              DISPLY  ,LIN8M ;'DIFFERENT ERROR DURING RETRY'
3708 010224 000137 006732              JMP      ERPRC1 ;SEE WHICH ERROR
3709 010230 033760 001250 000250 14$:    BIT      MASK, $RPER1(R0) ;LOOK AT CURRENT ERROR
3710 010236 001430              BEQ      5$ ;BR IF DIFFERENT ERROR
3711 010240 032760 010100 000250      BIT      #BIT12:BIT6, $RPER1(R0) ;'ECH' OR 'DTE' STILL SET ?
3712 010246 001437              BEQ      7$ ;BR IF NEITHER SET
3713 010250 105237 001253              INCB    RETRY+1 ;INCREMENT RETRY COUNT
3714 010254 123737 001252 001253      CMPB    RETRY, RETRY+1 ;DONE ?
3715 010262 001342              BNE      2$ ;BR IF NOT
3716 010264 005201              INC     R1 ;INCREMENT TABLE INDEX
3717 010266 116137 002220 045077      MOVWB  OFFCOD(R1), GENDPB+$FMT ;OFFSET CODE
3718 010274 001435              BEQ      9$ ;BR IF END OF OFFSET TABLE
3719 010276 062737 000002 001252      ADD     #2, RETRY ;NEW RETRY LIMIT
3720 010304 004737 015360              JSR      PC, OFFST ;OFFSET
3721 010310 005737 045114              4$:    TST      GENDPB+$STATUS ;SEE IF FINISHED WITH OFFSET
3722 010314 001775              BEQ      4$ ;BR IF NOT
3723 010316 100324              BPL      2$ ;BR IF NO ERROR PERFORMING OFFSET
3724 010320 004737 022316              5$:    JSR      PC, LINE8 ;PRINT LINE 8 OF ERROR MESSAGE
3725 010324 004737 023416              6$:    JSR      PC, INCHRD ;INCREMENT 'HARD' ERROR COUNT
3726 010330 004737 023512              JSR      PC, INCTOT ;INCREMENT TOTAL ERROR COUNT
3727 010334 004737 022050              JSR      PC, LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3728 010340 004737 015130              JSR      PC, PRIBAD ;PRINT THE BAD SECTOR
3729 010344 000436              BR      13$ ;CLEAN UP AND RETURN
3730 010346 004737 021760              7$:    JSR      PC, LINE6B ;PRINT LINE 6B OF ERROR MESSAGE
3731 010352 004737 021676              JSR      PC, LINE5B ;PRINT LINE 5B OF THE ERROR MESSAGE
3732 010356 004737 023372              8$:    JSR      PC, INCSOF ;INCREMENT 'SOFT' ERROR COUNT
3733 010362 004737 014370              JSR      PC, ECC ;CORRECT THE ERROR USING ECC AND CHECK IT
3734 010366 000407              BR      11$ ;COMPARE THE BUFFER
3735 010370 004737 021774              9$:    JSR      PC, LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3736 010374 000753              BR      6$ ;INCREMENT ERROR COUNT
3737 010376 004737 021752              10$:   JSR      PC, LINE6A ;PRINT LINE 6A OF ERROR MESSAGE
3738 010402 004737 023372              JSR      PC, INCSOF ;INCREMENT 'SOFT' ERROR COUNT
3739 010406 012737 000001 001300 11$:    MOV      #1, FRSTER ;SET PROCESSING 'DCKER' INDICATOR
3740 010414 004737 013260              JSR      PC, CMPARD ;COMPARE THE BUFFER
3741 010420 105737 001301              TSTB   FRSTER+1 ;ERROR IN COMPARE ?
3742 010424 100406              BMI     13$ ;BRANCH IF ERROR
3743 010426 004737 023512              JSR      PC, INCTOT ;INCREMENT TOTAL ERROR COUNT
3744 010432 104414 051710              DISPLY  ,LIN9G ;'DATA COMPARE OK' MESSAGE
3745 010436 004737 022050              12$:   JSR      PC, LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3746 010442 004737 006462              13$:   JSR      PC, REFORMAT ;REFORMAT THE ERROR SECTOR
3747 010446 000207              RTS     PC ;RETURN
3748
3749

```

;WRITE CHECK ERROR PROCESSING


```
3750
3751 010450 032760 100000 000250 WCKER: BIT #BIT15,$RPER1(RO) ;SEE IF 'DCK' SET ALSO
3752 010456 001034 BNE 2$ ;BR IF IT IS
3753 010460 004737 020266 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3754 010464 104414 046065 DISPLY ,EM23 ;PRINT WCE & DCK NOT
3755 010470 005037 001250 CLR MASK ;CLEAR ERROR MASK
3756 010474 004737 020332 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3757 010500 004737 020740 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3758 010504 004737 021414 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3759 010510 004737 021504 JSR PC,LINE5 ;PRINT LINE 5 OF ERROR MESSAGE
3760 010514 004737 023512 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3761 010520 012737 000003 001252 MOV #3,RETRY ;RETRY LIMIT
3762 010526 004737 015466 JSR PC,$RETRY ;RETRY THE OPERATION
3763 010532 000403 BR 1$ ;RETRY UNSUCCESSFUL
3764 010534 004737 021766 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3765 010540 000502 BR 8$ ;FINISH PROCESSING THE ERROR
3766 010542 004737 021774 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3767 010546 000506 BR 10$ ;FINISH PROCESSING THE ERROR
3768 010550 004737 020130 2$: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SPOT ON THE PACK
3769 010554 000507 BR 11$ ;EXIT IF AT BAD SPOT ON PACK
3770 010556 004737 020266 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3771 010562 012737 046012 010610 MOV #EM22,13$ ;ASSUME THAT EM22 WILL BE PRINTED
3772 010570 032760 040000 000244 BIT #BIT14,$RPCS2(RO) ;DID 'WCK' ALSO SET ?
3773 010576 001003 BNE 12$ ;BR IF IT DID
3774 010600 012737 046713 010610 MOV #EM37,13$ ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
3775 ;WRITE CHECK
3776 010606 104414 12$: DISPLY ;TYPE THE ERROR MESSAGE
3777 010610 000000 13$: .WORD 0 ;MESSAGE ADDRESS GOES HERE
3778 010612 004737 020332 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3779 010616 004737 020740 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3780 010622 004737 021414 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3781 010626 004737 021504 JSR PC,LINE5 ;PRINT LINE 5 OF ERROR MESSAGE
3782 010632 032760 000100 000250 BIT #BIT06,$RPER1(RO) ;ECH SET ALSO ?
3783 010640 001442 BEQ 8$ ;FINISH PROCESSING THE ERROR
3784 010642 012737 000020 001252 3$: MOV #16,RETRY ;RETRY LIMIT - 16 (10)
3785 010650 004737 016456 4$: JSR PC,GODRIV ;RETRY THE ORDER
3786 010654 005760 000016 5$: TST $STATUS(RO) ;ORDER FINISHED ?
3787 010660 001775 BEQ 5$ ;BR IF NOT
3788 010662 100405 BMI 6$ ;BR IF ERROR ON ORDER
3789 010664 105237 001253 INCB RETRY+1 ;INCREMENT RETRY COUNT
3790 010670 004737 021766 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3791 010674 000431 BR 9$ ;FINISH ERROR PROCESSING
3792 010676 105237 001253 6$: INCB RETRY+1 ;INCREMENT RETRY COUNT
3793 010702 123737 001252 001253 CMPB RETRY,RETRY+1 ;DONE ?
3794 010710 001714 BEQ 1$ ;BR IF AT RETRY LIMIT
3795 010712 032760 100000 000250 BIT #BIT15,$RPER1(RO) ;'DCK' SET
3796 010720 001407 BEQ 7$ ;BR IF NOT - DIFFERENT ERROR
3797 010722 032760 000100 000250 BIT #BIT06,$RPER1(RO) ;'ECH' ALSO SET ?
3798 010730 001347 BNE 4$ ;BR IF IT IS, RETRY ORDER
3799 010732 004737 021766 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3800 010736 000403 BR 8$ ;FINISH PROCESSING ERROR
3801 010740 004737 022316 7$: JSR PC,LINE8 ;PRINT LINE 8 - 'DIFFERENT ERROR '
3802 010744 000405 BR 9$ ;FINISH PROCESSING ERROR
3803 010746 004737 023512 8$: JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3804 010752 004737 022050 JSR PC,LINE7 ;FINISH THE ERROR MESSAGE
3805 010756 000406 BR 11$ ;EXIT
```



```

3806 010760 004737 023512      9$:   JSR   PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3807 010764 004737 022050      10$:  JSR   PC,LINE7       ;FINISH THE ERROR MESSAGE
3808 010770 004737 006462      JSR   PC,REFMT        ;REFORMAT THE SECTOR IN ERROR
3809 010774 000207              RTS    PC              ;RETURN
3810
3811      ;REPORT 'HCRC' ERROR
3812
3813 010776 004737 020130      HCRCER: JSR   PC,SPOTCK     ;SEE IF ERROR AT PACK BAD SPOT
3814 011002 000450              BR     3$             ;EXIT IF IT IS
3815 011004 004737 020266      JSR   PC,LINE1       ;PRINT LINE 1 OF ERROR MESSAGE
3816 011010 104414 045740      DISPLY ,EM20         ;REPORT 'HCRC'
3817 011014 004737 020332      JSR   PC,LINE2       ;PRINT LINE 2 OF ERROR MESSAGE
3818 011020 004737 020740      JSR   PC,LINE3       ;PRINT LINE 3 OF ERROR MESSAGE
3819 011024 004737 021414      JSR   PC,LINE4       ;PRINT LINE 4 OF ERROR MESSAGE
3820 011030 032760 040000      BIT   #BIT14,$RPCS2(R0) ;'WCE' ERROR ALSO ?
3821 011036 001402              BEQ   1$             ;BR IF NOT
3822 011040 004737 021504      JSR   PC,LINES5      ;DISPLAY WORDS WHICH CAUSED 'WCE'
3823 011044 004737 023372      1$:   JSR   PC,INCSOF     ;INCREMENT 'SOFT' ERROR COUNT
3824 011050 004737 023512      JSR   PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3825 011054 012737 000400      MOV   #BIT8,MASK     ;SET ERROR MASK
3826 011062 012737 000003      MOV   #3,RETRY       ;RETRY LIMIT
3827 011070 004737 015466      JSR   PC,$RETRY      ;RETRY ORDER
3828 011074 000405              BR     2$             ;RETRY NOT SUCCESSFUL
3829 011076 004737 021766      JSR   PC,LINE6C      ;PRINT LINE 6C OF ERROR MESSAGE
3830 011102 004737 022050      JSR   PC,LINE7       ;PRINT LINE 7 OF ERROR MESSAGE
3831 011106 000406              BR     3$             ;EXIT
3832 011110 004737 021774      2$:   JSR   PC,LINE6D     ;PRINT LINE 6D OF ERROR MESSAGE
3833 011114 004737 022050      JSR   PC,LINE7       ;PRINT LINE 7 OF ERROR MESSAGE
3834 011120 004737 006462      JSR   PC,REFMT        ;REFORMAT THE ERROR SECTOR
3835 011124 000207              3$:   RTS    PC              ;RETURN
3836
3837      ;REPORT DRIVE ERROR
3838
3839 011126 004737 020266      DRVER: JSR   PC,LINE1     ;PRINT LINE 1 OF ERROR MESSAGE
3840 011132 104414 046355      DISPLY ,EM30         ;REPORT DRIVE ERROR
3841 011136 004737 020332      JSR   PC,LINE2       ;PRINT LINE 2 OF ERROR MESSAGE
3842 011142 004737 020740      JSR   PC,LINE3       ;PRINT LINE 3 OF ERROR MESSAGE
3843 011146 004737 023512      JSR   PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3844 011152 004737 022050      JSR   PC,LINE7       ;PRINT LINE 7 OF ERROR MESSAGE
3845 011156 000207              RTS    PC              ;RETURN
3846
3847      ;PROCESS FORMAT ('FER') ERROR
3848
3849 011160 032760 000400      000250 CKFMT: BIT   #BIT8,$RPER1(R0) ;'HCRC' SET ON ORIGINAL ERROR ?
3850 011166 001402              BEQ   1$             ;BR IF NOT SET
3851 011170 000137 010776      JMP   HCRCER         ;REPORT HCRC ERROR
3852 011174 004737 022402      1$:   JSR   PC,READDR     ;GET CORRECTED TRACK & SECTOR ADDRSSES
3853 011200 004737 015404      JSR   PC,READHD      ;READ HEADER
3854 011204 032737 000400      045132 BIT   #BIT8,GENREG+RPER1 ;'HCRC' SET WHEN HEADER READ?
3855 011212 001002              BNE   2$             ;BR IF 'HCRC' SET
3856 011214 000137 012160      JMP   FMTER          ;NO, ERROR IS 'FMT' ONLY
3857 011220 004737 020130      2$:   JSR   PC,SPOTCK     ;SEE IF ERROR AT BAD SPOT ON THE PACK
3858 011224 000452              BR     5$             ;EXIT IF IT IS
3859 011226 004737 020266      JSR   PC,LINE1       ;PRINT LINE 1 OF ERROR MESSAGE
3860 011232 104414 046144      DISPLY ,EM24         ;HEADER READ ERROR - FMT BIT DROPPED UP
3861 011236 004737 020332      JSR   PC,LINE2       ;PRINT LINE 2 OF ERROR MESSAGE

```



```
3862 011242 004737 020740 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3863 011246 004737 021414 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3864 011252 032760 040000 000244 BIT #BIT14,$RPCS2(R0) ;'WCE' ERROR ALSO ?
3865 011260 001402 BEQ 3$ ;BR IF NOT
3866 011262 004737 021504 JSR PC,LINE5 ;DISPLAY WORDS WHICH CAUSED 'WCE'
3867 011266 004737 021604 3$: JSR PC,LINE5A ;DISPLAY HEADER
3868 011272 004737 023372 JSR PC,INCSOF ;INCREMENT SOFT ERROR COUNT
3869 011276 004737 023512 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3870 011302 012737 000020 001250 MOV #BIT4,MASK ;SET ERROR MASK
3871 011310 012737 000003 001252 MOV #3,RETRY ;RETRY LIMIT
3872 011316 004737 015466 JSR PC,$RETRY ;RETRY THE ORDER
3873 011322 000405 BR 4$ ;RETRY NOT SUCCESSFUL
3874 011324 004737 021766 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3875 011330 004737 022050 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3876 011334 000406 BR 5$ ;EXIT
3877 011336 004737 021774 4$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3878 011342 004737 022050 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3879 011346 004737 006462 JSR PC,REFMT ;REFORMAT THE ERROR SECTOR
3880 011352 000207 5$: RTS PC ;RETURN
3881
3882 ;PROCESS HEADER COMPARE ('HCE') ERROR
3883
3884 011354 032760 000400 000250 CKHCE: BIT #BIT8,$RPER1(R0) ;HCRC SET ON ORIGINAL ERROR ?
3885 011362 001402 BEQ 1$ ;BR IF NOT SET
3886 011364 000137 010776 JMP HRCRCR ;REPORT HEADER CRC ERROR
3887 011370 004737 022402 1$: JSR PC,READDR ;GET CURRENT SECTOR & TRACK ADDRS
3888 011374 004737 015404 JSR PC,READHD ;READ HEADER OF CURRENT SECTOR
3889 011400 032737 000400 045132 BIT #BIT8,GENREG+RPER1 ;'HCRC' SET ?
3890 011406 001016 BNE 3$ ;BR IF SET
3891 011410 042737 010000 054746 BIC #BIT12,CYLDER ;CLEAR FORMAT BIT FROM HEADER
3892 011416 026037 000272 054746 CMP $RPCC(R0),CYLDER ;CORRECT CYLINDER ?
3893 011424 001402 BEQ 2$ ;BR IF IT IS
3894 011426 000137 011600 JMP POSER ;REPORT POSITIONING ERROR
3895 011432 052737 010000 054746 2$: BIS #BIT12,CYLDER ;RESTORE THE FORMAT BIT
3896 011440 000137 012236 JMP HCEER ;REPORT 'HCE' ERROR
3897 011444 004737 020130 3$: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
3898 011450 000452 BR 6$ ;EXIT IF IT IS
3899 011452 004737 020266 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3900 011456 104414 046212 DISPLY ,EM25 ;HEADER READ ERROR - 'HCE' SET
3901 011462 004737 020332 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3902 011466 004737 020740 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3903 011472 004737 021414 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3904 011476 032760 040000 000244 BIT #BIT14,$RPCS2(R0) ;'WCE' ERROR ALSO ?
3905 011504 001402 BEQ 4$ ;BR IF NOT
3906 011506 004737 021504 JSR PC,LINE5 ;DISPLAY WORDS WHICH CAUSED 'WCE'
3907 011512 004737 021604 4$: JSR PC,LINE5A ;PRINT LINE 5 OF ERROR MESSAGE
3908 011516 004737 023372 JSR PC,INCSOF ;INCREMENT SOFT ERROR COUNT
3909 011522 004737 023512 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3910 011526 012737 000200 001250 MOV #BIT7,MASK ;SET ERROR MASK
3911 011534 012737 000003 001252 MOV #3,RETRY ;RETRY LIMIT
3912 011542 004737 015466 JSR PC,$RETRY ;RETRY THE ORDER
3913 011546 000405 BR 5$ ;RETRY NOT SUCCESSFUL
3914 011550 004737 021766 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3915 011554 004737 022050 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3916 011560 000406 BR 6$ ;EXIT
3917 011562 004737 021774 5$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
```



```
3918 011566 004737 022050          JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3919 011572 004737 006462          JSR    PC,REFMT      ;REFORMAT THE ERROR SECTOR
3920 011576 000207                   6$:   RTS    PC      ;RETURN
3921
3922                               ;REPORT POSSIBLE POSITIONING ERROR
3923
3924 011600 004737 015326          POSER: JSR    PC,RECALT ;RECALIBRATE
3925 011604 004737 020266          JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
3926 011610 104414 047540          DISPLY ,EM51         ;PROGRAM DETECTED POSITIONING ERROR
3927 011614 004737 020332          JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3928 011620 004737 020766          JSR    PC,LINE3C     ;PRINT LINE 3C OF ERROR MESSAGE
3929 011624 052737 010000          054746 BIS    #BIT12,CYLDER ;RESTORE THE FORMAT BIT
3930 011632 004737 021604          JSR    PC,LINE5A     ;PRINT LINE 5A OF THE ERROR MESSAGE
3931 011636 004737 023466          JSR    PC,INCMIS     ;INCREMENT MISPOSITIONING COUNT
3932 011642 004737 023512          JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
3933 011646 004737 022176          JSR    PC,LINE7A     ;PRINT LINE 7A OF ERROR MESSAGE
3934 011652 000207                   RTS    PC      ;EXIT
3935
3936                               ;REPORT 'OPI' ERROR
3937
3938 011654 004737 020130          OPIER: JSR    PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
3939 011660 000207                   RTS    PC      ;RETURN IF IT IS
3940 011662 004737 020266          JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
3941 011666 104414 046407          DISPLY ,EM31         ;'OPI' ERROR
3942 011672 004737 020332          JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3943 011676 004737 020740          JSR    PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
3944 011702 004737 021414          JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
3945 011706 004737 023512          JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
3946 011712 012737 020000          001250 MOV    #BIT13,MASK   ;ERROR MASK
3947 011720 012737 000003          001252 OPIER1: MOV   #3,RETRY ;RETRY LIMIT
3948 011726 004737 015466          JSR    PC,$RETRY     ;RETRY THE ORDER
3949 011732 000405                   BR     1$           ;RETRY UNSUCCESSFUL
3950 011734 004737 021766          JSR    PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
3951 011740 004737 022050          JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3952 011744 000207                   RTS    PC      ;EXIT
3953 011746 004737 021774          1$:   JSR    PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3954 011752 004737 022050          JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3955 011756 004737 006462          JSR    PC,REFMT      ;REFORMAT THE ERROR SECTOR
3956 011762 000207                   RTS    PC      ;RETURN
3957
3958                               ;REPORT 'DTE' ERROR
3959
3960 011764 004737 020130          DTEER: JSR    PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
3961 011770 000207                   RTS    PC      ;RETURN IF IT IS
3962 011772 004737 020266          JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
3963 011776 104414 046452          DISPLY ,EM32         ;'DTE' ERROR
3964 012002 000137 010010          JMP    DCKER1        ;FINISH PROCESSING THE 'DTE' ERROR
3965
3966                               ;REPORT 'PAR' ERROR
3967
3968 012006 004737 020266          PARER: JSR    PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3969 012012 104414 046505          DISPLY ,EM33         ;REPORT 'PAR'
3970 012016 004737 020332          JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3971 012022 004737 021044          JSR    PC,LINE3E     ;PRINT LINE 3E OF ERROR MESSAGE
3972 012026 004737 021414          JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
3973 012032 004737 023512          JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
```



```
3974 012036 012737 000010 001250      MOV      #BIT03,MASK      ;ERROR MASK
3975 012044 012737 000003 001252      MOV      #3,RETRY       ;RETRY LIMIT
3976 012052 004737 015466      JSR      PC,$RETRY      ;RETRY ORDER
3977 012056 000405          BR       2$             ;RETRY UNSUCCESSFUL
3978 012060 004737 021766      JSR      PC,LINE6C      ;RETRY SUCCESSFUL
3979 012064 004737 022050      1$:     JSR      PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3980 012070 000207          RTS      PC             ;EXIT
3981 012072 004737 021774      2$:     JSR      PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3982 012076 000772          BR       1$             ;FINISH ERROR MESSAGE
3983
3984      ;REPORT 'IAE' ERROR
3985
3986 012100 004737 020266      IAEER:  JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3987 012104 104414 046624      DISPLY  ,EM35          ;REPORT 'IAE'
3988 012110 004737 020332      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3989 012114 004737 021132      JSR      PC,LINE3F     ;PRINT LINE 3F OF ERROR MESSAGE
3990 012120 004737 023512      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
3991 012124 004737 022050      JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3992 012130 000207          RTS      PC             ;RETURN
3993
3994      ;REPORT 'WLE' ERROR
3995
3996 012132 004737 020266      WLEER:  JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3997 012136 104414 046662      DISPLY  ,EM36          ;REPORT 'WLE'
3998 012142 004737 020332      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3999 012146 004737 023512      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
4000 012152 004737 022050      JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
4001 012156 000207          RTS      PC             ;RETURN
4002
4003      ;REPORT FORMAT ERROR
4004
4005 012160 004737 020266      FMTER:  JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4006 012164 104414 046273      DISPLY  ,EM26          ;FORMAT ERROR
4007 012170 004737 020332      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4008 012174 004737 020740      JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
4009 012200 004737 021414      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
4010 012204 032760 040000 000244      BIT     #BIT14,$RPCS2(R0) ;'WCE' ERROR ALSO ?
4011 012212 001402          BEQ     1$             ;BR IF NOT
4012 012214 004737 021504      1$:     JSR      PC,LINE5 ;DISPLAY WORDS WHICH CAUSED 'WCE'
4013 012220 004737 021604      JSR      PC,LINE5A     ;PRINT LINE 5A OF ERROR MESSAGE
4014 012224 004737 023512      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
4015 012230 004737 022050      JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
4016 012234 000207          RTS      PC
4017
4018      ;REPORT HEADER COMPARE ERROR
4019
4020 012236 004737 020266      HCEER:  JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4021 012242 104414 046320      DISPLY  ,EM27          ;HEADER COMPARE ERROR
4022 012246 004737 020332      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4023 012252 004737 020740      JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
4024 012256 004737 021414      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
4025 012262 032760 040000 000244      BIT     #BIT14,$RPCS2(R0) ;'WCE' ERROR ALSO ?
4026 012270 001402          BEQ     1$             ;BR IF NOT
4027 012272 004737 021504      1$:     JSR      PC,LINE5 ;DISPLAY WORDS WHICH CAUSED 'WCE'
4028 012276 004737 021604      JSR      PC,LINE5A     ;PRINT LINE 5A OF ERROR MESSAGE
4029 012302 004737 023512      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
```



```
4030 012306 004737 022050      JSR   PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
4031 012312 004737 006462      JSR   PC,REFMT      ;REFORMAT THE ERROR SECTOR
4032 012316 000207              RTS   PC             ;RETURN
4033
4034      ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
4035
4036 012320 004737 020266      TRFER: JSR   PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
4037 012324 104414 046775      DISPLY ,EM40        ;RH11 OR UNIBUS TRANSFER ERROR
4038 012330 004737 020332      JSR   PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4039 012334 004737 020740      JSR   PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
4040 012340 004737 021414      JSR   PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
4041 012344 004737 023512      JSR   PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
4042 012350 032760 121400 000244 BIT   #BIT15!BIT13!BIT9!BIT8,$RPCS2(RO) ;'DLT','UPE','MXF','MDPE' SET ?
4043 012356 001415              BEQ   2$             ;BR IF NONE SET
4044 012360 012737 000003 001252 MOV   #3,RETRY      ;RETRY LIMIT
4045 012366 005037 001250      CLR   MASK          ;CLEAR ERROR MASK
4046 012372 004737 015466      JSR   PC,$RETRY     ;RETRY THE OPERATION
4047 012376 000403              BR    1$            ;RETURN HERE IF RETRY UNSUCCESSFUL
4048 012400 004737 021766      JSR   PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
4049 012404 000402              BR    2$            ;FINISH THE ERROR REPORT
4050 012406 004737 021774      1$: JSR   PC,LINE6D   ;PRINT LINE 6D OF ERROR MESSAGE
4051 012412 004737 022050      2$: JSR   PC,LINE7   ;PRINT LINE 7 OF ERROR MESSAGE
4052 012416 000207              RTS   PC
```

4053
4054 ;PROCESS 'SKI' OR 'OCYL' ERRORS

```
4055
4056 012420 004737 020266      SKIER: JSR   PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
4057 012424 104414 047451      DISPLY ,EM50        ;'SKI' OR 'OCYL' ERROR
4058 012430 004737 020332      JSR   PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4059 012434 004737 020754      JSR   PC,LINE3B     ;PRINT LINE 3B OF ERROR MESSAGE
4060 012440 004737 023512      JSR   PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
4061 012444 004737 023442      JSR   PC,INCSKI     ;INCREMENT 'SKI' OR 'OCYL' ERROR COUNT
4062 012450 004737 022176      JSR   PC,LINE7A     ;PRINT LINE 7A OF ERROR MESSAGE
4063 012454 000207              RTS   PC
```

4064
4065 ;REPORT WRITE CLOCK FAILURE ('WCF')

```
4066
4067 012456 004737 020266      WCFER: JSR   PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
4068 012462 104414 046562      DISPLY ,EM34        ;REPORT WRITE CLOCK FAILURE
4069 012466 004737 020332      JSR   PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4070 012472 004737 020746      JSR   PC,LINE3A     ;PRINT LINE 3A OF ERROR MESSAGE
4071 012476 004737 021414      JSR   PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
4072 012502 004737 023512      JSR   PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
4073 012506 004737 015130      JSR   PC,PRTBAD     ;SEE IF BAD SECTOR TO BE PRINTED
4074 012512 012737 000003 001252 MOV   #3,RETRY      ;RETRY COUNT
4075 012520 012737 000040 001250 MOV   #BIT05,MASK   ;ERROR MASK
4076 012526 004737 015466      JSR   PC,$RETRY     ;RETRY THE ORDER
4077 012532 000405              BR    2$            ;RETURN HERE IF RETRY UNSUCCESSFUL
4078 012534 004737 021766      JSR   PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
4079 012540 004737 022050      1$: JSR   PC,LINE7   ;PRINT LINE 7 OF ERROR MESSAGE
4080 012544 000207              RTS   PC
4081 012546 004737 021774      2$: JSR   PC,LINE6D   ;PRINT LINE 6D OF ERROR MESSAGE
4082 012552 000772              BR    1$
```

4083
4084 ;PROCESS DRIVE UNSAFE ERROR

4085


```
4086 012554 004737 020266 UNSAF: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4087 012560 104414 047603 DISPLY ,EM60 ;REPORT DRIVE UNSAFE
4088 012564 004737 020332 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4089 012570 004737 020740 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
4090 012574 004737 023512 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4091 012600 012737 000003 001252 MOV #3,RETRY ;RETRY COUNT
4092 012606 004737 015466 JSR PC,$RETRY ;RETRY THE ORDER
4093 012612 000403 BR 1$ ;RETRY WAS UNSUCCESSFUL
4094 012614 004737 021766 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
4095 012620 000402 BR 2$ ;CONTINUE WITH ERROR REPORT
4096 012622 004737 021774 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
4097 012626 004737 022050 2$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4098 012632 000207 RTS PC ;RETURN
4099
4100 ;REPORT AN 'UNKNOWN' DATA PATTERN
4101
4102 012634 105737 001300 NOMTCH: TSTB FRSTER ;FIRST ERROR IN THE SECTOR ?
4103 012640 001013 BNE 1$ ;BR IF NOT OR IF PROCESSING 'DCKER'
4104 012642 004737 020266 JSR PC,LINE1 ;TYPE LINE 1 OF ERROR MESSAGE
4105 012646 104414 047160 DISPLY ,EM43 ;'CAN'T MATCH DATA WITH PATTERN'
4106 012652 004737 020332 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4107 012656 004737 020746 JSR PC,LINE3A ;PRINT LINE 3A OF ERROR MESSAGE
4108 012662 004737 021414 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4109 012666 000404 BR 2$ ;CONTINUE PROCESSING ERROR
4110 012670 104414 047160 1$: DISPLY ,EM43 ;'CAN'T MATCH DATA WITH PATTERN'
4111 012674 104414 001165 DISPLY ,$CRLF ;CR-LF
4112 012700 104414 051640 2$: DISPLY ,LIN91 ;HEADER FOR DATA PRINTOUT
4113 012704 010146 MOV R1,-(SP) ;ADDRESS OF WORD 1
4114 012706 004737 022330 JSR PC,LINOC1 ;TYPE WORD 1
4115 012712 104414 052323 DISPLY ,LINS1 ;SPACES
4116 012716 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 1
4117 012720 004737 022330 JSR PC,LINOC1 ;TYPE WORD 1
4118 012724 104414 001165 DISPLY ,$CRLF ;CR-LF
4119 012730 010146 MOV R1,-(SP) ;ADDRESS OF WORD 2
4120 012732 004737 022330 JSR PC,LINOC2 ;TYPE WORD 2
4121 012736 104414 052323 DISPLY ,LINS2 ;SPACES
4122 012742 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 2
4123 012744 004737 022330 JSR PC,LINOC2 ;TYPE WORD 2
4124 012750 104414 001165 DISPLY ,$CRLF ;CR-LF
4125 012754 010146 MOV R1,-(SP) ;ADDRESS OF WORD 3
4126 012756 004737 022330 JSR PC,LINOC3 ;TYPE WORD 3
4127 012762 104414 052323 DISPLY ,LINS3 ;SPACES
4128 012766 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 3
4129 012770 004737 022330 JSR PC,LINOC3 ;TYPE WORD 3
4130 012774 104414 001165 DISPLY ,$CRLF ;CR-LF
4131 013000 010146 MOV R1,-(SP) ;ADDRESS OF WORD 4
4132 013002 004737 022330 JSR PC,LINOC4 ;TYPE WORD 4
4133 013006 104414 052323 DISPLY ,LINS4 ;SPACES
4134 013012 012146 MOV (R1)+,-(SP) ;ADDRESS OF WORD 4
4135 013014 004737 022330 JSR PC,LINOC4 ;TYPE WORD 4
4136 013020 104414 001165 DISPLY ,$CRLF ;CR-LF
4137 013024 062701 000770 ADD #<252.*2.>,R1 ;INCREMENT BUFFER POINTER
4138 013030 005002 CLR R2 ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
4139 013032 112737 000001 001300 MOVB #1,FRSTER ;SET 'NOT FIRST ERROR' INDICATOR
4140 013040 112737 177777 001301 MOVB #-1,FRSTER+1 ;SET ERROR FOUND INDICATOR
4141 013046 013737 001414 001310 MOV CMLMT,LIMIT ;RESET THE COMPARE ERROR TYPEOUT LIMIT
```



```

4142 013054 000207          RTS      PC          ;RETURN
4143
4144          ;CHECK ERROR BITS IN THE RH11 & RP04/5/6 REGISTERS
4145
4146 013056 032760 060000 000234 CKERR: BIT      #60000,$RPCS1(R0) ;SEE IF 'TRE' OR 'MCPE' SET
4147 013064 001015          BNE      1$          ;BR IF EITHER SET
4148 013066 032760 177400 000244          BIT      #177400,$RPCS2(R0) ;SEE IF ERROR BITS IN CS2 SET
4149 013074 001011          BNE      1$          ;BR IF ANY SET
4150 013076 005760 000250          TST     $RPER1(R0)    ;ANY BITS SET IN ER1
4151 013102 001006          BNE      1$          ;BR IF ANY SET
4152 013104 005760 000274          TST     $RPER2(R0)    ;ANY BITS SET IN ER2 ?
4153 013110 001003          BNE      1$          ;BR IF ANY SET
4154 013112 005760 000276          TST     $RPER3(R0)    ;ANY BITS SET IN ER3 ?
4155 013116 001416          BEQ      2$          ;BR IF NONE SET
4156 013120 004737 020266          1$:    JSR     PC,LINE1  ;PRINT LINE 1 OF ERROR MESSAGE
4157 013124 104414 047225          DISPLY  ,EM44         ;ERROR BITS SET, BUT 'SC' OR 'TRE' NOT SET
4158 013130 004737 020332          JSR     PC,LINE2  ;PRINT LINE 2 OF ERROR MESSAGE
4159 013134 004737 020740          JSR     PC,LINE3  ;PRINT LINE 3 OF ERROR MESSAGE
4160 013140 004737 021414          JSR     PC,LINE4  ;PRINT LINE 4 OF ERROR MESSAGE
4161 013144 004737 023512          JSR     PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4162 013150 004737 022050          JSR     PC,LINE7  ;PRINT LINE 7 OF ERROR MESSAGE
4163 013154 000207          2$:    RTS      PC          ;RETURN
4164
4165          ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
4166
4167 013156 005760 000236          CKBUS: TST     $RPWC(R0) ;CHECK WORD COUNT
4168 013162 001010          BNE      1$          ;BR IF NOT ZERO
4169 013164 016046 000020          MOV     $WRDL(R0),-(SP) ;WORD LENGTH
4170 013170 006316          ASL     (SP)         ;CHANGE INTO BYTE COUNT
4171 013172 066016 000006          ADD     $BUF(R0),(SP) ;ADD THE STARTING LOCATION
4172 013176 022660 000240          CMP     (SP)+,$RPBA(R0) ;BUFFER ADDRESS PROPER ?
4173 013202 001416          BEQ      2$          ;BR IF OK
4174 013204 004737 020266          1$:    JSR     PC,LINE1  ;PRINT LINE 1 OF ERROR MESSAGE
4175 013210 104414 047033          DISPLY  ,EM41         ;BUS ADDRESS OR WORD COUNT INCORRECT
4176 013214 004737 020332          JSR     PC,LINE2  ;PRINT LINE 2 OF ERROR MESSAGE
4177 013220 004737 020776          JSR     PC,LINE3D ;PRINT LINE 3D OF ERROR MESSAGE
4178 013224 004737 021414          JSR     PC,LINE4  ;PRINT LINE 4 OF ERROR MESSAGE
4179 013230 004737 023512          JSR     PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4180 013234 004737 022050          JSR     PC,LINE7  ;PRINT LINE 7 OF ERROR MESSAGE
4181 013240 000207          2$:    RTS      PC
4182
4183          ;COMPARE THE BUFFER
4184
4185 013242 132760 000004 000024          CMPAR: BITB     #BIT02,$CODE(R0) ;SEE IF READ ORDER
4186 013250 001001          BNE      1$          ;BR IF IT IS
4187 013252 000207          RTS      PC          ;RETURN
4188 013254 005037 001300          1$:    CLR     FRSTER    ;CLEAR 'FIRST ERROR' INDICATOR
4189 013260 032777 000002 165652          CMPARD: BIT     #SW01,@SWR    ;IS SWITCH 1 SET?
4190 013266 001401          BEQ      1$          ;BR IF NOT
4191 013270 000207          RTS      PC          ;YES, DON'T COMPARE
4192 013272 005037 001306          1$:    CLR     ERCTR    ;CLEAR THE ERROR COUNTER
4193 013276 016001 000006          MOV     $BUF(R0),R1  ;BUFFER ADDRESS
4194 013302 016037 000020 001312          MOV     $WRDL(R0),CMCNT ;WORD COUNT TO WORKING LOCATION
4195 013310 066037 000236 001312          ADD     $RPWC(R0),CMCNT ;CALCULATE ACTUAL WORDS TRANSFERED
4196 013316 016037 000012 001314          MOV     $CYL(R0),CMCYL ;CYLINDER ADDRESS WORKING LOCATION
4197 013324 052737 010000 001314          BIS     #BIT12,CMCYL  ;SET FORMAT BIT

```



```
4198 013332 016037 000010 001316      MOV      $SEC(R0),CMSEC ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
4199 013340 013737 001414 001310      MOV      CMPLMT,LIMIT ;DISPLAY LIMIT
4200 013346 005237 001310      INC      LIMIT ;CONVERT PARAMETER INTO LIMIT VALUE
4201 013352 012737 177777 001276 CMSTR:  MOV      #-1,ZROIND ;CLEAR THE 'ZERO'S' INDICATOR
4202 013360 005037 001302      CLR      SAVER1 ;CLEAR THE R1 SAVE WORD
4203 013364 005037 001304      CLR      SAVER5 ;CLEAR THE R5 SAVE WORD
4204 013370 023760 001312 000022      CMP      CMCNT,$SSEC(R0) ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
4205 013376 101003      BHI      1$ ;BR IF IT IS
4206 013400 013702 001312      MOV      CMCNT,R2 ;LESS THAN, USE REMAINING BUFFER
4207 013404 000402      BR      2$ ;
4208 013406 016002 000022      1$: MOV      $SSEC(R0),R2 ;COMPARE SECTOR
4209 013412 166037 000022 001312 2$: SUB      $SSEC(R0),CMCNT ;DECREMENT WORD COUNT
4210 013420 126027 000024 000005      CMPB     $CODE(R0),#5 ;READ HEADER & DATA?
4211 013426 001036      BNE      CMDAT ;BR IF NOT
4212 013430 023721 001314      CMHED:  CMP      CMCYL,(R1)+ ;CHECK CYLINDER
4213 013434 001402      BEQ      1$ ;BR IF COMPARE OK
4214 013436 004737 013512      JSR      PC,5$ ;REPORT ERROR
4215 013442 023721 001316      1$: CMP      CMSEC,(R1)+ ;COMPARE SECTOR & TRACK
4216 013446 001402      BEQ      2$ ;BR IF EQ
4217 013450 004737 013512      JSR      PC,5$ ;REPORT ERROR
4218 013454 005721      2$: TST      (R1)+ ;1ST KEY WORD ZERO?
4219 013456 001402      BEQ      3$ ;BR IF IT IS
4220 013460 004737 013512      JSR      PC,5$ ;REPORT ERROR
4221 013464 005721      3$: TST      (R1)+ ;CHECK 2ND KEY WORD
4222 013466 001402      BEQ      4$ ;BR IF ZERO
4223 013470 004737 013512      JSR      PC,5$ ;REPORT THE ERROR
4224 013474 162702 000004      4$: SUB      #4,R2 ;SUBTRACT HEADER LENGTH FROM SIZE
4225 013500 003530      BLE      CMPRX ;BR IF FINISHED
4226 013502 022702 000004      CMP      #4,R2 ;SEE IF AT LEAST 4 MORE WORDS TO CHECK
4227 013506 101125      BHI      CMPRX ;BR IF NOT
4228 013510 000405      BR      CMDAT ;COMPARE THE DATA PORTION
4229 013512 005237 001306      5$: INC      ERCTR ;INCREMENT THE ERROR COUNT
4230 013516 004737 013770      JSR      PC,CMPRT ;REPORT THE COMPARISON ERROR
4231 013522 000207      RTS      PC ;CHECK THE REST OF THE HEADER
4232 013524 004737 014312      CMDAT:  JSR      PC,MATCH ;FIND THE PATTERN
4233 013530 000403      BR      2$ ;FOUND A PATTERN
4234 013532 004737 012634      JSR      PC,NOMTCH ;RETURN HERE IF NO MATCH WITH PATTERN MADE
4235 013536 000456      BR      8$ ;BYPASS COMPARE ROUTINE
4236 013540 011405      2$: MOV      (R4),R5 ;ADDRESS OF PATTERN ADDRESS IN R4
4237 013542 012703 000020      MOV      #20,R3 ;R3 IS PATTERN POS COUNTER
4238 013546 022125      3$: CMP      (R1)+,(R5)+ ;COMPARE BUFFER WITH PATTERN
4239 013550 001016      BNE      5$ ;BR IF NOT EQUAL
4240 013552 005737 001306      TST      ERCTR ;ERRORS DETECTED ?
4241 013556 001406      BEQ      4$ ;BR IF NO ERRORS
4242 013560 032777 000010 165352      BIT      #SW3,@SWR ;SWITCH 3 SET ?
4243 013566 001402      BEQ      4$ ;BR IF NOT SET
4244 013570 004737 013770      JSR      PC,CMPRT ;DISPLAY THE WORD
4245 013574 005302      4$: DEC      R2 ;DECREMENT SIZE COUNT
4246 013576 001436      BEQ      8$ ;BR WHEN AT END
4247 013600 005303      DEC      R3 ;DECREMENT PATT POS COUNT
4248 013602 001361      BNE      3$ ;BR IF NOT AT END OF PATT
4249 013604 000755      BR      2$ ;RESTART THE PATTERN
4250 013606 005761 177776      5$: TST      -2(R1) ;IS MISCOMPARED CHARACTER=0
4251 013612 001410      BEQ      6$ ;BR IF YES
4252 013614 012737 177777 001276      MOV      #-1,ZROIND ;SET NON-ZERO MISCOMPARED INDICATOR
4253 013622 005237 001306      INC      ERCTR ;INCREMENT THE ERROR COUNTER
```



```
4254 013626 004737 013770      JSR    PC,CMPRT      ;REPORT ERROR
4255 013632 000760              BR     4$            ;CONTINUE COMPARE
4256 013634 105737 001300      6$:  TSTB   FRSTER   ;FIRST ERROR?
4257 013640 100407              BMI    7$            ;BR IF NOT
4258 013642 005037 001276      CLR    ZROIND       ;SET THE ZERO INDICATOR
4259 013646 010137 001302      MOV    R1,SAVER1    ;SAVE CURRENT R1
4260 013652 010537 001304      MOV    R5,SAVER5    ;SAVE CURRENT R5
4261 013656 000746              BR     4$            ;CONTINUE COMPARE
4262 013660 005737 001276      7$:  TST    ZROIND   ;ANY MISCOMPARISONS NOT ZEROS ?
4263 013664 001743              BEQ    4$            ;BR IF NONE-ALL ERRORS=ZERO
4264 013666 004737 013770      JSR    PC,CMPRT     ;REPORT ERROR
4265 013672 000740              BR     4$            ;CONTINUE COMPARING
4266 013674 005737 001312      8$:  TST    CMCNT   ;AT END OF BUFFER
4267 013700 003430              BLE    CMPRX        ;BR IF AT END
4268 013702 126027 000024 000005  CMPB   $CODE(R0),#5 ;SEE IF READ HEADER & DATA
4269 013710 001220              BNE    CMSTR        ;BR IF NOT
4270 013712 105237 001316              INCB  CMSEC        ;INCREMENT SECTOR
4271 013716 123727 001316 000026  CMPB   CMSEC,#22.   ;SECTOR GREATER THAN MAX ?
4272 013724 103612              BLO   CMSTR        ;BR IF NOT GREATER THAN MAX
4273 013726 105037 001316              CLRB  CMSEC        ;CLEAR SECTOR ADDRESS
4274 013732 105237 001317              INCB  CMTRK        ;INCREMENT TRACK
4275 013736 123727 001317 000023  CMPB   CMTRK,#19.   ;TRACK GREATER THAN MAX ?
4276 013744 103602              BLO   CMSTR        ;BR IF NOT GREATER
4277 013746 105037 001317              CLRB  CMTRK        ;RESET TRACK ADDRESS
4278 013752 005237 001314              INC   CMCYL        ;INCREMENT CYLINDER ADDRESS
4279 013756 000137 013352              JMP   CMSTR        ;CONTINUE WITH COMPARE
4280 013762 004737 014244      CMPRX: JSR   PC,ENDCMP ;PRINT LAST LINE IF ERRORS
4281 013766 000207              RTS    PC
4282
4283      ;TYPE DATA COMPARE ERRORS
4284
4285 013770 005737 001302      CMPRT: TST    SAVER1   ;PRINT SAVED VALUES ?
4286 013774 001010              BNE   2$            ;BR IF NOT
4287 013776 105737 001300      TSTB   FRSTER   ;FIRST ERROR?
4288 014002 100402              BMI   1$            ;BR IF NOT
4289 014004 004737 014064      JSR    PC,4$        ;PRINT INITIAL MESSAGE INFO
4290 014010 004737 014146      1$:  JSR    PC,8$        ;PRINT REMAINDER OF MESSAGE
4291 014014 000422              BR     3$            ;EXIT
4292 014016
4293 014016 010146              2$:  MOV    R1,-(SP)    ;:PUSH R1 ON STACK
4294 014020 010546              MOV    R5,-(SP)    ;:PUSH R5 ON STACK
4295 014022 013701 001302      MOV    SAVER1,R1   ;:DISPLAY SAVED R1
4296 014026 013705 001304      MOV    SAVER5,R5   ;:DISPLAY SAVED R5
4297 014032 004737 014064      JSR    PC,4$        ;:PRINT INITIAL MESSAGE INFO
4298 014036 004737 014146      JSR    PC,8$        ;:PRINT SAVED VALUES
4299 014042 005037 001302      CLR    SAVER1      ;:CLEAR SAVED REGISTER INDICATORS
4300 014046 005037 001304      CLR    SAVER5      ;:CLEAR THE OTHER ONE
4301 014052 012605              MOV    (SP)+,R5    ;:POP STACK INTO R5
4302 014054 012601              MOV    (SP)+,R1    ;:POP STACK INTO R1
4303 014056 004737 014146      JSR    PC,8$        ;:PRINT REMAINDER OF MESSAGE
4304 014062 000207              3$:  RTS    PC         ;:RETURN
4305 014064 105737 001300      4$:  TSTB   FRSTER   ;:FIRST ERROR ?
4306 014070 100425              BMI   7$            ;:BR IF NOT
4307 014072 001013              BNE   5$            ;:BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
4308 014074 004737 020266      JSR    PC,LINE1    ;:PRINT LINE 1 OF ERROR MESSAGE
4309 014100 104414 047077      DISPLY ,EM42       ;:DATA COMPARE ERROR
```



```

4310 014104 004737 020332      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4311 014110 004737 020746      JSR    PC,LINE3A     ;PRINT LINE 3A OF ERROR MESSAGE
4312 014114 004737 021414      JSR    PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
4313 014120 000404              BR     6$            ;GO TO TYPE HEADER
4314 014122 104414 051533      5$:   DISPLY ,LIN9B   ;HEADER MESSAGE OF PROCESSING 'DCK' ERROR
4315 014126 104414 001165              DISPLY ,%CRLF       ;CR-LF
4316 014132 104414 051562      6$:   DISPLY ,LIN9H   ;DISPLAY HEADER
4317 014136 012737 177777      001300 MOV    #-1,FRSTER   ;SET ERROR FLAG
4318 014144 000207              RTS     PC           ;RETURN
4319 014146 005737 001310      8$:   TST    LIMIT    ;TIMEOUT LIMIT REACHED ?
4320 014152 001403              BEQ    9$            ;BR IF IT HAS
4321 014154 005337 001310              DEC    LIMIT        ;DECREMENT LIMIT COUNTER
4322 014160 001005              BNE    10$          ;BR IF NOT AT LIMIT
4323 014162 032777 000200      164750 9$:   BIT    #SW07,%SWR  ;PRINT ALL DATA COMPARE ERRORS ?
4324 014170 001001              BNE    10$          ;BR IF YES
4325 014172 000207              RTS     PC           ;RETURN
4326 014174 010146      10$:   MOV    R1,-(SP)   ;BUFFER ADDRESS
4327 014176 162716 000002      SUB    #2,(SP)     ;ADJUST ADDRESS
4328 014202 004737 022330      JSR    PC,LINOC7   ;TYPE IT
4329 014206 104414 052323      DISPLY ,LINSR      ;2 SPACES
4330 014212 016546 177776      MOV    -2(R5),-(SP) ;PUT GOOD DATA ON THE STACK
4331 014216 004737 022330      JSR    PC,LINOC7   ;TYPE IT
4332 014222 104414 052323      DISPLY ,LINSR      ;2 SPACES
4333 014226 016146 177776      MOV    -2(R1),-(SP) ;BAD DATA
4334 014232 004737 022330      JSR    PC,LINOC7   ;TYPE IT
4335 014236 104414 001165      DISPLY ,%CRLF     ;CR-LF
4336 014242 000207              RTS     PC           ;RETURN
4337
4338      ;LAST LINE OF COMPARE ERROR REPORTING
4339
4340 014244 105737 001301      ENDCMP: TSTB   FRSTER+1 ;ANY COMPARE ERRORS FOUND ?
4341 014250 001417              BEQ    2$            ;BR IF NOT
4342 014252 005737 001306              TST   ERCTR         ;SEE HOW MANY ERRORS
4343 014256 001410              BEQ    1$            ;BR IF ONLY CAN'T MATCH PATTERN
4344 014260 104414 051660              DISPLY ,LIN9E      ;'NUMBER OF ERRORS='
4345 014264 013746 001306              MOV    ERCTR,-(SP)  ;NUMBER OF ERRORS
4346 014270 004737 022362              JSR    PC,LINDEC   ;TYPE IT
4347 014274 104414 001165              DISPLY ,%CRLF     ;CR-LF
4348 014300 004737 023512      1$:   JSR    PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4349 014304 004737 022050              JSR    PC,LINE7    ;PRINT LINE 7 OF ERROR MESSAGE
4350 014310 000207      2$:   RTS     PC           ;RETURN
4351
4352      ;ROUTINE TO MATCH THE DATA WITH A PATTERN
4353      ;CALL:
4354      ;
4355      ;   MOV    #BUFFER,R1 ;BUFFER ADDRESS
4356      ;   JSR    PC,MATCH
4357      ;   RETURN1 ;PATTERN ADDRESS IN R4
4358      ;   RETURN2 ;COULDN'T MATCH PATTERN
4359
4360 014312 010146      MATCH: MOV    R1,-(SP)   ;SAVE R1 ON THE STACK
4361 014314 012704 000044      MOV    #4,R4        ;PATTERN TABLE INDEX
4362 014320 011601      1$:   MOV    (SP),R1    ;RELOAD R1
4363 014322 162704 000002      SUB    #2,R4        ;DECREMENT INDEX
4364 014326 016405 002762      MOV    STNDAT(R4),R5 ;ADDRESS OF PATTERN ADDRESS
4365 014332 001411      BEQ    3$            ;BR IF ALL PATTERNS CHECKED AND NO MATCH

```



```
4366  
4367 014334 012703 000004  
4368 014340 022125 2$: MOV #4,R3 ;FOUND  
4369 014342 001366 CMP (R1)+,(R5)+ ;NUMBER OF LOCATIONS TO CHECK  
4370 014344 005303 BNE 1$ ;COMPARE THE BUFFER AGAINST THE PATTERN  
4371 014346 001374 DEC R3 ;BR IF NOT EQUAL, TRY NEXT PATTERN  
4372 014350 062704 002762 BNE 2$ ;FINISHED CHECKING?  
4373 014354 000403 ADD #STNDAT,R4 ;BR IF NOT FINISHED  
4374 014356 062766 000002 000002 3$: BR 4$ ;MAKE PATTERN ADDRESS ABSOLUTE  
4375 014364 012601 4$: ADD #2,2(SP) ;EXIT  
4376 014366 000207 MOV (SP)+,R1 ;INCREMENT RETURN ADDRESS  
4377 RTS PC ;RESTORE R1  
4378 ;USE ECC TO CORRECT THE DATA ERROR ;RETURN  
4379  
4380 014370 016037 000240 001322 ECC: MOV $RPBA(RO),ECSEC ;ADDRESS OF LAST LOCN XFERED  
4381 014376 016046 000236 MOV $RPWC(RO),-(SP) ;ACT WORDS XFERED (2'S COMP)  
4382 014402 066016 000020 ADD $WRDL(RO),(SP) ;ADD WORDS REQUESTED  
4383 014406 005046 CLR -(SP) ;CLEAR NEXT STACK LOCN  
4384 014410 016046 000022 MOV $SSEC(RO),-(SP) ;SECTOR SIZE  
4385 014414 004737 027152 JSR PC,LINKDV ;DIVIDE WORDS XFERED BY SECTOR SIZE  
4386 014420 005716 TST (SP) ;PARTIAL SECTOR XFERED ?  
4387 014422 001413 BEQ 1$ ;BR IF NOT  
4388 014424 006316 ASL (SP) ;CONVERT INTO NUMBER OF BYTES  
4389 014426 161637 001322 SUB (SP),ECSEC ;SUBTRACT SECTOR RESIDUE  
4390 014432 126027 000024 000005 CMPB $CODE(RO),#5 ;WAS OP READ HEAD & DATA  
4391 014440 001007 BNE 2$ ;BR IF NOT  
4392 014442 062737 000010 001322 ADD #8.,ECSEC ;ADD HEADER SIZE (IN BYTES) BACK IN  
4393 014450 000403 BR 2$ ;GO ADJUST THE STACK POINTER  
4394 014452 162737 001000 001322 1$: SUB #1000,ECSEC ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)  
4395 014460 062706 000004 2$: ADD #4,SP ;ADJUST THE STACK POINTER  
4396 014464 016037 000300 001320 MOV $RPEC1(RO),ECBIT ;ECC POSITION COUNT  
4397 014472 005337 001320 DEC ECBIT ;ADJUST THE POSITION COUNT  
4398 014476 013737 001320 001330 MOV ECBIT,ECWRD ;LOAD THE WORD COUNT LOCATION  
4399 014504 042737 177760 001320 BIC #^C17,ECBIT ;SAVE THE BIT OFFSET COUNT  
4400 014512 042737 000017 001330 BIC #17,ECWRD ;CLEAR THE BIT OFFSET  
4401 014520 006237 001330 ASR ECWRD ;CHANGE TO BYTE COUNT  
4402 014524 006237 001330 ASR ECWRD ;CHANGE TO BYTE COUNT  
4403 014530 006237 001330 ASR ECWRD ;CHANGE TO BYTE COUNT  
4404 014534 104414 051737 DISPLY ,LIN10A ;'ERROR BURST BEGINS AT '  
4405 014540 013746 001330 MOV ECWRD,-(SP) ;PUT THE WORD COUNT ON THE STACK  
4406 014544 006216 ASR (SP) ;CONVERT TO WORD COUNT FOR MESSAGE  
4407 014546 004737 030066 JSR PC,$SB2D ;CONVERT THE WORD COUNT  
4408 014552 004737 027466 JSR PC,$SUPRS ;PRINT IT  
4409 014556 104414 051773 DISPLY ,LIN10B ;' IN DATA FIELD OF ERROR SECTOR'  
4410 014562 063737 001322 001330 ADD ECSEC,ECWRD ;FIND THE BEGINNING OF THE ERROR BURST  
4411 014570 026037 000240 001330 CMP $RPBA(RO),ECWRD ;SEE IF BURST WAS IN DATA READ  
4412 014576 101002 BHI .+6 ;BR IF IN DATA READ  
4413 014600 000137 015116 JMP ECC2 ;NOT IN DATA READ - REPORT IT  
4414 014604 016037 000302 001324 MOV $RPEC2(RO),ECMSK0 ;GET THE ERROR MASK  
4415 014612 005037 001326 CLR ECMSK1 ;CLEAR THE UPPER MASK WORD  
4416 014616 005737 001320 3$: TST ECBIT ;BIT OFFSET EQUAL ZERO  
4417 014622 001407 BEQ 4$ ;BR IF IT IS  
4418 014624 005337 001320 DEC ECBIT ;DECREMENT THE BIT OFFSET COUNT  
4419 014630 006337 001324 ASL ECMSK0 ;SHIFT THE ERROR MASK  
4420 014634 006137 001326 ROL ECMSK1 ;SHIFT THE LOWER INTO THE UPPER  
4421 014640 000766 BR 3$ ;CONTINUE THE SHIFT
```



```

4422 014642 017737 164462 001334 4$: MOV @ECWRD,ECBADO ;SAVE THE INCORRECT WORD
4423 014650 005037 001336 CLR ECWRD1 ;CLEAR SECOND INCORRECT WORD ADDRESS
4424 014654 013746 001324 MOV ECMSK0,-(SP) ;PUT LOWER MASK ON STACK
4425 014660 047716 164444 BIC @ECWRD,(SP) ;CLEAR ERRONEOUS ONE BITS FROM MASK
4426 014664 043777 001324 164436 BIC ECMSK0,@ECWRD ;CLEAR ERRONEOUS ONE BITS FROM BAD WORD
4427 014672 052677 164432 BIS (SP)+,@ECWRD ;SET DROPPED BITS
4428 014676 005737 001326 TST ECMSK1 ;DOES BURST GO INTO NEXT WORD ?
4429 014702 001431 BEQ ECC1 ;BR IF BURST ONLY IN ONE WORD
4430 014704 013737 001330 001336 MOV ECWRD,ECWRD1 ;DUPLICATE ADDRESS
4431 014712 062737 000002 001336 ADD #2,ECWRD1 ;INCREMENT ERROR ADDRESS
4432 014720 026037 000240 001336 CMP $RPBA(R0),ECWRD1 ;IS NEXT WORD IN THE BUFFER
4433 014726 101003 BHI 5$ ;BR IF IT IS
4434 014730 005037 001336 CLR ECWRD1 ;CLEAR 2ND WORD ADDRESS
4435 014734 000414 BR ECC1 ;PRINT WORD CORRECTED
4436 014736 017737 164374 001342 5$: MOV @ECWRD1,ECBAD1 ;SAVE THE SECOND BAD WORD
4437 014744 013746 001326 MOV ECMSK1,-(SP) ;PUT THE UPPER MASK ON THE STACK
4438 014750 047716 164362 BIC @ECWRD1,(SP) ;CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
4439 014754 043777 001326 164354 BIC ECMSK1,@ECWRD1 ;CLEAR ERRONECUS ONE BITS FROM DATA WORD
4440 014762 052677 164350 BIS (SP)+,@ECWRD1 ;SET DROPPED BITS
4441 014766 104414 052141 ECC1: DISPLY ,LIN10H ;HEADER
4442 014772 013746 001330 MOV ECWRD,-(SP) ;PUT ECWRD ON THE STACK
4443 014776 004737 022330 JSR PC,LIN0CT ;TYPE ECWRD
4444 015002 104414 052323 DISPLY ,LINSF ;SPACES
4445 015006 013746 001334 MOV ECBADO,-(SP) ;PUT ECBADO ON THE STACK
4446 015012 004737 022330 JSR PC,LIN0CT ;TYPE ECBADO
4447 015016 104414 052323 DISPLY ,LINSF ;SPACES
4448 015022 017746 164302 MOV @ECWRD,-(SP) ;PUT @ECWRD ON THE STACK
4449 015026 004737 022330 JSR PC,LIN0CT ;TYPE @ECWRD
4450 015032 104414 052323 DISPLY ,LINSF ;SPACES
4451 015036 005737 001336 TST ECWRD1 ;PRINT THE NEXT WORD ?
4452 015042 001427 BEQ ECCX ;BR IF NOT
4453 015044 104414 001165 DISPLY ,$CRLF ;CR-LF
4454 015050 013746 001336 MOV ECWRD1,-(SP) ;PUT ECWRD1 ON THE STACK
4455 015054 004737 022330 JSR PC,LIN0CT ;TYPE ECWRD1
4456 015060 104414 052323 DISPLY ,LINSF ;SPACES
4457 015064 013746 001342 MOV ECBAD1,-(SP) ;PUT ECBAD1 ON THE STACK
4458 015070 004737 022330 JSR PC,LIN0CT ;TYPE ECBAD1
4459 015074 104414 052323 DISPLY ,LINSF ;SPACES
4460 015100 017746 164232 MOV @ECWRD1,-(SP) ;PUT @ECWRD1 ON THE STACK
4461 015104 004737 022330 JSR PC,LIN0CT ;TYPE @ECWRD1
4462 015110 104414 052323 DISPLY ,LINSF ;SPACES
4463 015114 000402 BR ECCX ;EXIT
4464 015116 104414 052034 ECC2: DISPLY ,LIN10C ;ERROR BURST WAS NOT TRANSFERED TO MEMGRY
4465 015122 104414 001165 ECCX: DISPLY ,$CRLF ;CR-LF
4466 015126 000207 RTS PC ;RETURN
4467
4468 ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
4469
4470 015130 032777 000010 164002 PRTBAD: BIT #SW3,@SWR ;PRINT THE BAD SECTOR ?
4471 015136 001460 BEQ 6$ ;BR IF NOT
4472 015140 016001 000240 MOV $RPBA(R0),R1 ;PUT THE END ADDRESS INTO R1
4473 015144 016046 000020 MOV $WRDL(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
4474 015150 066016 000236 ADD $RPWC(R0),(SP) ;SUBTRACT THE WORDS NOT TRANSFERED
4475 015154 005046 CLR -(SP) ;MAKE THE UPPER DIVIDEND 0
4476 015156 016046 000022 MOV $SSEC(R0),-(SP) ;DIVDE THE WORDS TRANSFERED BY THE SECTOR SIZE
4477 015162 004737 027152 JSR PC,LINKDV ;DIVIDE

```



```
4478 015166 005716          TST      (SP)          ;REMAINDER = 0 ?
4479 015170 001403          BEQ      1$            ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
4480 015172 006316          ASL      (SP)          ;CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
4481 015174 161601          SUB      (SP),R1       ;SUBTRACT IT FROM THE END ADDRESS
4482 015176 000410          BR       2$            ;FINISH THE SIZING
4483 015200 162701 001000    1$:      SUB      #1000,R1     ;SUBTRACT FULL SECTOR SIZE FROM END ADDR
4484 015204 126027 000024 000005    CMPB     $CODE(R0),#5   ;WAS OPERATION READ HEADER & DATA ?
4485 015212 001002          BNE      2$            ;BR IF NOT
4486 015214 162701 000010    SUB      #10,R1        ;SUBTRACT HEADER SIZE FROM ADDR
4487 015220 062706 000004    2$:      ADD      #4,SP      ;RESTORE THE STACK POINTER
4488 015224 104414 052226    DISPLY   ,LIN11H       ;PRINT THE HEADER
4489 015230 012702 000007    3$:      MOV      #7,R2      ;R2 CONTAINS THE WORDS/LINE COUNT
4490 015234 010146          MOV      R1,-(SP)      ;PUT THE ADDRESS ON THE STACK
4491 015236 004737 022330    JSR      PC,LIN0CT     ;TYPE THE ADDRESS
4492 015242 020160 000240    4$:      CMP      R1,$RPBA(R0) ;PRINTED ALL THE SECTOR ?
4493 015246 001412          BEQ      5$            ;BR IF ALL PRINTED
4494 015250 104414 052323    DISPLY   ,LINSR        ;SPACES
4495 015254 012146          MOV      (R1)+,-(SP)   ;PUT THE DATA ON THE STACK
4496 015256 004737 022330    JSR      PC,LIN0CT     ;TYPE THE DATA
4497 015262 005302          DEC      R2            ;DECREMENT THE HORIZONTAL COUNT
4498 015264 001366          BNE      4$            ;BR IF NOT AT THE END OF THE LINE
4499 015266 104414 001165    DISPLY   ,$CRLF        ;CR-LF
4500 015272 000756          BR       3$            ;RESTORE THE WORDS/LINE COUNT
4501 015274 104414 001165    5$:      DISPLY   ,$CRLF        ;PRINT WHAT REMAINS IN THE BUFFER
4502 015300 000207    6$:      RTS      PC          ;RETURN
4503
4504          ;ROUTINE TO DO AN RTC - DRIVE SELECTED IN R0
4505          ;CALL:
4506          ;      MOV      #DPB,R0          ;DPB ADDRESS
4507          ;      JSR      PC,RTNCTR
4508          ;      RETURN
4509
```



```
4510 015302 111037 045076      RTNCTR: MOVB   (R0),GENDPB   ;MOVE THE DRIVE # TO THE GENERAL DPB
4511 015306 112737 000117 045100  MOVB   #RTC,GENDPB+$COMND ;COMMAND CODE
4512 015314 004037 034266      1$:   JSR    RO,RP04        ;DRIVER ENTRANCE
4513 015320 045076              GENDPB ;DPB ADDRESS FOR ORDER
4514 015322 000774              BR     1$                  ;DRIVER DIDN'T ACCEPT ORDER
4515 015324 000207              RTS     PC                 ;RETURN
4516
4517 ;ROUTINE TO DO A RECALIBRATE - DRIVE SELECTED IN R0
4518 ;CALL:
4519 ;       MOV    #DPB,R0        ;DPB ADDRESS
4520 ;       JSR    PC,RECALT
4521 ;       RETURN
4522
4523 ;OR
4524 ;
4525 ;       MOV    #DPB,R0        ;DPB ADDRESS
4526 ;       MOVB   #DRIVE,GENDPB ;DRIVE ADDRESS
4527 ;       JSR    PC,RECALTO
4528 ;       RETURN
4529
4530 015326 111037 045076      RECALT: MOVB   (R0),GENDPB   ;MOVE THE DRIVE # TO THE GENERAL DPB
4531 015332 112737 000107 045100  RECALO: MOVB   #RECAL,GENDPB+$COMND ;REL CALIBRATE COMMAND
4532 015340 004037 034266      1$:   JSR    RO,RP04        ;DRIVER ENTRANCE
4533 015344 045076              GENDPB ;DPB ADDRESS FOR ORDER
4534 015346 000774              BR     1$                  ;DRIVER DIDN'T ACCEPT THE ORDER
4535 015350 005737 045114      2$:   TST    GENDPB+$STATUS ;SEE IF FINISHED
4536 015354 001775              BEQ    2$                  ;BR IF NOT FINISHED
4537 015356 000207              RTS     PC                 ;RETURN
4538
4539 ;OFFSET THE DRIVE IN R0 (OFFSET CODE PRELOADED INTO 'RPOF')
4540 ;CALL:
4541 ;       MOVB   #OFFSET,GENDPB+$FMT ;OFFSET CODE
4542 ;       MOV    #DPB,R0        ;DPB ADDRESS
4543 ;       JSR    PC,OFFST
4544 ;       RETURN
4545
4546 015360 111037 045076      OFFST: MOVB   (R0),GENDPB   ;DRIVE # TO GENERAL DPB
4547 015364 112737 000115 045100  MOVB   #OFFSET,GENDPB+$COMND ;COMMAND
4548 015372 004037 034266      1$:   JSR    RO,RP04        ;DRIVER ENTRANCE
4549 015376 045076              GENDPB ;DPB ADDRESS FOR ORDER
4550 015400 000774              BR     1$                  ;DRIVER DIDN'T ACCEPT ORDER
4551 015402 000207              RTS     PC
4552
4553 ;UTILITY READ HEADER ROUTINE
4554 ;CALL:
4555 ;       MOV    #DPB,R0        ;DPB ADDRESS
4556 ;       MOV    #SECTOR,-(SP)  ;SECTOR ADDRESS
4557 ;       MOV    #TRACK,-(SP)  ;TRACK ADDRESS
4558 ;       JSR    PC,READDR
4559 ;       RETURN
4560
4561 015404 116637 000002 045107  READHD: MOVB   2(SP),GENDPB+$TRK ;TRACK ADDRESS
4562 015412 116637 000004 045106  MOVB   4(SP),GENDPB+$SEC ;SECTOR ADDRESS
4563 015420 111037 045076              MOVB   (R0),GENDPB        ;DRIVE NUMBER
4564 015424 016037 000272 045110  MOV    $RPCC(R0),GENDPB+$CYL ;CYLINDER ADDRESS
4565 015432 112737 000173 045100  MOVB   #RDHD,GENDPB+$COMND ;COMMAND
```



```
4566 015440 004037 034266 1$: JSR RO,RP04 ;DRIVER ENTRANCE
4567 015444 045076 GENDPB ;DPB ADDRESS FOR ORDER
4568 015446 000774 BR 1$ ;DRIVER DIDN'T ACCEPT COMMAND
4569 015450 005737 045114 2$: TST GENDPB+$STATUS ;FINISHED?
4570 015454 001775 BEQ 2$ ;BR IF NOT
4571 015456 012666 000002 MOV (SP)+,2(SP) ;ADJUST STACK FOR RETURN
4572 015462 005726 TST (SP)+
4573 015464 000207 RTS PC ;RETURN
4574
4575 ;RETRY THE PRESENT OPERATION
4576 ;CALL:
4577 ;: MOV #COUNT,RETRY ;RETRY COUNT
4578 ;: JSR PC,$RETRY
4579 ;: RETURN1 ;RETRY UNSUCCESSFUL
4580 ;: RETURN2 ;SUCCESSFUL RETRY
4581 ;: ;NOTE: IF A DIFFERENT ERROR OCCURS DURING
4582 ;: ;RETRY, THE ROUTINE EXITS TO 'ERPRC1'
4583
4584 015466 004737 016456 $RETRY: JSR PC,GODRIV ;RE-START ORDER
4585 015472 005760 000016 1$: TST $STATUS(RO) ;ORDER FINISHED?
4586 015476 001775 BEQ 1$ ;BR IF NOT
4587 015500 100405 BMI 2$ ;BR IF ERROR
4588 015502 105237 001253 INCB RETRY+1 ;INCREMENT RETRY COUNT
4589 015506 062716 000002 ADD #2,(SP) ;INCREMENT RETURN
4590 015512 000425 BR 5$ ;GO TO EXIT
4591 015514 032760 000200 000016 2$: BIT #BIT7,$STATUS(RO) ;DID ORDER TERMINATE NORMALLY ?
4592 015522 001430 BEQ 7$ ;BR IF NOT
4593 015524 005737 001250 TST MASK ;IS ERROR MASK 0 ?
4594 015530 001004 BNE 3$ ;BR IF NOT
4595 015532 005760 000250 TST $RPER1(RO) ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
4596 015536 001014 BNE 6$ ;BR IF NOT
4597 015540 000404 BR 4$ ;CONTINUE RETRY
4598 015542 033760 001250 000250 3$: BIT MASK,$RPER1(RO) ;SAME ERROR?
4599 015550 001407 BEQ 6$ ;BR IF NOT
4600 015552 105237 001253 4$: INCB RETRY+1 ;INCREMENT RETRY COUNT
4601 015556 123737 001252 001253 CMPB RETRY,RETRY+1 ;DONE ?
4602 015564 001340 BNE $RETRY ;BR IF NOT DONE
4603 015566 000207 5$: RTS PC ;RETURN
4604 015570 004737 022316 6$: JSR PC,LINE8 ;REPORT DIFFERENT ERROR
4605 015574 004737 022050 JSR PC,LINE7 ;PRINT LINE 7
4606 015600 005726 TST (SP)+ ;ADJUST STACK POINTER FOR DIRECT RETURN
4607 015602 000207 RTS PC ;RETURN
4608 015604 104414 051476 7$: DISPLY ,LIN8M ;'DIFFERENT ERROR DURING RETRY'
4609 015610 000137 006732 JMP ERPRC1 ;REPORT THE ERROR
4610
4611 ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
4612 ;CALL:
4613 ;: MOV #DPB,RO ;DPB ADDRESS
4614 ;: JSR PC,STATIS
4615 ;: RETURN
4616
4617 015614 032760 000300 000016 STATIS: BIT #BIT07:BIT06,$STATUS(RO) ;CHECK FOR DATA TERMINATION
4618 015622 001454 BEQ 3$ ;BR IF NOT DATA TERMINATION
4619 015624 016037 000240 015756 MOV $RPBA(RO),FACTOR ;STORE THE FINAL BUFFER ADDRESS
4620 015632 166037 000006 015756 SUB $BUF(RO),FACTOR ;SUBTRACT THE INITIAL ADDRESS
4621 015640 001434 BEQ 2$ ;BR IF NO DATA TRANSFER
```



```

4622 015642 006237 015756 ASR FACTOR ; CONVERT TO A WORD COUNT
4623 015646 063760 015756 000046 ADD FACTOR,$TRANS(RO) ; UPDATE WORD COUNT
4624 015654 005560 000050 ADC $TRANS+2(RO) ; ADD ANY CARRY
4625 015660 132760 000002 000024 BITB #BIT01,$CODE(RO) ; SEE IF ORDER READ OR WRITE
4626 015666 001021 BNE 2$ ; BRANCH IF ORDER WRITE
4627 015670 005737 001424 TST AUTOCK ; AUTO WRITE CHECKS BEING PERFORMED
4628 015674 001411 BEQ 1$ ; BR IF NOT
4629 015676 126027 000024 000001 CMPB $CODE(RO),#1 ; PRESENT OPERATION AN AUTOMATIC WRITE CHECK ?
4630 015704 101005 BHI 1$ ; BR IF NOT
4631 015706 066060 000020 000046 ADD $WRDL(RO),$TRANS(RO) ; ADD WORDS WRITTEN
4632 015714 005560 000050 ADC $TRANS+2(RO) ; ADD A CARRY
4633 015720 063760 015756 000052 1$: ADD FACTOR,$READ(RO) ; UPDATE THE READ WORD COUNT
4634 015726 005560 000054 ADC $READ+2(RO) ; ADD ANY CARRY
4635 015732 026060 000012 000270 2$: CMP $CYL(RO),$RPCA(RO) ; DID MID-TRANSFER SEEK OCCUR
4636 015740 001405 BEQ 3$ ; BR IF NOT
4637 015742 062760 000001 000042 ADD #1,$POSIT(RO) ; INCREMENT SEEK COUNT
4638 015750 005560 000044 ADC $POSIT+2(RO) ; ADD CARRY TO UPPER WORD
4639 015754 000207 3$: RTS PC
4640
4641 015756 000000 FACTOR: .WORD 0 ; USED FOR WORDS TRANSFERED
4642
4643 ;ROUTINE TO GET A BUFFER
4644 ;CALL:
4645 ; MOV #DPB,RO ;DPB ADDRESS
4646 ; CLR -(SP) ;CLEAR THE STACK
4647 ; JSR PC,GETBUF
4648 ; RETURN ;BUFFER ADDRESS WILL BE ON THE STACK
4649 ; ;STACK WILL BE ZERO IF NO BUFFER AVAILABLE
4650
4651 015760 010146 GETBUF: MOV R1,-(SP) ;SAVE R1
4652 015762 010246 MOV R2,-(SP) ;SAVE R2
4653 015764 010346 MOV R3,-(SP) ;SAVE R3
4654 015766 013702 001616 MOV BUFTBL,R2 ;NUMBER OF SEPARATE BUFFERS
4655 015772 001444 BEQ 6$ ;BR IF NONE AVAILABLE
4656 015774 012701 001620 MOV #BUFTBL+2,R1 ;FIRST ADDRESS OF ALLOCATION TABLE
4657 016000 026061 000020 000002 1$: CMP $WRDL(RO),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
4658 016006 101405 BLOS 3$ ;BRANCH IF IT IS
4659 016010 005302 DEC R2 ;DECREMENT TABLE COUNT
4660 016012 001434 BEQ 6$ ;BR IF THROUGH TABLE
4661 016014 062701 000004 ADD #4,R1 ;INCREMENT TABLE POINTER
4662 016020 000767 BR 1$ ;CONTINUE LOOKING
4663 016022 011166 000010 3$: MOV (R1),10(SP) ;BUFFER ADDRESS TO STACK
4664 016026 166061 000020 000002 SUB $WRDL(RO),2(R1) ;ADJUST BUFFER SIZE
4665 016034 001407 BEQ 4$ ;BR IF DIFFERENCE IS ZERO
4666 016036 006360 000020 ASL $WRDL(RO) ;CONVERT # WORDS TO BYTES
4667 016042 066011 000020 ADD $WRDL(RO),(R1) ;MAKE NEW STARTING ADDRESS
4668 016046 006260 000020 ASR $WRDL(RO) ;RETURN # BYTES TO WORDS
4669 016052 000414 BR 6$ ;RETURN
4670 016054 005337 001616 4$: DEC BUFTBL ;DECREMENT ENTRIES COUNT
4671 016060 001411 BEQ 6$ ;BR IF ALLOCATION TABLE EMPTY
4672 016062 005302 DEC R2 ;DECREMENT TABLE COUNT
4673 016064 001407 BEQ 6$ ;BR IF ITEM WERE LAST ENTRY
4674 016066 010103 MOV R1,R3 ;MOVE TABLE POINTER
4675 016070 062703 000004 ADD #4,R3 ;POINT TO NEXT ENTRY
4676 016074 012321 5$: MOV (R3)+,(R1)+ ;MOVE ITEMS
4677 016076 012321 MOV (R3)+,(R1)+

```



```

4678 016100 005302          DEC      R2          ;DECREMENT TABLE COUNT
4679 016102 001374          BNE      5$          ;CONTINUE IF NOT AT END OF TABLE
4680 016104 012603      6$:  MOV      (SP)+,R3      ;RESTORE R3
4681 016106 012602          MOV      (SP)+,R2      ;RESTORE R2
4682 016110 012601          MOV      (SP)+,R1      ;RESTORE R1
4683 016112 000207          RTS       PC          ;RETURN
4684
4685
4686          ;ROUTINE TO PUT BUFFER BACK IN TABLE
4687          ;CALL:
4688          ;      MOV      #DPB,R0          ;DPB ADDRESS
4689          ;      JSR      PC,RELBUF
4690          ;      RETURN
4691
4692 016114 010146      RELBUF: MOV      R1,-(SP)      ;SAVE R1
4693 016116 012701 001620      MOV      #BUFTBL+2,R1  ;BEGINNING OF TABLE
4694 016122 013702 001616      MOV      BUFTBL,R2     ;ENTRY COUNT
4695 016126 001424          BEQ      2$           ;BR IF EMPTY TABLE
4696 016130 016003 000020      MOV      $WRDL(R0),R3  ;TRIAL ADDRESS
4697 016134 006303          ASL      R3           ;CHANGE TO BYTE COUNT
4698 016136 066003 000006      ADD      $BUF(R0),R3   ;ADDRESS OF HIGHER ADJACENT BLOCK
4699 016142 021103      1$:  CMP      (R1),R3      ;UPPER ADJACENT BLOCK
4700 016144 001424          BEQ      4$           ;BR IF YES
4701 016146 062701 000004      ADD      #4,R1        ;INCREMENT POINTER
4702 016152 005302          DEC      R2           ;DECREMENT ENTRY COUNT
4703 016154 001372          BNE      1$           ;CONTINUE SEARCHING
4704 016156 016011 000006      MOV      $BUF(R0),(R1) ;PUT THE BUFFER BLOCK INTO THE TABLE
4705 016162 016061 000020 000002      MOV      $WRDL(R0),2(R1) ;BLOCK SIZE
4706 016170 005237 001616      INC      BUFTBL       ;INCREMENT ENTRY COUNT
4707 016174 005202          INC      R2           ;INCREMENT R2 FOR USE LATER
4708 016176 000414          BR       5$           ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
4709 016200 016021 000006      2$:  MOV      $BUF(R0),(R1)+ ;BLOCK ADDRESS TO TABLE
4710 016204 016021 000020      MOV      $WRDL(R0),(R1)+ ;SIZE TO TABLE
4711 016210 005237 001616      INC      BUFTBL       ;INCREMENT ENTRY COUNT
4712 016214 000443          BR       10$          ;EXIT
4713 016216 016011 000006      4$:  MOV      $BUF(R0),(R1) ;RELEASED BUFFER IS LOWER ADJACENT
4714 016222 066061 000020 000002      ADD      $WRDL(R0),2(R1) ;INCREMENTED SIZE
4715 016230 010246      5$:  MOV      R2,-(SP)     ;SAVE R2
4716 016232 013702 001616      MOV      BUFTBL,R2     ;ENTRY COUNT
4717 016236 012705 001620      MOV      #BUFTBL+2,R5  ;BEGINNING OF TABLE
4718 016242 016504 000002      6$:  MOV      2(R5),R4     ;BLOCK SIZE (IN WORDS)
4719 016246 006304          ASL      R4           ;CHANGE TO BYTE COUNT
4720 016250 061504          ADD      (R5),R4      ;ADD BLOCK BEGINNING ADDRESS
4721 016252 020411          CMP      R4,(R1)      ;R1 STILL POINTS TO INSERTED ENTRY
4722 016254 001406          BEQ      8$           ;LOWER ADJACENT IN TABLE
4723 016256 062705 000004      ADD      #4,R5        ;INCREMENT POINTER
4724 016262 005302          DEC      R2           ;DECREMENT ENTRY COUNT
4725 016264 001366          BNE      6$           ;CONTINUE LOOKING
4726 016266 005726          TST      (SP)+        ;RESTORE STACK POINTER
4727 016270 000415          BR       10$          ;END
4728 016272 012602      8$:  MOV      (SP)+,R2     ;RESTORE R2
4729 016274 066165 000002 000002      ADD      2(R1),2(R5)   ;INCREMENT LOWER BLOCK LENGTH
4730 016302 005337 001616      DEC      BUFTBL       ;DECREMENT ENTRY COUNT
4731 016306 010105          MOV      R1,R5        ;GET READY TO COMPRESS
4732 016310 062705 000004      ADD      #4,R5        ;INCREMENT TO NEXT ENTRY
4733 016314 012521      9$:  MOV      (R5)+,(R1)+  ;COMPRESS TABLE

```



```

4734 016316 012521      MOV      (R5)+,(R1)+      ;MOVE SIZE FIELD DOWN
4735 016320 005302      DEC      R2                ;DECREMENT ENTRY COUNT
4736 016322 001374      BNE     9$                ;BR IF NOT FINISHED
4737 016324 012601      10$:   MOV      (SP)+,R1    ;RESTORE R1
4738 016326 000207      RTS     PC                ;RETURN
4739
4740
4741      ;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK ORDER)
4742      ;CALL:
4743      ;      MOV      #DPB,R0      ;DPB ADDRESS
4744      ;      MOV      #BUFADR,$BUF(R0) ;LOAD BUFFER ADDRESS INTO THE DPB
4745      ;      MOVB   #PATTERN,$PATT(R0) ;PATTERN CODE
4746      ;      JSR     PC,FILBUF
4747      ;
4748
4749 016330 104412      FILBUF: SAVREG           ;SAVE THE REGISTERS
4750 016332 132760 000004 000024  BITB   #BIT02,$CODE(R0) ;SEE IF READ ORDER
4751 016340 001044      BNE     4$                ;BR IF READ
4752 016342 016001 000006      1$:   MOV      $BUF(R0),R1    ;BUFFER ADDRESS
4753 016346 016002 000020      MOV      $WRDL(R0),R2     ;POSITIVE WORD COUNT
4754 016352 132760 000001 000024  BITB   #BIT00,$CODE(R0) ;SEE IF WRITE HEADER TYPE ORDER
4755 016360 001413      BEQ     2$                ;BR IF NOT
4756 016362 016011 000012      MOV      $CYL(R0),(R1)    ;CYLINDER ADDRESS
4757 016366 052721 010000      BIS     #BIT12,(R1)+     ;SET FMT22 BIT
4758 016372 016021 000010      MOV      $SEC(R0),(R1)+  ;MOVE SECTOR & TRACK
4759 016376 005021      CLR     (R1)+            ;CLEAR FIRST KEY WORD
4760 016400 005021      CLR     (R1)+            ;CLEAR THE SECOND
4761 016402 162702 000004      SUB     #4,R2            ;ADJUST THE WORD COUNT
4762 016406 003421      BLE     4$                ;BR IF END OF PATTERN
4763 016410 005004      2$:   CLR     R4                ;CLEAR R4
4764 016412 116004 000030      MOVB   $PATT(R0),R4     ;RELATIVE PATTERN ADDRESS
4765 016416 016405 002762      MOV     STNDAT(R4),R5    ;PATTERN ADDRESS
4766 016422 012703 000020      MOV     #20,R3          ;PATTERN COUNT
4767 016426 012521      3$:   MOV      (R5)+,(R1)+    ;MOVE THE PATTERN INTO THE BUFFER
4768 016430 005302      DEC     R2                ;DECREMENT THE WORD COUNT
4769 016432 001407      BEQ     4$                ;BR IF DONE (WORD COUNT = 0)
4770 016434 005303      DEC     R3                ;DECREMENT THE PATTERN COUNT
4771 016436 001373      BNE     3$                ;BR IF MORE PATTERN
4772 016440 012703 000020      MOV     #20,R3          ;RESTORE PATTERN COUNT
4773 016444 016405 002762      MOV     STNDAT(R4),R5    ;RESTORE THE ADDRESS
4774 016450 000766      BR     3$                ;CONTINUE DISTRIBUTING THE PATTERN
4775 016452 104413      4$:   RESREG           ;RESTORE THE REGISTERS
4776 016454 000207      RTS     PC                ;RETURN
4777
4778      ;START THE ORDER FOR THE DPB IN R0
4779      ;CALL:
4780      ;      MOV      #DPB,R0      ;DPB ADDRESS
4781      ;      JSR     PC,GODRIV
4782      ;
4783
4784 016456 010046      GODRIV: MOV     R0,-(SP)      ;SAVE R0
4785 016460 010037 016470      MOV     R0,2$           ;CURRENT DPB ADDRESS
4786 016464 004037 034266      1$:   JSR     R0,RPO4      ;CALL THE DRIVE HANDLER
4787 016470 000000      2$:   .WORD   0           ;DRIVE BLOCK ADDRESS GOES HERE
4788 016472 000000      HALT
4789 016474 012600      MOV     (SP)+,R0       ;RESTORE R0

```



```

4790 016476 062760 000001 000036      ADD      #1,$OPERC(R0)      ;INCREMENT THE OPERATION COUNT
4791 016504 005560 000040      ADC      $OPERC+2(R0)
4792 016510 026060 000034 000012      CMP      $PREVA+2(R0),$CYL(R0) ;DID ORDER REQUIRE A CYLINDER CHANGE
4793 016516 001405      BEQ      3$                ;BR IF NOT
4794 016520 062760 000001 000042      ADD      #1,$POSIT(R0)     ;INCREMENT SEEK COUNT
4795 016526 005560 000044      ADC      $POSIT+2(R0)      ;ADD ANY CARRY
4796 016532 000207      3$:     RTS      PC
4797
4798                                     ;GENERATE PARAMETERS FOR THE OPERATION
4799                                     ;CALL:
4800                                     ;     MOV      #DPB,R0      ;DPB ADDRESS
4801                                     ;     JSR      PC,SELPAR
4802                                     ;     RETURN
4803
4804 016534 004737 032462      SELPAR: JSR      PC,$RAND      ;CYCLE THE RANDOM NUMBER GENERATOR
4805 016540 032777 000001 162372      BIT      #SW0,@SWR        ;SEE IF SW0 SET
4806 016546 001012      BNE      2$                ;BR IF SET - READ ONLY
4807 016550 012705 000010      1$:     MOV      #10,R5      ;READ/WRITE SELECTION DIVISOR
4808 016554 004737 027124      JSR      PC,GETREM        ;GET SELECTION VALUE
4809 016560 020537 001422      CMP      R5,RATIO        ;DETERMINE IF READ OR WRITE
4810 016564 103003      BHS      2$                ;BR IF READ
4811 016566 004737 017256      JSR      PC,RANWRT        ;SELECT A WRITE ORDER
4812 016572 000406      BR       3$                ;CONTINUE WITH THE SELECTION
4813 016574 013705 032562      2$:     MOV      $LONUM,R5   ;SELECT READ OPERATION CODE
4814 016600 042705 177776      BIC      #^C1,R5         ;MASK OUT ALL BUT BIT 0
4815 016604 062705 000004      ADD      #4,R5           ;TABLE OFFSET FOR READ CODE
4816 016610 110560 000074      3$:     MOVB     R5,$NCODE(R0) ;ORDER SELECTION CODE TO CONTROL BLOCK
4817
4818                                     ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
4819
4820 016614 016005 000116      RANSEC: MOV      MAXSEC(R0),R5 ;GET MAXIMUM SECTOR ADDRESS
4821 016620 026005 000120      CMP      MINSEC(R0),R5   ;'MINSEC' AND 'MAXSEC' THE SAME ?
4822 016624 001417      BEQ      2$                ;BR IF THEY ARE
4823 016626 166005 000120      SUB      MINSEC(R0),R5   ;SUBTRACT MINIMUM SECTOR ADDRESS
4824 016632 100002      BPL      1$                ;BR IF MAX LARGER THAN MIN
4825 016634 062705 000026      ADD      #22.,R5        ;CORRECT THE NUMBER
4826 016640 005205      1$:     INC      R5          ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4827 016642 004737 027124      JSR      PC,GETREM        ;GET THE RANDOM AUGMENT
4828 016646 066005 000120      ADD      MINSEC(R0),R5   ;NEW ADDRESS
4829 016652 020527 000025      CMP      R5,#21.        ;IS VALUE TOO LARGE ?
4830 016656 101402      BLOS     2$                ;BR IF NOT
4831 016660 162705 000026      SUB      #22.,R5        ;CORRECT VALUE
4832 016664 110560 000076      2$:     MOVB     R5,$NSEC(R0) ;STORE SECTOR ADDRESS IN DPB
4833
4834                                     ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
4835
4836 016670 016005 000112      RANTRK: MOV      MAXTRK(R0),R5 ;GET MAXIMUM TRACK ADDRESS
4837 016674 026005 000114      CMP      MINTRK(R0),R5   ;'MINTRK' AND 'MAXTRK' THE SAME ?
4838 016700 001417      BEQ      2$                ;BR IF THEY ARE
4839 016702 166005 000114      SUB      MINTRK(R0),R5   ;SUBTRACT MINIMUM TRACK ADDRESS
4840 016706 100002      BPL      1$                ;BR IF MAX LARGER THAN MIN
4841 016710 062705 000023      ADD      #19.,R5        ;CORRECT THE NUMBER
4842 016714 005205      1$:     INC      R5          ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4843 016716 004737 027124      JSR      PC,GETREM        ;GET THE RANDOM AUGMENT
4844 016722 066005 000114      ADD      MINTRK(R0),R5   ;NEW TRACK ADDRESS
4845 016726 020527 000022      CMP      R5,#18.        ;IS VALUE TOO LARGE ?

```



```

4846 016732 101402          BLOS 2$          ;BR IF NOT
4847 016734 162705 000023  SUB   #19.,R5    ;CORRECT VALUE
4848 016740 110560 000077  2$:  MOVB  R5,$NTRK(RO) ;STORE TRACK ADDRESS IN DPB
4849
4850          ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'
4851
4852 016744 012737 000633 001350  MOV   #411.,CYLIMT ;ASSUME AN RP04/5
4853 016752 032760 000002 000262  BIT   #BIT01,$RPDT(RO) ;SEE IF RP06
4854 016760 001403          BEQ   RANCYL      ;BR IF NOT
4855 016762 012737 001457 001350  MOV   #815.,CYLIMT ;CHANGE CYLINDER LIMIT
4856 016770 016005 000106  RANCYL: MOV  MAXCYL(RO),R5 ;GET MAXIMUM CYLINDER ADDRESS
4857 016774 026005 000110  CMP   MINCYL(RO),R5 ;'MINCYL' AND 'MAXCYL' THE SAME ?
4858 017000 001417          BEQ   2$          ;BR IF THEY ARE
4859 017002 166005 000110  SUB   MINCYL(RO),R5 ;SUBTRACT MINIMUM CYLINDER ADDRESS
4860 017006 100002          BPL   1$          ;BR IF MAX LARGER THAN MIN
4861 017010 063705 001350  ADD   CYLIMT,R5    ;CORRECT THE NUMBER
4862 017014 005205 1$:  INC   R5          ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4863 017016 004737 027124  JSR   PC,GETREM    ;GET THE RANDOM AUGMENT
4864 017022 066005 000110  ADD   MINCYL(RO),R5 ;NEW CYLINDER ADDRESS
4865 017026 023705 001350  CMP   CYLIMT,R5    ;IS VALUE TOO LARGE ?
4866 017032 101002          BHI   2$          ;BR IF NOT
4867 017034 163705 001350  SUB   CYLIMT,R5    ;CORRECT VALUE
4868 017040 010560 000100  2$:  MOV   R5,$NCYL(RO) ;STORE CYLINDER ADDRESS IN DPB
4869 017044 122760 000003 000074  CMPB  #3,$NCODE(RO) ;WRITE HEADER & DATA ?
4870 017052 001013          BNE   RANSIZ      ;BR IF NOT
4871 017054 012760 000404 000102  MOV   #260.,$NWRDL(RO) ;CHANGE WORD LENGTH TO 260 FOR WRTHD ORDER
4872 017062 023727 001404 000404  CMP   MAXDL,#260.  ;CAN A FULL SECTOR BE WRITTEN ?
4873 017070 103062          BHIS  RANPAT      ;BR IF IT CAN
4874 017072 013760 001404 000102  MOV   MAXDL,$NWRDL(RO) ;CHANGE TRANSFER SIZE
4875 017100 000456          BR    RANPAT      ;CONTINUE WITH THE SELECTION
4876
4877          ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 4 & THE VALUE IN 'MAXDL'
4878
4879 017102 013705 001404  RANSIZ: MOV  MAXDL,R5 ;GET BUFFER SIZE
4880 017106 005737 001420  TST   WCSEL       ;SELECT A RANDOM WORD COUNT ?
4881 017112 001010          BNE   1$          ;BR IF NOT
4882 017114 005205          INC   R5          ;INCREMENT THE MAXIMUM SIZE
4883 017116 004737 027124  JSR   PC,GETREM    ;DIVIDE BY MAX VALUE
4884 017122 005705          TST   R5          ;IS THE REMAINDER 0 ?
4885 017124 001003          BNE   1$          ;NOT 0, CONTINUE
4886 017126 004737 032462  JSR   PC,$RAND     ;CYCLE THE RANDOM NUMBER GENERATOR
4887 017132 000763          BR    RANSIZ      ;TRY AGAIN
4888 017134 010560 000102  1$:  MOV   R5,$NWRDL(RO) ;WORD LENGTH TO CONTROL BLOCK
4889 017140 010546          MOV   R5,-(SP)    ;NEW WORD LENGTH ON STACK FOR CHECK
4890 017142 005046          CLR   -(SP)       ;MAKE UPPER DIVIDEND ZERO
4891 017144 012746 000400  MOV   #256.,-(SP) ;SECTOR SIZE IS THE DIVISOR
4892 017150 132760 000001 000074  BITB  #1,$NCODE(RO) ;SEE IF NEXT ORDER IS A HEADER ORDER
4893 017156 001402          BEQ   2$          ;BR IF NOT
4894 017160 062716 000004  ADD   #4,(SP)     ;ADD HEADER SIZE TO SECTOR SIZE
4895 017164 004737 027152  2$:  JSR   PC,LINKDV   ;DIVIDE BUFFER SIZE BY SECTOR SIZE
4896 017170 012616          MOV   (SP)+,(SP)  ;MOV REMAINDER UP THE STACK
4897 017172 021627 000004  CMP   (SP),#4 ;SEE IF REMAINDER LESS THAN 4
4898 017176 103012          BHIS  4$          ;BR IF NOT
4899 017200 005737 001420  TST   WCSEL       ;SELECTING RANDOM TRANSFER SIZES ?
4900 017204 001403          BEQ   3$          ;BR IF YES
4901 017206 161660 000102  SUB   (SP),$NWRDL(RO) ;ADJUST WORD LENGTH DOWNWARD

```



```
4902 017212 000404          BR      4$      ;CONTINUE
4903 017214 005726          3$:   TST      (SP)+ ;CORRECT THE STACK POINTER
4904 017216 004737 032462   JSR     PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
4905 017222 000727          BR      RANSIZ  ;TRY AGAIN
4906 017224 005726          4$:   TST      (SP)+ ;CORRECT THE STACK POINTER
4907 017226 122760 000002 000074  CMPB   #2,$NCODE(RO) ;SEE IF WRITE DATA
4908 017234 001004          BNE    RANXIT  ;BR IF NOT WRITE DATA
4909
4910          ;GET A RANDOM PATTERN NUMBER
4911
4912 017236 004737 017362   RANPAT: JSR    PC,GETPAT ;GET PATTERN CODE
4913 017242 110560 000075   MOVB   R5,$NPATC(RO) ;MOVE PATTERN CODE TO CONTROL BLOCK
4914 017246 012760 177777 000104  RANXIT: MOV   #-1,$NEXT(RO) ;SET PARAMETERS SELECTED INDICATOR
4915 017254 000207          RTS     PC      ;RETURN
4916
4917          ;ROUTINE TO SELECT A WRITE (OR WRITE CHECK) OPERATION
4918
4919 017256 012705 000004   RANWRT: MOV   #4,R5 ;WRITE OPERATION SELECTION DIVISOR
4920 017262 004737 027124   JSR    PC,GETREM ;GET SELECTION CODE
4921 017266 005737 001424   TST   'AUTOCK ;ARE WRITE CHECK ORDERS TO BE SELECTED
4922          ;RANDOMLY ?
4923          BEQ   1$ ;BR IF THEY ARE
4924 017274 152705 000002   BISB  #2,R5 ;SET CODE TO EXCLUDE WRITE CHECK ORDERS
4925 017300 000420          BR    3$ ;COMPLETE SELECTION
4926 017302 020527 000001   1$:   CMP   R5,#1 ;WRITE CHECK SELECTED ?
4927 017306 101015          BHI   3$ ;BR IF NOT
4928 017310 132760 000002 000024  BITB  #2,$CODE(RO) ;PREVIOUS WRITE OPERATION ?
4929 017316 001407          BEQ   2$ ;BR IF PREVIOUS WAS READ OR WRITE CHECK
4930 017320 116060 000024 000074  MOVB  $CODE(RO),$NCODE(RO) ;MOVE CODE TO 'NEXT CODE'
4931 017326 142760 000002 000074  BICB  #2,$NCODE(RO) ;CHANGE WRITE TO WRITE CHECK
4932 017334 000411          BR    5$ ;EXIT
4933 017336 052705 000002   2$:   BIS   #2,R5 ;CHANGE WRITE CHECK TO WRITE
4934 017342 005737 001416   3$:   TST   FORMAT ;WRITE HEADER ORDERS ALLOWED ?
4935 017346 001002          BNE   4$ ;BR IF THEY ARE
4936 017350 042705 000001   BIC   #1,R5 ;ALTER POSSIBLE WRITE HEADER
4937 017354 110560 000074   4$:   MOVB  R5,$NCODE(RO) ;SETUP 'NEXT' CODE
4938 017360 000207          5$:   RTS   PC ;RETURN
4939
4940          ;ROUTINE TO SELECT A PATTERN
4941
4942 017362 012705 000020   GETPAT: MOV   #20,R5 ;SELECT PATTERN
4943 017366 004737 027124   JSR    PC,GETREM ;GET CODE
4944 017372 005705          TST   R5 ;WAS PATTERN ZERO SELECTED ?
4945 017374 001003          BNE   1$ ;BR IF NOT ZERO
4946 017376 004737 032462   JSR    PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
4947 017402 000767          BR    GETPAT ;TRY AGAIN
4948 017404 006305          1$:   ASL   R5 ;MAKE CODE INTO TABLE INDEX
4949 017406 000207          RTS   PC
4950
4951          ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
4952          ;CALL:
4953          ;   MOV   #DPB,RO ;DPB ADDRESS
4954          ;   JSR   PC,SELPAR ;SELECT THE PARAMETERS
4955          ;   JSR   PC,GETPAR
4956          ;   RETURN
4957
```



```
4958 017410 010546          GETPAR: MOV      R5,-(SP)          ;SAVE R5
4959 017412 116060 000234 000027      MOVB     $RPCS1(R0),$PREV0(R0) ;SAVE CURRENT PARAMETERS
4960 017420 032760 000006 000074      BIT      #6,$NCODE(R0)        ;SEE IF NEXT OPERATION IS READ OR WRITE
4961 017426 001007          BNE      1$                    ;BR IF EITHER
4962 017430 016060 000012 000034      MOV      $CYL(R0),$PREVA+2(R0) ;SAVE STARTING CYLINDER
4963 017436 016060 000010 000032      MOV      $SEC(R0),$PREVA(R0)  ;SAVE STARTING SECTOR AND TRACK
4964 017444 000411          BR       2$
4965 017446 004737 022402          1$: JSR      PC,READR            ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
4966 017452 112660 000033      MOVB     (SP)+,$PREVA+1(R0)    ;TRACK ADDRESS
4967 017456 112660 000032      MOVB     (SP)+,$PREVA(R0)     ;SECTOR ADDRESS
4968 017462 016060 000272 000034      MOV      $RPCC(R0),$PREVA+2(R0);CURRENT CYLINDER
4969 017470 032777 000100 161442          2$: BIT      #SW06,@SWR        ;SWITCH 6 SET ?
4970 017476 001043          BNE      3$                    ;BR IF SET
4971 017500 116060 000074 000024      MOVB     $NCODE(R0),$CODE(R0) ;LOGICAL CODE FOR OPERATION
4972 017506 116005 000074          MOVB     $NCODE(R0),R5        ;LOAD R5 FOR USE AS TABLE INDEX
4973 017512 116560 001760 000002      MOVB     COMTBL(R5),$COMND(R0) ;RPO4 COMMAND CODE
4974 017520 116060 000075 000030      MOVB     $NPATC(R0),$PATTC(R0);PATTERN CODE
4975 017526 016060 000076 000010      MOV      $NSEC(R0),$SEC(R0)   ;TRACK AND SECTOR ADDRESSES
4976 017534 016060 000100 000012      MOV      $NCYL(R0),$CYL(R0)   ;CYLINDER ADDRESS
4977 017542 016060 000102 000020      MOV      $NWRDL(R0),$WRDL(R0) ;BUFFER SIZE
4978 017550 016060 000102 000004      MOV      $NWRDL(R0),$WRDM(R0) ;WORD COUNT FOR THE RH11
4979 017556 005460 000004          NEG      $WRDM(R0)            ;COMPLEMENT IT
4980 017562 012760 000400 000022      MOV      #256,$SSEC(R0)      ;INITIAL VALUE OF SECTOR SIZE
4981 017570 032760 000001 000024      BIT      #1,$CODE(R0)        ;HEADER OPERATION ?
4982 017576 001403          BEQ      3$                    ;BR IF NOT
4983 017600 062760 000004 000022      ADD      #4,$SSEC(R0)        ;ADD HEADER SIZE
4984 017606 005060 000104          3$: CLR      $NEXT(R0)         ;RESET 'PARAMETERS LOADED' INDICATOR
4985 017612 012605          MOV      (SP)+,R5            ;RESTORE R5
4986 017614 000207          RTS      PC                  ;RETURN
4987
4988          ;ROUTINE TO COMPRESS A LIST
4989          ;CALL:
4990          ;      MOV      #ADDRS,R1          ;COMPRESS LIST STARTING AT THIS ADDRESS
4991          ;      JSR      PC,CMPRES
4992          ;      RETURN
4993
4994 017616 016111 000002          CMPRES: MOV      2(R1),(R1)     ;COMPRESS THE TABLE IN R1
4995 017622 001403          BEQ      1$                    ;BR WHEN ZERO FOUND
4996 017624 062701 000002          ADD      #2,R1                ;INCREMENT R1
4997 017630 090772          BR      CMPRES                ;CONTINUE COMPRESSING TABLE
4998 017632 000207          1$: RTS      PC              ;RETURN
4999
5000          ;ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
5001          ;CALL:
5002          ;      MOV      #DPB,R0          ;DPB ADDRESS
5003          ;      MOV      #-1,$PACK(R0)    ;'WRITE PACK' FLAG
5004          ;      OR
5005          ;      MOV      #1,$PACK(R0)    ;'READ PACK' FLAG
5006          ;      JSR      PC,WRTPK
5007          ;      RETURN
5008
5009 017634 004737 032462          WRTPK: JSR      PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
5010 017640 005760 000040          TST      $OPERC+2(R0)        ;SEE IF FIRST OPERATION
5011 017644 001007          BNE      WRTPK1              ;BR IF UPPER WORD OF COUNTER NOT ZERO
5012 017646 005760 000036          TST      $OPERC(R0)         ;LOWER WORD ZERO ?
5013 017652 001004          BNE      WRTPK1              ;BR IF NOT 1ST OPERATION
```



```
5014 017654 105760 000026          TSTB   $PACK(R0)          ;SEE WHICH - 'R' OR 'W'  
5015 017660 100503          BMI    WRTPK3            ;BR IF 'W'  
5016 017662 000470          BR     WRTPK2            ;'R' OPERATION  
5017 017664 116060 000234 000027 WRTPK1: MOVB   $RPCS1(R0), $PREV0(R0) ;SAVE CURRENT PARAMETERS  
5018 017672 004737 022402          JSR    PC, READDR        ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES  
5019 017676 112660 000033          MOVB   (SP)+, $PREVA+1(R0) ;TRACK ADDRESS  
5020 017702 112660 000032          MOVB   (SP)+, $PREVA(R0)  ;SECTOR ADDRESS  
5021 017706 016060 000272 000034          MOV    $RPCC(R0), $PREVA+2(R0) ;CURRENT CYLINDER  
5022 017714 016060 000242 000010          MOV    $RPDA(R0), $SEC(R0)  ;NEW SECTOR & TRACK ADDRESS  
5023 017722 016060 000270 000012          MOV    $RPCA(R0), $CYL(R0)  ;NEW CYLINDER ADDRESS  
5024 017730 026060 000012 000106          CMP    $CYL(R0), $MAXCYL(R0) ;SEE IF AT END  
5025 017736 103427          BLO    2$                ;BR IF LESS THAN 'MAXCYL'  
5026 017740 101004          BHI    1$                ;BR IF GREATER THAN 'MAXCYL'  
5027 017742 126060 000011 000112          CMPB   $TRK(R0), $MAXTRK(R0) ;SEE IF AT MAX TRACK  
5028 017750 101422          BLOS   2$                ;BR IF NOT GREATER  
5029 017752 116060 000114 000011 1$:  MOVB   MINTRK(R0), $TRK(R0) ;RESET TRACK ADDRESS  
5030 017760 116060 000120 000010          MOVB   MINSEC(R0), $SEC(R0) ;RESET SECTOR ADDRESS  
5031 017766 016060 000110 000012          MOV    MINCYL(R0), $CYL(R0) ;RESET CYLINDER ADDRESS  
5032 017774 004737 026724          JSR    PC, EOP2          ;DROP THE DRIVE (NORMAL TERMINATION)  
5033 020000 032777 000020 161132          BIT    #SW04, @SWR       ;IS SWITCH 4 SET ?  
5034 020006 001003          BNE    2$                ;BR IF SET  
5035 020010 005726          TST    (SP)+             ;INCREMENT THE STACK POINTER  
5036 020012 000137 005462          JMP    MAIN              ;RETURN DIRECTLY TO 'MAIN'  
5037 020016 013760 001404 000020 2$:  MOV    $MAXDL, $WRDL(R0)  ;BUFFER SIZE IS MAXIMUM  
5038 020024 013760 001404 000004          MOV    $MAXDL, $WRDM(R0)  ;WORD COUNT  
5039 020032 005460 000004          NEG    $WRDM(R0)         ;CHANGE WORD COUNT TO 2'S COMPLEMENT  
5040 020036 105760 000026          TSTB   $PACK(R0)         ;READ OR WRITE ?  
5041 020042 100412          BMI    WRTPK3            ;BR IF WRITE  
5042 020044 012760 000404 000022 WRTPK2: MOV    #260., $SSEC(R0) ;SECTOR SIZE FOR READ  
5043 020052 112760 000005 000024          MOVB   #5, $CODE(R0)     ;CODE FOR READ HEADER & DATA  
5044 020060 112760 000173 000002          MOVB   #RDHD, $COMND(R0) ;DRIVE CODE FOR OPERATION  
5045 020066 000415          BR     WRTPK4            ;SET UP FOR EXIT  
5046 020070 012760 000400 000022 WRTPK3: MOV    #256., $SSEC(R0) ;SECTOR SIZE  
5047 020076 112760 000002 000024          MOVB   #2, $CODE(R0)     ;CODE FOR WRDAT  
5048 020104 112760 000161 000002          MOVB   #WRTDAT, $COMND(R0) ;OP CODE  
5049 020112 004737 017362          JSR    PC, GETPAT        ;GET PATTERN CODE  
5050 020116 110560 000030          MOVB   R5, $PATIC(R0)    ;PATTERN CODE  
5051 020122 005060 000104          WRTPK4: CLR    $NEXT(R0)  ;CLEAR 'PARAMETER SELECTED' INDICATOR  
5052 020126 000207          RTS    PC                ;RETURN
```

```
5053  
5054 ;ROUTINE TO DETERMINE OF ERROR IS AT A LOCATION ON THE PACK DEFINED  
5055 ; IN THE BAD TRACK/SECTOR TABLE FOR THE DRIVE.  
5056 ;CALL:  
5057 ; JSR    PC, SPOTCK  
5058 ; RETURN1 ;ERROR AT AN ADDRESS IN TABLE  
5059 ; RETURN2 ;NO TABLE ENTRY FOR ERROR ADDRESS OR  
5060 ; ;PARAMETER 'NOTPRT' IS 0  
5061 ;
```

```
5062 SPOTCK: 020130  
5063 020130 010146          MOV    R1, -(SP)         ;;PUSH R1 ON STACK  
5064 020132 010246          MOV    R2, -(SP)         ;;PUSH R2 ON STACK  
5065 020134 012701 000124          MOV    # $BDSEC, R1     ;;INCREMENT FOR BAD SECTOR TABLE  
5066 020140 060001          ADD    R0, R1           ;ADD THE BLOCK'S STARTING ADDRESS  
5067 020142 012702 000020          MOV    #16., R2        ;BAD SECTOR TABLE SIZE COUNT  
5068 020146 021160 000272 1$:  CMP    (R1), $RPCC(R0)  ;IS CYLINDER IN THE TABLE ?  
5069 020152 001022          BNE    4$                ;BR IF NOT
```



```
5070 020154 105761 000003      TSTB    3(R1)      ;TRACK ENTRY ?
5071 020160 100426              BMI     5$        ;BR IF NOT
5072 020162 004737 022402      JSR    PC,READDR  ;DECREMENT THE SECTOR/TRACK ADDRESS
5073 020166 122661 000003      CMPB   (SP)+,3(R1);COMPARE THE TRACK ADDRESS
5074 020172 001011              BNE    3$        ;BR IF IT IS NOT EQUAL
5075 020174 105761 000002      TSTB   2(R1)     ;IS A SECTOR ADDRESS IN THE TABLE ?
5076 020200 100002              BPL    2$        ;BR IF ONE IS
5077 020202 005726              TST    (SP)+     ;INCREMENT THE STACK POINTER
5078 020204 000414              BR     5$        ;DISPLAY THE MESSAGE
5079 020206 122661 000002      2$:    CMPB   (SP)+,2(R1);COMPARE THE SECTOR ADDRESS
5080 020212 001002              BNE    4$        ;BR IF NOT EQUAL
5081 020214 000410              BR     5$        ;CHECK 'NOTPRT'
5082 020216 005726              3$:    TST    (SP)+ ;INCREMENT THE STACK POINTER
5083 020220 062701 000004      4$:    ADD    #4,R1 ;GO TO THE NEXT LOCATION IN THE TABLE
5084 020224 005711              TST    (R1)     ;PAST THE TABLE ENTRIES ?
5085 020226 100411              BMI    6$        ;BR IF PAST
5086 020230 005302              DEC    R2       ;DECREMENT THE MAXIMUM ENTRY COUNT
5087 020232 001345              BNE    1$        ;BR IF MORE TO CHECK
5088 020234 000406              BR     6$        ;END, EXIT
5089 020236 005737 001426      5$:    TST    NOTPRT ;PRINT THE ERROR ANYWAY ?
5090 020242 001006              BNE    7$        ;BR IF NOT
5091 020244 012737 177777 001264  MOV    #-1,BADSEC ;SET THE INDICATOR FOR THE IDENTIFICATION LINE
5092 020252 062766 000002 000004  6$:    ADD    #2,4(SP);INCREMENT THE RETURN
5093 020260              7$:
5094 020260 012602              MOV    (SP)+,R2  ;;POP STACK INTO R2
5095 020262 012601              MOV    (SP)+,R1  ;;POP STACK INTO R1
5096 020264 000207              RTS    PC        ;RETURN
```

5097
5098
5099

;;*****

.SBTTL ERROR MESSAGE GENERATION ROUTINES

5100
5101
5102
5103

;;*****

:PRINT LINE 1 OF ERROR MESSAGE:
:'HH:MM:SS'

```
5107 020266 032777 002000 160644 LINE1:
5108 020274 001402
5109 020276 104401 001160
5110 020302 032777 020000 160630 1$:
5111 020310 001403
5112 020312 104414 001165
5113 020316 000404
5114 020320 004737 023536 2$:
5115 020324 104414 052324
5116 020330 000207 3$:
```

```
BIT    #SW10,@SWR ;SWITCH 10 SET ?
BEQ    1$         ;BR IF NOT
TYPE   ,SBELL    ;RING THE BELL
1$:    BIT    #SW13,@SWR ;INHIBIT TYPEOUT ?
BEQ    2$         ;BR IF NOT
DISPLY ,SCRLF    ;CR-LF
RR     3$         ;EXIT
2$:    JSR    PC,$TIME ;TYPE THE TIME
DISPLY ,LINSPO   ;SPACES
3$:    RTS    PC    ;RETURN & TYPE DESCRIPTION
```

5117
5118
5119
5120
5121
5122
5123
5124
5125

:PRINT LINE 2 OF ERROR MESSAGE
:'PRESENT ORDER = XXXX PREVIOUS ORDER = XXXX'
:'* ERROR AT BAD TRACK/SECTOR'
:'DRV RPCS1 RPCS2 RPDS1 RPER1 RPER2 RPER3 RPEC1 RPEC2'
:'RPWC RPBA RPDA RPAS RPLA RPDB RPMR RPD1'
:'RPSN RPOF RPCA RPCC STATUS'
:'BUS ADDRESS OR WORD COUNT NOT CONSISTENT'
:'RPBA = XXXXXX RPWC = XXXXXX'

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

```
5126  
5127  
5128 020332  
5129 020332 010346  
5130 020334 010446  
5131 020336 010546  
5132 020340 104414 001165  
5133 020344 005037 020472  
5134 020350 005004  
5135 020352 012737 050410 020472  
5136 020360 116004 000234  
5137 020364 042704 177701  
5138 020370 004737 020426  
5139 020374 005737 020476  
5140 020400 001440  
5141 020402 012737 050431 020472  
5142 020410 116004 000027  
5143 020414 042704 177701  
5144 020420 004737 020426  
5145 020424 000426  
5146 020426 005005  
5147 020430 126504 001766  
5148 020434 001405  
5149 020436 105765 001766  
5150 020442 100402  
5151 020444 005205  
5152 020446 000770  
5153 020450 006305  
5154 020452 006305  
5155 020454 006305  
5156 020456 012737 002010 020476  
5157 020464 060537 020476  
5158 020470 104414  
5159 020472 000000  
5160 020474 104414  
5161 020476 000000  
5162 020500 000207  
5163 020502 005737 001264  
5164 020506 001404  
5165 020510 104414 001165  
5166 020514 104414 050455  
5167 020520 104414 001165  
5168 020524 104414 050004  
5169 020530 104414 052324  
5170 020534 013746 001246  
5171 020540 004737 022362  
5172 020544 104414 052323  
5173 020550 012705 050330  
5174 020554 004737 020704  
5175 020560 032777 000040 160352  
5176 020566 001014  
5177 020570 104414 050110  
5178 020574 012705 050352  
5179 020600 004737 020704  
5180 020604 104414 050207  
5181 020610 012705 050374
```

LINE2:
MOV R3,-(SP) ;:PUSH R3 ON STACK
MOV R4,-(SP) ;:PUSH R4 ON STACK
MOV R5,-(SP) ;:PUSH R5 ON STACK
DISPLY ,%CRLF ;:CR-LF
CLR 4\$;:CLEAR MESSAGE ADDRESS STORAGE
CLR R4 ;:WORKING REGISTER
MOV #LIN2C,4\$;:ADDRESS OF 'PRESENT ORDER = ' MSG
MOVB \$RPCS1(R0),R4 ;:GET THE OPCODE
BIC #^C76,R4 ;:SAVE ONLY SIGNIFICANT BITS
JSR PC,1\$;:TYPE THE FIRST MNEMONIC
TST 5\$;:SEE IF MNEMONIC ENTRY FOUND
BEQ LINE2A ;:BR IF NOT
MOV #LIN2P,4\$;:ADDRESS OF 'PREVIOUS ORDER = ' MSG
MOVB \$PREVO(R0),R4 ;:PREVIOUS OPERATION CODE
BIC #^C76,R4 ;:SAVE ONLY SIGNIFICANT BITS
JSR PC,1\$;:TYPE THE PREVIOUS MNEMONIC
BR LINE2A ;:CONTINUE
1\$: CLR R5 ;:CLEAR THE TABLE INDEX
2\$: CMPB OPTBL(R5),R4 ;:LOOK FOR THE OPCODE
BEQ 3\$;:BR WHEN OPCODE COUNT EQUALS OPCODE
TSTB OPTBL(R5) ;:LOOK FOR END OF TABLE
BMI 3\$;:BR IF END
INC R5 ;:INCREMENT THE POINTER
BR 2\$;:CONTINUE - NOT END OF TABLE
3\$: ASL R5 ;:SHIFT INDEX
ASL R5 ;:SHIFT THE INDEX
ASL R5 ;:SHIFT THE INDEX
MOV #MNTBL,5\$;:ADDRESS OF ASCII TEXT TABLE
ADD R5,5\$;:ADD THE INDEX
DISPLY ;:TYPE IT
4\$: .WORD 0 ;:ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
DISPLY ;:TYPE THE OPERATION MNEMONIC
5\$: .WORD 0 ;:ADDRESS OF MESSAGE
RTS PC ;:RETURN TO MAIN ROUTINE
LINE2A: TST BADSEC ;:PRINT THE BAD SECTOR LINE ?
BEQ LINE2B ;:BR IF NOT
DISPLY ,%CRLF ;:CR-LF
DISPLY ,LIN2S ;:ERROR ADDRESS DEFINED AS BAD AREA
LINE2B: DISPLY ,%CRLF ;:CR-LF
DISPLY ,DH14 ;:STANDARD RP04/5/6 REGISTER HEADER
DISPLY ,LINSPO ;:TYPE A SPACE
MOV UNIT,-(SP) ;:PUT THE DRIVE NUMBER ON THE STACK
JSR PC,LINDEC ;:TYPE DRIVE NUMBER
DISPLY ,LINSPO ;:SPACES
MOV #DT14,R5 ;:REGISTER INDEXES
JSR PC,3\$;:PRINT THE REGISTERS
BIT #SW05,@SWR ;:PRINT THE OPTIONAL REGISTERS ?
BNE 1\$;:BR IF NOT
DISPLY ,DH15 ;:SECOND DATA LINE
MOV #DT15,R5 ;:SECOND DATA LINE
JSR PC,3\$;:PRINT THEM
DISPLY ,DH16 ;:THIRD DATA LINE
MOV #DT16,R5 ;:THIRD DATA LINE


```

5182 020614 004737 020704      JSR    PC,3$          ;PRINT THE REGISTERS
5183 020620 032760 000100 000016 1$: BIT    #BIT6,$STATUS(R0) ;DATA ERROR ?
5184 020626 001422              BEQ    2$             ;BR IF NOT
5185 020630 016046 000020      MOV    $WRDL(R0),-(SP) ;TRANSFER SIZE
5186 020634 066C16 000236      ADD    $RPWC(R0),'SP' ;ADD REMAINING WORD COUNT
5187 020640 006316              ASL    (SP)           ;CONVERT TO AN BYTE INCREMENT
5188 020642 066016 000006      ADD    $BUF(R0),(SP)  ;BUFFER STARTING ADDRESS
5189 020646 022660 000240      CMP    (SP)+,$RPBA(R0) ;CORRECT BUFFER ADDRESS ?
5190 020652 001410              BEQ    2$             ;BR IF YES
5191 020654 104414 047377      DISPLY ,EM46          ;'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
5192 020660 104414 001165      DISPLY ,%CRLF         ;CR-LF
5193 020664 004737 020776      JSR    PC,LINE3D      ;PRINT LINE 3D OF ERROR MESSAGE
5194 020670 004737 021414      JSR    PC,LINE4       ;PRINT LINE 4 OF ERROR MESSAGE
5195 020674              2$:
5196 020674 012605              MOV    (SP)+,R5       ;;POP STACK INTO R5
5197 020676 012604              MOV    (SP)+,R4       ;;POP STACK INTO R4
5198 020700 012603              MOV    (SP)+,R3       ;;POP STACK INTO R3
5199 020702 000207              RTS    PC             ;RETURN TO ERROR PROCESSING ROUTINE
5200 020704 012546 3$: MOV    (R5)+,-(SP)     ;PUT THE REGISTER INDEX ON THE STACK
5201 020706 060016              ADD    R0,(SP)        ;ADD DRIVE'S TABLE ADDRESS
5202 020710 017646 000000      MOV    @ (SP),-(SP)   ;VALUE
5203 020714 004737 022330      JSR    PC,LINOC7      ;TYPE IT
5204 020720 005726              TST    (SP)+          ;CORRECT THE STACK POINTER
5205 020722 104414 052323      DISPLY ,LINSF         ;PRINT 2 SPACES
5206 020726 005715              TST    (R5)           ;AT END OF LINE ?
5207 020730 001365              BNE    3$             ;BR IF NOT
5208 020732 104414 001165      4$: DISPLY ,%CRLF     ;CR-LF
5209 020736 000207              RTS    PC             ;RETURN
5210
5211              ;PRINT LINE 3 OF ERROR MESSAGE
5212              ;'ERROR AT CCC TT SS  PREVIOUS ADR = CCC TT SS'
5213
5214 020740 104414 050511      LINE3: DISPLY ,LINM3   ;LINE 3 ENTRANCE
5215 020744 000517              BR     LIN3.1         ;FINISH PRINTOUT
5216
5217              ;PRINT LINE 3A OF ERROR MESSAGE
5218              ;'START CYL = CCC  END CYL = CCC'
5219
5220 020746 104414 050527      LINE3A: DISPLY ,LINN3  ;LINE 3A ENTRANCE
5221 020752 000514              BR     LIN3.1         ;FINISH ERROR LINE
5222
5223              ;PRINT LINE 3B OF ERROR MESSAGE
5224              ;'START CYL = CCC  END CYL = CCC  ACTUAL CYL = CCC'
5225
5226 020754 004737 021316      LINE3B: JSR    PC,LIN3.3 ;LINE 3B ENTRANCE
5227 020760 104414 001165      DISPLY ,%CRLF
5228 020764 000207              RTS    PC
5229
5230              ;PRINT LINE 3C OF ERROR MESSAGE
5231              ;'START CYL = CCC  END CYL = CCC  ACTUAL CYL = CCC  TRK = TT'
5232
5233 020766 004737 021316      LINE3C: JSR    PC,LIN3.3 ;LINE 3C ENTRANCE
5234 020772 000137 021350      JMP    LIN3.4         ;FINISH MESSAGE
5235
5236              ;PRINT LINE 3D OF ERROR MESSAGE
5237              ;'RPBA = XXXXXX  RPWC = XXXXXX'

```



```
5238
5239 020776 032777 000040 160134 LINE3D: BIT #SW05,@SWR ;SWITCH 5 SET ?
5240 021004 001416 BEQ 1$ ;BR IF IT IS
5241 021006 104414 050700 DISPLY ,LINB3 ;'RPBA = '
5242 021012 016046 000240 MOV $RPBA(R0),-(SP) ;BUFFER ADDR REG CONTENTS
5243 021016 004737 022330 JSR PC,LINOCT ;CONVERT TO OCTAL AND TYPE IT
5244 021022 104414 050710 DISPLY ,LINW3 ;' RPWC = '
5245 021026 016046 000236 MOV $RPWC(R0),-(SP) ;WORD COUNT REGISTER CONTENTS
5246 021032 004737 022330 JSR PC,LINOCT ;CONVERT TO OCTAL AND TYPE IT
5247 021036 104414 001165 DISPLY ,%CRLF
5248 021042 000207 1$: RTS PC
5249
5250 ;PRINT LINE 3E OF ERROR MESSAGE
5251 ;'START CYL = CCC START TRK = TT START SEC = SS'
5252
5253 021044 104414 050574 LINE3E: DISPLY ,LINS3 ;'START CYL = '
5254 021050 016046 000012 MOV $CYL(R0),-(SP) ;MOVE CYL TO STACK
5255 021054 004737 022362 JSR PC,LINDEC ;TYPE IT IN DECIMAL
5256 021060 104414 052323 DISPLY ,LINS3 ;SPACES
5257 021064 104414 050722 DISPLY ,LINST3 ;'START TRK = '
5258 021070 005046 CLR -(SP) ;CLEAR STACK
5259 021072 116016 000011 MOV $TRK(R0),(SP) ;TRACK TO STACK
5260 021076 004737 022362 JSR PC,LINDEC ;TYPE IT IN DECIMAL
5261 021102 104414 052323 DISPLY ,LINS3 ;SPACES
5262 021106 104414 050737 DISPLY ,LINS3 ;'START SEC = '
5263 021112 005046 CLR -(SP) ;CLEAR STACK
5264 021114 116016 000010 MOV $SEC(R0),(SP) ;SECTOR ADDR TO STACK
5265 021120 004737 022362 JSR PC,LINDEC ;TYPE IT IN DECIMAL
5266 021124 104414 001165 DISPLY ,%CRLF
5267 021130 000207 RTS PC
5268
5269 ;PRINT LINE 3F OF ERROR MESSAGE
5270 ;'RPDA = XXXXXX RPCA = XXXXXX'
5271
5272 021132 032777 000040 160000 LINE3F: BIT #SW5,@SWR ;SWITCH 5 SET ?
5273 021140 001420 BEQ 1$ ;BR IF NOT
5274 021142 104414 050670 DISPLY ,LINDA3 ;'RPDA = '
5275 021146 016046 000242 MOV $RPDA(R0),-(SP) ;PUT SECTOR/TRACK ADDRESS ON THE STACK
5276 021152 004737 022330 JSR PC,LINOCT ;TYPE IT
5277 021156 104414 052323 DISPLY ,LINS3 ;SPACES
5278 021162 104414 050657 DISPLY ,LINCA3 ;'RPCA = '
5279 021166 016046 000270 MOV $RPCA(R0),-(SP) ;PUT DESIRED CYLINDER ADDRESS ON THE STACK
5280 021172 004737 022330 JSR PC,LINOCT ;TYPE IT
5281 021176 104414 001165 DISPLY ,%CRLF
5282 021202 000207 1$: RTS PC
5283
5284 ;'CCC TT SS PREV ADR = CCC TT SS'
5285
5286 021204 016046 000272 LIN3.1: MOV $RPCC(R0),-(SP) ;PUT CYLINDER ADDR ON STACK
5287 021210 004737 022362 JSR PC,LINDEC ;TYPE IT IN DECIMAL
5288 021214 104414 050524 DISPLY ,T ;PRINT 'T'
5289 021220 004737 022402 JSR PC,READDR ;DECREMENT TRACK AND SECTOR ADDRESSES
5290 021224 004737 022362 JSR PC,LINDEC ;TYPE TRACK IN DECIMAL
5291 021230 104414 050550 DISPLY ,S ;PRINT 'S'
5292 021234 004737 022362 JSR PC,LINDEC ;TYPE SECTOR ADDRESS
5293 021240 104414 050553 DISPLY ,LINS3 ;PRINT 'PREV ADDR'
```



```
5350 ;'GOOD DATA = XXXXXX BAD DATA = XXXXXX SECT POS = XXX'  
5351  
5352 021504 104414 051040  
5353 021510 162760 000002 000240 LINE5: DISPLY ,LIND5 ;PRINT 'GOOD DATA'  
5354 021516 017046 000240 SUB #2,$RPBA(R0) ;BACK THE ADDRESS UP  
5355 021522 004737 022330 MOV @,$RPBA(R0),-(SP) ;'GOOD' DATA - AT THE BUFFER LOCATION  
5356 021526 104414 051055 JSR PC,LINOC ;TYPE IT  
5357 021532 016046 000256 DISPLY ,LINB5 ;PRINT 'BAD DATA'  
5358 021536 004737 022330 MOV $RPDB(R0),-(SP) ;BAD DATA FROM BUFFER  
5359 021542 016046 000236 JSR PC,LINOC ;TYPE IT  
5360 021546 066016 000020 MOV $RPWC(R0),-(SP) ;WORD LENGTH ON STACK  
5361 021552 005046 ADD $WRDL(R0),(SP) ;MAKE INTO A POSITIVE NUMBER  
5362 021554 016046 000022 CLR -(SP) ;UPPER DIVIDEND TO ZERO  
5363 021560 004737 027152 MOV $SSEC(R0),-(SP) ;SECTOR SIZE ON THE STACK  
5364 021564 012616 JSR PC,LINKDV ;DIVIDE WORDS XFERED BY SECTOR SIZE  
5365 021566 104414 051073 MOV (SP)+,(SP) ;MOVE REMAINDER UP THE STACK  
5366 021572 004737 022362 DISPLY ,LINS5 ;PRINT 'SECT POS'  
5367 021576 104414 001165 JSR PC,LINDEC ;TYPE THE POSITION  
5368 021602 000207 DISPLY ,%CRLF  
5369 RTS PC  
5370 ;PRINT LINE 5A OF THE ERROR MESSAGE  
5371 ;'HEADER FROM ERROR SECTOR XXXXXX XXXXXX XXXXXX XXXXXX'  
5372  
5373 021604 104414 051111 LINE5A: DISPLY ,LINS5 ;'HEADER CONTENTS OF ERROR SECTOR'  
5374 021610 013746 054746 MOV CYLDER, -(SP) ;HEADER POSITION  
5375 021614 004737 022330 JSR PC,LINOC ;TYPE IT  
5376 021620 104414 052323 DISPLY ,LINS5 ;SPACES  
5377 021624 013746 054750 MOV CYLDER+2, -(SP) ;HEADER POSITION +2  
5378 021630 004737 022330 JSR PC,LINOC ;TYPE IT  
5379 021634 104414 052323 DISPLY ,LINS5 ;SPACES  
5380 021640 013746 054752 MOV CYLDER+4, -(SP) ;HEADER POSITION +4  
5381 021644 004737 022330 JSR PC,LINOC ;TYPE IT  
5382 021650 104414 052323 DISPLY ,LINS5 ;SPACES  
5383 021654 013746 054754 MOV CYLDER+6, -(SP) ;HEADER POSITION +6  
5384 021660 004737 022330 JSR PC,LINOC ;TYPE IT  
5385 021664 104414 052323 DISPLY ,LINS5 ;SPACES  
5386 021670 104414 001165 DISPLY ,%CRLF  
5387 021674 000207 RTS PC  
5388  
5389 ;PRINT LINE 5B OF ERROR MESSAGE  
5390 ;'RPEC1 = XXXXXX RPEC2 = XXXXXX'  
5391  
5392 021676 104414 051145 LINE5B: DISPLY ,LINEP5 ;'RPEC1 = '  
5393 021702 016046 000300 MOV $RPEC1(R0), -(SP) ;PUT REGISTER CONTENTS ON THE STACK  
5394 021706 004737 022330 JSR PC,LINOC ;TYPE IT  
5395 021712 104414 052323 DISPLY ,LINS5 ;SPACES  
5396 021716 104414 051156 DISPLY ,LINEO5 ;' RPEC2 = '  
5397 021722 016046 000302 MOV $RPEC2(R0), -(SP) ;PUT REGISTER CONTENTS ON THE STACK  
5398 021726 004737 022330 JSR PC,LINOC ;TYPE IT  
5399 021732 104414 001165 DISPLY ,%CRLF  
5400 021736 000207 RTS PC ;RETURN  
5401  
5402 ;PRINT LINE 6 OF ERROR MESSAGE  
5403 ;'SECTOR IS ECC CORRECTABLE'  
5404  
5405 021740 104414 051170 LINE6: DISPLY ,LINB6 ;ECC CORRECTABLE
```



```
5406 021744 104414 001165          DISPLY ,SCLRF
5407 021750 000207          RTS      PC
5408
5409          ;PRINT LINE 6A OF THE ERROR MESSAGE
5410          ;'SECTOR READ CORRECTLY AT OFFSET N'
5411
5412 021752 104414 051223  LINE6A: DISPLY ,LINC6          ;PRINT 'READ CORRECTLY AT OFFSET N'
5413 021756 000411          BR      LIN6.1          ;TYPE THE REST OF THE LINE
5414
5415          ;PRINT LINE 6B OF THE ERROR MESSAGE
5416          ;'SECTOR IS ECC CORRECTABLE AT OFFSET N'
5417
5418 021760 104414 051170  LINE6B: DISPLY ,LINB6          ;PRINT 'SECTOR IS ECC CORRECTABLE '
5419 021764 000406          BR      LIN6.1
5420
5421          ;PRINT LINE 6C OF THE ERROR MESSAGE
5422          ;'CORRECTED ON NTH RETRY'
5423
5424 021766 104414 051252  LINE6C: DISPLY ,LING6          ;'CORRECTED ON NTH RETRY'
5425 021772 000414          BR      LIN6.2          ;TYPE THE REST OF THE LINE
5426
5427          ;PRINT LINE 6D OF THE ERROR MESSAGE
5428          ;'UNCORRECTABLE AFTER N RETRIES'
5429
5430 021774 104414 051301  LINE6D: DISPLY ,LINUO6          ;'UNCORRECTABLE AFTER N RETRIES'
5431 022000 000411          BR      LIN6.2          ;FINISH
5432
5433          ;TYPE THE OFFSET VALUE IN MICRO-INCHES
5434
5435 022002 006301          LIN6.1: ASL      R1          ;DOUBLE THE OFFSET TABLE INDEX
5436 022004 016137 002240 022014  MOV      OFMTBL(R1),1$      ;ADDRESS OF OFFSET POSITION MESSAGE
5437 022012 104414          DISPLY
5438 022014 000000 1$:      .WORD      0          ;OFFSET VALUE
5439 022016 104414 001165  DISPLY ,SCLRF
5440 022022 000207          RTS      PC
5441
5442          ;RETRY COUNT TYPEOUT
5443
5444 022024 005046          LIN6.2: CLR      -(SP)          ;CLEAR STACK
5445 022026 113716 001253  MOV      RETRY+1,(SP)      ;RETRY COUNT
5446 022032 004737 022362  JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
5447 022036 104414 051270  DISPLY ,LINR6          ;'RETRY'
5448 022042 104414 001165  DISPLY ,SCLRF
5449 022046 000207          RTS      PC
5450
5451          ;PRINT LINE 7 OF THE ERROR MESSAGE
5452          ;'ORDERS:XXXXX TOTAL ERRORS:XXX WRDS XFRD:XXXXXXX WRDS READ:XXXXXXX'
5453
5454 022050 104414 051354  LINE7:  DISPLY ,LIN70          ;PRINT ORDER COUNT
5455 022054 012746 000036  MOV      #$OPERC,-(SP)      ;TO STACK
5456 022060 060016          ADD      RO,(SP)          ;ADD THE BASE ADDRESS
5457 022062 004737 032660  JSR      PC,$DB2D          ;CONVERT IT
5458 022066 004737 027466  JSR      PC,$SUPRS          ;PRINT IT
5459 022072 104414 051433  DISPLY ,LIN71          ;TOTAL ERRORS
5460 022076 016046 000056  MOV      $TOTAL(RO),-(SP)      ;TO STACK
5461 022102 004737 022362  JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
```



```
5462 022106 104414 051445      DISPLY ,LIN7X      ;PRINT 'WRDS XFR'  
5463 022112 012746 000046      MOV  #STRANS,-(SP) ;ADDRESS OF LOW WORD ON STACK  
5464 022116 060016              ADD  RO,(SP)      ;  
5465 022120 004737 032660      JSR  PC,$DB2D     ;CONVERT  
5466 022124 004737 027466      JSR  PC,$SUPRS    ;PRINT  
5467 022130 104414 051461      DISPLY ,LIN7R     ;'BITS READ'  
5468 022134 012746 000052      MOV  #SREAD,-(SP) ;LOW WORD ADDRESS  
5469 022140 060016              ADD  RO,(SP)      ;  
5470 022142 004737 032660      JSR  PC,$DB2D     ;CONVERT  
5471 022146 004737 027466      JSR  PC,$SUPRS    ;PRINT  
5472 022152 104414 001165      DISPLY ,SCLF      ;  
5473 022156 104414 001165      DISPLY ,SCLF      ;  
5474 022162 032777 100000 156750 BIT  #SW15,@SWR    ;SEE IF 'HALT ON ERROR' - SWITCH 15  
5475 022170 001401              BEQ  1$           ;BR IF NOT  
5476 022172 000000              HALT              ;SWITCH 15 HALT  
5477 022174 000207      1$: RTS  PC  
5478  
5479      ;PRINT LINE 7A OF ERROR MESSAGE  
5480      ;'ORDERS:XXXXX TOTAL SEEKS=XXXXX TOTAL MISPOS ERR = XXX TOTAL SKI,OCYL ERR = XXX'  
5481  
5482 022176 104414 051354      LINE7A: DISPLY ,LIN7O ;'ORDERS = '  
5483 022202 012746 000036      MOV  #SOPERC,-(SP) ;ORDER COUNT INCREMENT  
5484 022206 060016              ADD  RO,(SP)      ;ADD BASE ADDRESS  
5485 022210 004737 032660      JSR  PC,$DB2D     ;CONVERT THE COUNT  
5486 022214 004737 027466      JSR  PC,$SUPRS    ;PRINT IT  
5487 022220 104414 051364      DISPLY ,LIN7P     ;'TOTAL SEEKS = '  
5488 022224 012746 000042      MOV  #SPOSIT,-(SP) ;TOTAL SEEKS  
5489 022230 060016              ADD  RO,(SP)      ;DEVICE TABLE ADDRESS  
5490 022232 004737 032660      JSR  PC,$DB2D     ;CONVERT THE SEEK COUNT  
5491 022236 004737 027466      JSR  PC,$SUPRS    ;PRINT IT  
5492 022242 104414 051326      DISPLY ,LIN7M     ;' TOTAL MISPOS ERR = '  
5493 022246 016046 000066      MOV  $MISPO(RO),-(SP) ;TOTAL ERRORS  
5494 022252 004737 022362      JSR  PC,LINDEC    ;TYPE IT IN DECIMAL  
5495 022256 104414 051404      DISPLY ,LIN7S     ;' TOTAL SKI,OCYL ERR = '  
5496 022262 016046 000064      MOV  $SKI(RO),-(SP) ;CONVERT & PRINT IT  
5497 022266 004737 022362      JSR  PC,LINDEC    ;TYPE IT IN DECIMAL  
5498 022272 104414 001165      DISPLY ,SCLF      ;  
5499 022276 104414 001165      DISPLY ,SCLF      ;  
5500 022302 032777 100000 156630 BIT  #SW15,@SWR    ;SEE IF HALT ON ERROR - SWITCH 15 SET  
5501 022310 001401              BEQ  1$           ;BR IF NOT  
5502 022312 000000              HALT              ;SWITCH 15 HALT  
5503 022314 000207      1$: RTS  PC  
5504  
5505      ;PRINT LINE 8 OF THE ERROR MESSAGE  
5506      ;'DIFFERENT ERROR DURING RETRY'  
5507  
5508 022316 104414 051476      LINE8: DISPLY ,LIN8M ;  
5509 022322 004737 020332      JSR  PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE  
5510 022326 000207      RTS  PC  
5511  
5512      ;OCTAL TYPEOUT ROUTINE  
5513      ;CALL:  
5514      ;   MOV  NUM,-(SP) ;PUT THE NUMBER ON THE STACK  
5515      ;   JSR  PC,LINOCT  
5516      ;   RETURN  
5517
```


5518	022330	016646	000002		LINOCT: MOV	2(SP),-(SP)	:PUT NUMBER IN PROPER LOCATION ON STACK	
5519	022334	004737	030116			JSR	PC,\$SB20	:CONVERT THE NUMBER TO OCTAL
5520	022340	012637	022354			MOV	(SP)+,1\$:GET THE ADDRESS OF THE ASCII STRING
5521	022344	062737	000005	022354		ADD	#5.,1\$:ADDRESS THE LAST 6 ASCII DIGITS
5522	022352	104414				DISPLY		:TYPE IT
5523	022354	000000			1\$:	.WORD	0	:ADDRESS
5524	022356	012616				MOV	(SP)+,(SP)	:CORRECT THE STACK
5525	022360	000207				RTS	PC	:RETURN


```
5526  
5527 ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH  
5528 ;LEADING ZERO SUPPRESSION  
5529 ;CALL:  
5530 ;      MOV      NUM,-(SP)      ;PUT THE NUMBER ON THE STACK  
5531 ;      JSR      PC,LINDEC  
5532 ;      RETURN  
5533  
5534 022362 016646 000002 LINDEC: MOV      2(SP),-(SP)      ;SET UP STACK FOR CONVERT  
5535 022366 004737 030066      JSR      PC,$SB2D      ;CONVERT IT TO DECIMAL  
5536 022372 004737 027466      JSR      PC,$SUPRS      ;TYPE IT (WITH LEADING ZEROS SUPRESSED)  
5537 022376 012616      MOV      (SP)+,(SP)      ;RESTORE STACK POINTER  
5538 022400 000207      RTS      PC  
5539  
5540 ;:*****  
5541  
5542 .SBTTL GENERAL SUPPORT SUBROUTINES  
5543 ;:*****  
5544  
5545 ;DECREMENT THE SECTOR-TRACK ADDRESS  
5546 ;CALL:  
5547 ;      MOV      #DPB,R0      ;DPB ADDRESS  
5548 ;      JSR      PC,READDR  
5549 ;      RETURN  
5550 ;      (SP) CONTAINS THE TRACK ADDRESS  
5551 ;      2(SP) CONTAINS THE SECTOR ADDRESS  
5552  
5553  
5554 022402 162706 000004 READDR: SUB      #4,SP      ;DECREMENT THE STACK POINTER  
5555 022406 016616 000004      MOV      4(SP),(SP)      ;MOVE THE RETURN ADDR DOWN THE STACK  
5556 022412 005066 000004      CLR      4(SP)      ;CLEAR STACK FOR SECTOR  
5557 022416 005066 000002      CLR      2(SP)      ;CLEAR STACK FOR TRACK  
5558 022422 116066 000242 000004      MOVB     $RPDA(R0),4(SP) ;INCREMENTED SECTOR ON STACK  
5559 022430 005366 000004      DEC      4(SP)      ;DECREMENT THE SECTOR ADDRESS  
5560 022434 100015      BPL      1$      ;BR IF SECTOR GREATER THAN 0  
5561 022436 012766 000025 000004      MOV      #21,4(SP)      ;JAM SECTOR ADDRESS TO 21(10)  
5562 022444 116066 000243 000002      MOVB     $RPDA+1(R0),2(SP) ;TRACK ADDRESS  
5563 022452 005366 000002      DEC      2(SP)      ;DECREMENT TRACK ADDRESS  
5564 022456 100007      BPL      2$      ;BR IF IT DIDN'T GO NEG  
5565 022460 012766 000022 000002      MOV      #18,2(SP)      ;RESET TRACK TO 18(10)  
5566 022466 000403      BR      2$  
5567 022470 116066 000243 000002 1$: MOVB     $RPDA+1(R0),2(SP) ;TRACK ADDRESS  
5568 022476 000207      2$: RTS      PC      ;RETURN  
5569  
5570 ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS  
5571  
5572 022500 012737 177777 001210 CKCLK: MOV      #-1,CLKFLG      ;CLEAR CLOCK AVAILABILITY FLAG  
5573 022506 012737 177777 001206      MOV      #-1,PCLOCK      ;CLEAR KW11-P CLOCK AVAILABILITY FLAG  
5574 022514 012737 022574 000004      MOV      #CKCLK1,ERRVEC      ;SET UP VECTOR FOR CLOCK CHECK  
5575 022522 005037 000006      CLR      @ERRVEC+2      ;NEW PSW  
5576 022526 005777 156442      TST     @$LKCSR      ;CHECK FOR KW11-P  
5577 022532 005037 001210      CLR      CLKFLG      ;SET CLOCK AVAILABILITY FLAG  
5578 022536 005037 001206      CLR      PCLOCK      ;SET KW11-P CLOCK FLAG  
5579 022542 013701 001200      MOV      $LPVEC,R1      ;KW11-P VECTOR ADDRESS  
5580 022546 012721 023634      MOV      #CLOCK,(R1)+      ;SET UP KW11-P VECTOR  
5581 022552 012711 000300      MOV      #300,(R1)      ;PSW - PRI 6
```



```
5582 022556 012777 174575 156412      MOV      #-1667.,@%LKCSB ;LOAD COUNTER BUFFER WITH 16.67
5583 022564 012777 000131 156402      MOV      #131,@%LKCSR  ;SET CLOCK - CNT UP, 10US, CONT INT
5584 022572 000437                BR      CKCLK3
5585 022574 062706 000004                CKCLK1: ADD     #4,SP      ;RESTORE THE STACK POINTER
5586 022600 012737 022642 000004      MOV      #CKCLK2,@#ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
5587 022606 005777 156370                TST     @%LKS          ;LOOK FOR KW11-L
5588 022612 005037 001210                CLR     CLKFLG        ;SET CLOCK FLAG
5589 022616 013701 001204                MOV     $LLVEC,R1     ;KW11-L VECTOR ADDRESS
5590 022622 012721 023634                MOV     #CLOCK,(R1)+  ;SET UP KW11-L VECTOR
5591 022626 012711 000300                MOV     #300,(R1)    ;PSW - PRI 6
5592 022632 012777 000100 156342      MOV     #100,@%LKS    ;SET KW11-L INTERRUPT
5593 022640 000414                BR      CKCLK3
5594 022642 062706 000004                CKCLK2: ADD     #4,SP      ;RESTORE THE STACK POINTER
5595 022646 104401 053223                TYPE   ,NEDCLK        ;'P OR L CLOCK MUST BE ON SYSTEM'
5596 022652 005737 000042                TST     42            ;UNDER MONITOR CONTROL ?
5597 022656 001402                BEQ     1$            ;BR IF NOT
5598 022660 000137 005432                JMP     $GET42        ;ABORT PROGRAM
5599 022664 000000 1$:                HALT
5600 022666 000137 004122                JMP     START         ;TRY AGAIN
5601 022672 012737 000006 000004      CKCLK3: MOV     #6,@#ERRVEC ;RESTORE THE ERROR VECTOR
5602 022700 000207                RTS     PC
5603
5604                ;ROUTINE TO DISPLAY STATISTICS FOR ALL DRIVES ASSIGNED
5605                ;CALL:
5606                ;      JSR     PC,STATPR
5607                ;      RETURN
5608
5609 022702 010046                STATPR: MOV     R0,-(SP)   ;SAVE R0
5610 022704 010446                MOV     R4,-(SP)   ;SAVE R4
5611 022706 005737 001462                TST     ASNLST      ;ANY DRIVES ASSIGNED ?
5612 022712 001421                BEQ     3$          ;BR IF NOT
5613 022714 004737 023012                JSR     PC,SHDTYP   ;TYPE THE HEADING
5614 022720 005004                CLR     R4          ;CLEAR THE DRIVE INDEX
5615 022722 006304                1$:     ASL     R4          ;CHANGE TO INDEX WORDS
5616 022724 016400 001740                MOV     BLKADR(R4),R0 ;GET THE DRIVE'S BLOCK ADDRESS
5617 022730 006204                ASR     R4          ;RESTORE R4
5618 022732 136437 033416 001462      BITB   ATABIT(R4),ASNLST ;IS THIS DRIVE ASSIGNED ?
5619 022740 001402                BEQ     2$          ;BR IF NOT
5620 022742 004737 023034                JSR     PC,SDETAL   ;TYPE THE PERFORMANCE SUMMARY
5621 022746 005204                2$:     INC     R4          ;INCREMENT THE INDEX
5622 022750 020427 000010                CMP     R4,#8.      ;FINISHED ?
5623 022754 001362                BNE     1$          ;BR IF NOT
5624 022756 012604                3$:     MOV     (SP)+,R4  ;RESTORE R4
5625 022760 012600                MOV     (SP)+,R0  ;RESTORE R0
5626 022762 000207                RTS     PC          ;RETURN
5627
5628                ;ROUTINE TO TYPE STATISTICS FOR AN INDIVIDUAL DRIVE
5629                ;CALL:
5630                ;      MOV     #DPB,R0      ;DPB ADDRESS
5631                ;      JSR     PC,TYPEST
5632                ;      RETURN
5633
5634 022764 010046                TYPEST: MOV     R0,-(SP)   ;SAVE R0
5635 022766 010446                MOV     R4,-(SP)   ;SAVE R4
5636 022770 004737 023012                JSR     PC,SHDTYP   ;TYPE THE HEADING
5637 022774 005004                CLR     R4          ;CLEAR R4 FOR DRIVE NUMBER
```



```

5638 022776 111004          MOVB   (R0),R4          ;DRIVE NUMBER
5639 023000 004737 023034 JSR    PC,SDETAL      ;TYPE THE STATISTICS
5640 023004 012604          MOV    (SP)+,R4       ;RESTORE R4
5641 023006 012600          MOV    (SP)+,R0       ;RESTORE R0
5642 023010 000207          RTS     PC            ;RETURN
5643
5644          ;TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT
5645          ;CALL:
5646          ;      JSR    PC,SHDTYP
5647          ;      RETURN
5648
5649 023012 004737 023536 SHDTYP: JSR    PC,$TIME   ;TYPE THE TIME OF DAY
5650 023016 004537 027526 JSR    R5,TYPRI4      ;TYPE AT PRIORITY 4
5651 023022 001165          $CRLF                ;CR-LF
5652 023024 004537 027526 JSR    R5,TYPRI4      ;TYPE THE HEADER
5653 023030 052673          STATHD              ;HEADER
5654 023032 000207          RTS     PC            ;RETURN
5655
5656          ;TYPE THE PERFORMANCE SUMMARY DATE LINE
5657          ;CALL:
5658          ;      MOV    #DRIVE,R4      ;DRIVE NUMBER
5659          ;      MOV    #DPB,R0        ;DPB ADDRESS
5660          ;      RETURN
5661
5662 023034 010246          SDETAL: MOV    R2,-(SP)   ;SAVE R2
5663 023036 010002          MOV    R0,R2         ;DPB ADDRESS
5664 023040 062702 000036 ADD    #SOPERC,R2     ;FIRST STATISTICAL FIELD
5665 023044 010446          MOV    R4,-(SP)     ;;SAVE R4 FOR TYPEOUT
5666          ;      TYPE DRIVE NUMBER
5667 023046 104403          TYP0S                ;;GO TYPE--OCTAL ASCII
5668 023050 002          .BYTE 2            ;;TYPE 2 DIGIT(S)
5669 023051 000          .BYTE 0            ;;SUPPRESS LEADING ZEROS
5670 023052 104401 052323 TYPE    ,LINS          ;SPACES
5671 023056 016046 000070 MOV    $PASSC(R0),-(SP) ;PUT THE PASS COUNT ON THE STACK
5672 023062 004737 030066 JSR    PC,$SB2D       ;CONVERT IT
5673 023066 004537 027376 JSR    R5,REPLZ       ;TYPE IT
5674 023072 000003          .WORD 3             ;TYPE 3 DIGITS
5675 023074 104401 052323 TYPE    ,LINS          ;SPACES
5676 023100 010246          MOV    R2,-(SP)     ;PUT $OPERC ON THE STACK
5677 023102 004737 032660 JSR    PC,$DB2D       ;CONVERT IT
5678 023106 004537 027376 JSR    R5,REPLZ       ;TYPE $OPERC
5679 023112 000006          .WORD 6             ;TYPE 6 DIGITS
5680 023114 104401 052323 TYPE    ,LINS          ;SPACES
5681 023120 062702 000004 ADD    #4,R2          ;INCREMENT R2
5682 023124 010246          MOV    R2,-(SP)     ;PUT $POSIT ON THE STACK
5683 023126 004737 032660 JSR    PC,$DB2D       ;CONVERT IT
5684 023132 004537 027376 JSR    R5,REPLZ       ;TYPE $POSIT
5685 023136 000006          .WORD 6             ;TYPE 6 DIGITS
5686 023140 104401 052323 TYPE    ,LINS          ;SPACES
5687 023144 062702 000004 ADD    #4,R2          ;INCREMENT R2
5688 023150 010246          MOV    R2,-(SP)     ;PUT $STRANS ON THE STACK
5689 023152 004737 032660 JSR    PC,$DB2D       ;CONVERT $STRANS
5690 023156 004537 027376 JSR    R5,REPLZ       ;TYPE IT
5691 023162 000012          .WORD 10            ;TYPE 10 DIGITS
5692 023164 104401 052323 TYPE    ,LINS          ;SPACES
5693 023170 062702 000004 ADD    #4,R2          ;INCREMENT R2

```



```
5694 023174 010246      MOV      R2,-(SP)      ;PUT $READ ON THE STACK
5695 023176 004737 032660 JSR      PC,$SB2D      ;CONVERT $READ
5696 023202 004537 027376 JSR      R5,REPLZ      ;TYPE IT
5697 023206 000012      .WORD    10           ;TYPE 10 DIGITS
5698 023210 104401 052324 TYPE     ,LINSPO       ;1 SPACE
5699 023214 062702 000004 ADD      #4,R2         ;INCREMENT R2
5700 023220 062702 000002 ADD      #2,R2         ;INCREMENT R2 AGAIN
5701 023224 012246      MOV      (R2)+,-(SP)   ;PUT $SOFT ON THE STACK
5702 023226 004737 030066 JSR      PC,$SB2D      ;CONVERT $SOFT
5703 023232 004537 027376 JSR      R5,REPLZ      ;TYPEOUT $SOFT
5704 023236 000004      .WORD    4           ;TYPE 4 DIGITS
5705 023240 104401 052324 TYPE     ,LINSPO       ;SPACES
5706 023244 012246      MOV      (R2)+,-(SP)   ;PUT $HARD ON THE STACK
5707 023246 004737 030066 JSR      PC,$SB2D      ;CONVERT $HARD
5708 023252 004537 027376 JSR      R5,REPLZ      ;TYPEOUT $HARD
5709 023256 000004      .WORD    4           ;TYPE 4 DIGITS
5710 023260 104401 052324 TYPE     ,LINSPO       ;SPACES
5711 023264 012246      MOV      (R2)+,-(SP)   ;PUT $SKI ON THE STACK
5712 023266 004737 030066 JSR      PC,$SB2D      ;CONVERT $SKI
5713 023272 004537 027376 JSR      R5,REPLZ      ;TYPEOUT $SKI
5714 023276 000004      .WORD    4           ;TYPE 4 DIGITS
5715 023300 104401 052324 TYPE     ,LINSPO       ;SPACES
5716 023304 012246      MOV      (R2)+,-(SP)   ;PUT $MISPO ON THE STACK
5717 023306 004737 030066 JSR      PC,$SB2D      ;CONVERT $MISPO
5718 023312 004537 027376 JSR      R5,REPLZ      ;TYPEOUT $MISPO
5719 023316 000004      .WORD    4           ;TYPE 4 DIGITS
5720 023320 104401 052324 TYPE     ,LINSPO       ;SPACES
5721 023324 016046 000056 MOV      $TOTAL(R0),-(SP) ;CALCULATE NUMBER OF OTHER ERRORS
5722 023330 166016 000060 SUB      $SOFT(R0),(SP) ;SUBTRACT $SOFT FROM $TOTAL
5723 023334 166016 000062 SUB      $HARD(R0),(SP) ;SUBTRACT $HARD FROM $TOTAL
5724 023340 166016 000064 SUB      $SKI(R0),(SP)  ;SUBTRACT $SKI FROM $TOTAL
5725 023344 166016 000066 SUB      $MISPO(R0),(SP) ;SUBTRACT $MISPO FROM $TOTAL
5726 023350 004737 030066 JSR      PC,$SB2D      ;CONVERT 'OTHER' COUNT
5727 023354 004537 027376 JSR      R5,REPLZ      ;TYPE IT
5728 023360 000004      .WORD    4           ;TYPE 4 DIGITS
5729 023362 104401 001165 TYPE     ,%CRLF        ;
5730 023366 012602      MOV      (SP)+,R2      ;RESTORE R2
5731 023370 000207      RTS      PC
```

```
5732
5733
5734
5735      ;ROUTINE TO INCREMENT $SOFT
5736      ;
5737      ;NOTE: $SOFT WILL NOT BE INCREMENTED BEYOND 9999 (10)
5738
5739 023372 005737 001264 INCSOF: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
5740 023376 001006      BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5741 023400 026027 000060 023417 CMP      $SOFT(R0),#9999.  ;IS $SOFT ALREADY AT MAXIMUM ?
5742 023406 103002      BHIS     1$              ;BR IF IT IS
5743 023410 005260 000060      INC      $SOFT(R0)      ;INCREMENT $SOFT
5744 023414 000207 1$:      RTS      PC          ;RETURN
5745
5746
5747      ;ROUTINE TO INCREMENT $HARD
5748      ;
5749      ;NOTE: $HARD WILL NOT BE INCREMENTED BEYOND 9999 (10)
```



```
5750
5751 023416 005737 001264      INCHRD: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
5752 023422 001006              BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5753 023424 026027 000062 023417      CMP      $SHARD(RO),#9999. ;IS $SHARD ALREADY AT MAXIMUM ?
5754 023432 103002              BHIS     1$              ;BR IF IT IS
5755 023434 005260 000062      INC      $SHARD(RO)      ;INCREMENT $SHARD
5756 023440 000207      1$:      RTS      PC      ;RETURN
5757
5758
5759      ;ROUTINE TO INCREMENT $SKI
5760      ;
5761      ;NOTE: $SKI WILL NOT BE INCREMENTED BEYOND 9999 (10)
5762
5763 023442 005737 001264      INCSKI: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
5764 023446 001006              BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5765 023450 026027 000064 023417      CMP      $SKI(RO),#9999. ;IS $SKI ALREADY AT MAXIMUM ?
5766 023456 103002              BHIS     1$              ;BR IF IT IS
5767 023460 005260 000064      INC      $SKI(RO)        ;INCREMENT $SKI
5768 023464 000207      1$:      RTS      PC      ;RETURN
5769
5770
5771      ;ROUTINE TO INCREMENT $MISPO
5772      ;
5773      ;NOTE: $MISPO WILL NOT BE INCREMENTED BEYOND 9999 (10)
5774
5775 023466 005737 001264      INCMIS: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
5776 023472 001006              BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5777 023474 026027 000066 023417      CMP      $MISPO(RO),#9999. ;IS $MISPO ALREADY AT MAXIMUM ?
5778 023502 103002              BHIS     1$              ;BR IF IT IS
5779 023504 005260 000066      INC      $MISPO(RO)     ;INCREMENT $MISPO
5780 023510 000207      1$:      RTS      PC      ;RETURN
5781
5782
5783      ;ROUTINE TO INCREMENT $TOTAL
5784      ;
5785      ;NOTE: $TOTAL WILL NOT BE INCREMENTED BEYOND 9999 (10)
5786
5787 023512 005737 001264      INCTOT: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
5788 023516 001006              BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5789 023520 026027 000056 023417      CMP      $TOTAL(RO),#9999. ;IS $TOTAL ALREADY AT MAXIMUM ?
5790 023526 103002              BHIS     1$              ;BR IF IT IS
5791 023530 005260 000056      INC      $TOTAL(RO)     ;INCREMENT $TOTAL
5792 023534 000207      1$:      RTS      PC      ;RETURN
5793
5794
5795      ;ROUTINE TO TYPE THE TIME
5796
5797 023536 005737 001210      $TIME: TST      CLKFLG      ;CLOCK ON THE SYSTEM ?
5798 023542 001033              BNE      1$              ;BR IF NOT
5799 023544 104401 001165      TYPE     , $CRLF        ;CR-LF
5800 023550 013746 001266      MOV      HOUR,-(SP)     ;PUT 'HOURS' ON THE STACK
5801 023554 004737 030066      JSR      PC,$SB2D       ;CONVERT TO DECIMAL
5802 023560 004537 027376      JSR      R5,REPLZ      ;TYPE IT
5803 023564 000002              .WORD    2              ;TYPE 2 DIGITS
5804 023566 104401 053477      TYPE     ,COLON        ;':'
5805 023572 013746 001270      MOV      MINUTE,-(SP)   ;PUT 'MINUTES' ON THE STACK
```



```
5806 023576 004737 030066      JSR    PC,$SB2D      ;CONVERT TO DECIMAL
5807 023602 004537 027376      JSR    R5,REPLZ     ;TYPE IT
5808 023606 000002              .WORD    2          ;TYPE 2 DIGITS
5809 023610 104401 053477      TYPE    ,COLON      ;':'
5810 023614 013746 001272      MOV    SECOND,-(SP) ;PUT SECONDS ON THE STACK
5811 023620 004737 030066      JSR    PC,$SB2D     ;CONVERT TO DECIMAL
5812 023624 004537 027376      JSR    R5,REPLZ     ;TYPE IT
5813 023630 000002              .WORD    2          ;TYPE 2 DIGITS
5814 023632 000207      1$:   RTS    PC
5815
5816      ;CLOCK HANDLER ROUTINE
5817
5818 023634 005337 001274      CLOCK: DEC    SIXTEE      ;INCREMENT THE 1/60 SECOND COUNTER
5819 023640 001035              BNE    1$          ;BR IF A SECOND NOT COUNTED
5820 023642 013737 001212 001274      MOV    HZ,SIXTEE    ;RESTORE THE VALUE
5821 023650 005237 001272              INC    SECOND       ;COUNT THE SECOND
5822 023654 022737 000074 001272      CMP    #60.,SECOND ;AT MAXIMUM ?
5823 023662 001024              BNE    1$          ;BR IF NOT
5824 023664 005037 001272      CLR    SECOND       ;CLEAR THE SECOND'S COUNTER
5825 023670 005237 001412              INC    INTRVL+2     ;COUNT THE PERFORMANCE SUMMARY INTERVAL
5826 023674 005237 001270              INC    MINUTE       ;COUNT THE MINUTE
5827 023700 022737 000074 001270      CMP    #60.,MINUTE ;AT MAXIMUM ?
5828 023706 001012              BNE    1$          ;BR IF NOT
5829 023710 005037 001270              CLR    MINUTE       ;CLEAR THE MINUTE'S COUNTER
5830 023714 005237 001266              INC    HOUR         ;COUNT THE HOURS
5831 023720 022737 001747 001266      CMP    #999.,HOUR   ;AT MAXIMUM
5832 023726 103002              BHS    1$          ;BR IF NOT
5833 023730 005037 001266              CLR    HOUR         ;CLEAR THE HOURS
5834 023734 012746 000021      1$:   MOV    #17.,-(SP) ;17 MS ON THE STACK
5835 023740 004737 040026      JSR    PC,RPTMR     ;DRIVER TIMER ROUTINE
5836 023744 005737 001410              TST    INTRVL       ;DISPLAY THE PERFORMANCE SUMMARY ?
5837 023750 001411              BEQ    2$          ;BR IF NOT
5838 023752 023737 001410 001412      CMP    INTRVL,INTRVL+2 ;DISPLAY INTERVAL FINISHED ?
5839 023760 001005              BNE    2$          ;BR IF NOT
5840 023762 012737 177777 001214      MOV    #-1,STATIN  ;SET PERFORMANCE SUMMARY DISPLAY FLAG
5841 023770 005037 001412              CLR    INTRVL+2    ;CLEAR THE PERFORMANCE INTERVAL COUNTER
5842 023774 000002      2$:   RTI
5843
5844      ;COMMAND DECODE ROUTINE
5845      ;CALL:
5846      ;      MOV    #-1,CFLAG ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
5847      ;      ;          ;ROUTINE IN INTERRUPT MODE
5848      ;      JSR    PC,KSR
5849      ;      RETURN1 ;SYSTEM BUSY RETURN
5850      ;      RETURN2 ;RETURN AFTER KEYBOARD SERVICED
5851
5852 023776 005737 001440      KSR:  TST    ORDERQ    ;ANY OPERATIONS ACTIVE ?
5853 024002 001003              BNE    1$          ;BR IF SOME ARE
5854 024004 005037 024030              CLR    3$          ;CLEAR THE LOOP COUNTER
5855 024010 000410              BR     KSR1        ;PROCESS THE KEYBOARD REQUEST
5856 024012 062737 000001 024030      1$:   ADD    #1,3$      ;COUNT THE TIMES THROUGH THE LOOP
5857 024020 001002              BNE    2$          ;BR IF NOT ENOUGH
5858 024022 104401 053610              TYPE    ,BUSY      ;'SYSTEM BUSY...'
5859 024026 000207      2$:   RTS    PC        ;PROCESS ANY COMPLETED DRIVES
5860 024030 000000      3$:   .WORD    0      ;LOOP COUNTER
5861 024032 104412      KSR1: SAVREG      ;SAVE THE REGISTERS
```



```

5862 024034 012737 000200 177776      MOV      #PR4,PS      ;SET PRIORITY TO 4
5863 024042 005037 001262      CLR      CFLAG      ;CLEAR THE 'CONTROL C' FLAG
5864 024046 004737 023536      JSR      PC,$TIME    ;TYPE THE TIME
5865 024052 005777 155070      TST      @TKB        ;CLEAR ANY GARBAGE IN THE TTY BUFFER
5866 024056 104401 053323      TYPE     ,ENTCOM     ;'ENTER COMMANDS'
5867 024062 104411      RDLIN      ;READ THE KEYBOARD
5868 024064 012605      MOV      (SP)+,R5    ;GET ADDRESS OF INPUT STRING
5869 024066 005737 001262      TST      CFLAG      ;CHECK THE CONTROL C FLAG
5870 024072 001065      BNE      7$         ;EXIT IF 'CONTROL C' ENTERED
5871 024074 005205      INC      R5         ;POINT TO SECOND CHARACTER
5872 024076 122715 000101      CMPB     #'A',(R5)   ;EQ TO AN 'A'
5873 024102 001410      BEQ      1$         ;BR IF IT IS
5874 024104 121527 000067      CMPB     (R5),#'7    ;DRIVE NUMBER GREATER THAN AN ASCII 7 ?
5875 024110 101054      BHI      6$         ;BR IF IT IS
5876 024112 121527 000060      CMPB     (R5),#'0    ;DRIVE NUMBER LESS THAN AN ASCII 0 ?
5877 024116 103451      BLO      6$         ;BR IF IT IS
5878 024120 142715 177770      BICB     #'^C7,(R5) ;LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
5879 024124 122765 000124 177777 1$:  CMPB     #'T,-1(R5) ;EQ TO 'T'
5880 024132 001003      BNE      2$         ;BR IF NOT EQ
5881 024134 004737 024640      JSR      PC,NEWASN   ;ASSIGN DRIVE FOR TEST
5882 024140 000442      BR       7$         ;EXIT
5883 024142 122765 000104 177777 2$:  CMPB     #'D,-1(R5) ;EQ TO 'D' ?
5884 024150 001003      BNE      3$         ;BR IF NOT EQ
5885 024152 004737 024650      JSR      PC,DEASGN   ;DEASSIGN DRIVE
5886 024156 000433      BR       7$         ;EXIT
5887 024160 122765 000123 177777 3$:  CMPB     #'S,-1(R5) ;EQ TO 'S'
5888 024166 001003      BNE      4$         ;BR IF NOT EQ
5889 024170 004737 024756      JSR      PC,SCMND    ;TYPE STATISTICS
5890 024174 000424      BR       7$         ;EXIT
5891 024176 122765 000127 177777 4$:  CMPB     #'W,-1(R5) ;EQ TO 'W'
5892 024204 001007      BNE      5$         ;BR IF NOT EQ
5893 024206 032777 000001 154724      BIT      #SW0,@SWR   ;IS SWITCH 0 SET ?
5894 024214 001012      BNE      6$         ;BR IF SET, CAN'T DO 'W' COMMAND
5895 024216 004737 025226      JSR      PC,DATAPK   ;WRITE A DATA PACK
5896 024222 000411      BR       7$         ;EXIT
5897 024224 122765 000122 177777 5$:  CMPB     #'R,-1(R5) ;EQ TO 'R' ?
5898 024232 001003      BNE      6$         ;BR IF NOT EQ
5899 024234 004737 025240      JSR      PC,REDAPK   ;READ A DATA PACK
5900 024240 000402      BR       7$         ;EXIT
5901 024242 104401 053301      6$:     TYPE     ,INVLD ;TYPE 'INVALID COMMAND' MESSAGE
5902 024246 104413      7$:     RESREG    ;RESTORE R0 - R5
5903 024250 062716 000002      ADD      #2,(SP)    ;INCREMENT THE RETURN ADDRESS
5904 024254 005777 154666      TST      @TKB        ;CLEAR THE TTY BUFFER
5905 024260 052777 000100 154656      BIS      #BIT06,@TKS ;SET TTY INTERRUPT ENABLE
5906 024266 005037 177776      CLR      PS         ;SET PRIORITY BACK TO ZERO
5907 024272 000207      RTS      PC         ;RETURN
5908
5909      ;ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)
5910
5911 024274 122715 000101      ASSIGN: CMPB     #'A,(R5) ;ASSIGN ALL DRIVES?
5912 024300 001430      BEQ      ASGN2      ;BR IF ALL DRIVES
5913 024302 111504      ASGN1: MOV      (R5),R4    ;PUT DRIVE # IN R4
5914 024304 012737 052521 026534      MOV      #UNTASN,ASNMSG ;'DRIVE ASSIGNED' MESSAGE ADDRESS
5915 024312 012703 000001      MOV      #1,R3      ;RELOAD R3 FOR 1 UNIT
5916 024316 136437 033416 001462      BITB     ATABIT(R4),ASNLS1 ;DRIVE ALREADY ASSIGNED ?
5917 024324 001013      BNE      2$         ;BR IF IT IS

```


5918	024326	005704				TST	R4		: TRYING TO ASSIGN DRIVE 0 ?
5919	024330	001007				BNE	1\$: BR IF NOT
5920	024332	012737	052605	026534		MOV	#NOTAVL,ASNMSG		: 'NOT AVAILABLE' MESSAGE ADDRESS
5921	024340	122737	000011	000041		CMPB	#11,41		: SEE IF LOADED FROM AN RP04/5/6
5922	024346	001402				BEQ	2\$: BR IF RP04/5/6 IS THE LOAD DEVICE
5923	024350	004737	024434		1\$:	JSR	PC,ASGN3		: SEE IF DRIVE ON THE SYSTEM
5924	024354	000137	026514		2\$:	JMP	ASNERR		: RETURN HERE IF DRIVE NOT AVAIL
5925	024360	000207				RTS	PC		: EXIT
5926	024362	122737	000011	000041	ASGN2:	CMPB	#11,41		: LOADED FROM AN RP04/5/6 ?
5927	024370	001005				BNE	1\$: BR IF NOT
5928	024372	012704	000001			MOV	#1,R4		: START WITH DRIVE 1
5929	024376	012703	000007			MOV	#7.,R3		: SETUP FOR ONLY 7 DRIVES
5930	024402	000403				BR	2\$: CONTINUE
5931	024404	005004			1\$:	CLR	R4		: START WITH DRIVE 0
5932	024406	012703	000010			MOV	#8.,R3		: DRIVE COUNT FOR 8 DRIVES
5933	024412	004737	024434		2\$:	JSR	PC,ASGN3		: ASSIGN ALL UNASSIGNED, AVAIL DRIVES
5934	024416	000137	024424		3\$:	JMP	4\$: DRIVE NOT ON SYSTEM
5935	024422	000207				RTS	PC		: RETURN
5936	024424	012746	024416		4\$:	MOV	#3\$,-(SP)		: PUT RETURN ADDRESS ON THE STACK
5937	024430	000137	024552			JMP	ASGN4		: LOOK FOR MORE DRIVES
5938	024434	136437	033416	001462	ASGN3:	BITB	ATABIT(R4),ASNLS		: DRIVE ALREADY ASSIGNED ?
5939	024442	001043				BNE	ASGN4		: BR IF IT IS
5940	024444	005737	033414		1\$:	TST	DTUW		: DATA TRANSFER UNDER WAY ?
5941	024450	100375				BPL	1\$: BR IF IT IS
5942	024452	110437	045076			MOVB	R4,GENDPB		: DRIVE NUMBER
5943	024456	004737	015332			JSR	PC,RECALO		: RECALIBRATE DRIVE
5944	024462	105764	033302			TSTB	DRVSTA(R4)		: DRIVE AVAILABLE?
5945	024466	001444				BEQ	ASGN7		: BR IF DRIVE OFFLINE OR NONEXISTENT
5946	024470	100437				BMI	ASGN6		: BR IF DRIVE UNSAFE
5947	024472	006304				ASL	R4		: MAKE R4 INTO WORD INDEX
5948	024474	016464	001740	001506		MOV	BLKADR(R4),NEWUNT(R4)		: DPB ADDRESS
5949	024502	016400	001740			MOV	BLKADR(R4),R0		: PUT BLOCK'S ADDR INTO R0
5950	024506	004737	025252			JSR	PC,CLRDPB		: CLEAR BLOCK FOR DRIVE JUST ASSIGNED
5951	024512	004737	025456			JSR	PC,DRVPRM		: GET THE DRIVE'S ADDRESS LIMITS
5952	024516	004737	025740			JSR	PC,GETID		: GET DRIVE I.D.
5953	024522	004737	026050			JSR	PC,GETADR		: GET BAD SECTOR ADDRESSES
5954	024526	012760	000001	000070		MOV	#1,\$PASSC(R0)		: PRESET PASS COUNT TO 1
5955	024534	005737	001216			TST	PACK		: WRITE DATA PACK ?
5956	024540	001403				BEQ	2\$: BR IF NOT
5957	024542	113760	001216	000026		MOVB	PACK,\$PACK(R0)		: SET READ/WRITE DATA PACK INDICATOR
5958	024550	006204			2\$:	ASR	R4		: RESTORE DRIVE ADDRESS
5959	024552	005303			ASGN4:	DEC	R3		: DECREMENT DRIVE COUNT
5960	024554	001402				BEQ	ASGN5		: BR IF FINISHED
5961	024556	005204				INC	R4		: INCREMENT DRIVE NUMBER
5962	024560	000725				BR	ASGN3		: CONTINUE
5963	024562	062716	000004		ASGN5:	ADD	#4,(SP)		: INCREMENT RETURN
5964	024566	000207				RTS	PC		: RETURN
5965	024570	012737	052624	026534	ASGN6:	MOV	#NOTSAF,ASNMSG		: 'UNSAFE' MESSAGE ADDRESS
5966	024576	000207				RTS	PC		: RETURN
5967	024600	105764	033312		ASGN7:	TSTB	DRVSTYP(R4)		: DRIVE PRESENT?
5968	024604	001405				BEQ	1\$: BR IF NOT
5969	024606	100010				BPL	2\$: BR IF DRIVE OFFLINE
5970	024610	012737	052547	026534		MOV	#NOTRP,ASNMSG		: ADDRESS OF 'NOT RP04/5/6' MSG
5971	024616	000407				BR	3\$: EXIT
5972	024620	012737	052570	026534	1\$:	MOV	#NOTPRS,ASNMSG		: ADDRESS OF 'NOT PRESENT' MSG
5973	024626	000403				BR	3\$: EXIT


```

5974 024630 012737 052456 026534 2$: MOV #UNTOFF,ASNMSG ;ADDRESS OF 'DRIVE OFFLINE' MESSAGE
5975 024636 000207 3$: RTS PC ;ERROR RETURN
5976
5977 ;'T' COMMAND (ROUTINE TO ASSIGN A DRIVE)
5978
5979 024640 005037 001216 NEWASN: CLR PACK ;CLEAR 'W' COMMAND INDICATOR
5980 024644 000137 024274 JMP ASSIGN ;GO TO THE ASSIGN ROUTINE
5981
5982 ;'D' COMMAND (ROUTINE TO DEASSIGN A DRIVE)
5983
5984 024650 005004 DEASGN: CLR R4
5985 024652 122715 000101 CMPB #'A',(R5) ;DEASSIGN ALL DRIVES ?
5986 024656 001434 BEQ 5$ ;BR IF YES
5987 024660 012703 000001 MOV #BIT00,R3 ;SET R3 FOR ONE UNIT
5988 024664 111504 MOV (R5),R4 ;GET DRIVE NUMBER
5989 024666 136437 033416 001462 1$: BITB ATABIT(R4),ASNLS ;DRIVE ASSIGNED ?
5990 024674 001414 BEQ 3$ ;BR IF NOT
5991 024676 146437 033416 001462 BICB ATABIT(R4),ASNLS ;DELETE THE DRIVE FROM THE ASSIGNED LIST
5992 024704 006304 ASL R4 ;MAKE ADDR INTO A WORD INDEX
5993 024706 016464 001740 001464 MOV BLKADR(R4),DUNIT(R4) ;PUT ADDRESS IN DEASSIGN LIST
5994 024714 006204 ASR R4
5995 024716 005303 2$: DEC R3 ;ANY MORE DRIVES ?
5996 024720 001412 BEQ 4$ ;BR IF NOT
5997 024722 005204 INC R4
5998 024724 000760 BR 1$
5999 024726 122715 000101 3$: CMPB #'A',(R5) ;DEASSIGN ALL DRIVES ?
6000 024732 001771 BEQ 2$ ;BR IF YES
6001 024734 012737 052477 026534 MOV #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MESSAGE
6002 024742 004737 026514 JSR PC,ASNERR ;REPORT IT
6003 024746 000207 4$: RTS PC
6004 024750 012703 000010 5$: MOV #8.,R3 ;SET UNIT COUNT TO 8
6005 024754 000744 BR 1$
6006
6007 ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)
6008
6009 024756 005004 SCMND: CLR R4
6010 024760 122715 000101 CMPB #'A',(R5) ;ALL STATISTICS ?
6011 024764 001421 BEQ 2$ ;BR IF YES
6012 024766 111504 MOV (R5),R4 ;GET DRIVE NUMBER
6013 024770 136437 033416 001462 BITB ATABIT(R4),ASNLS ;SEE IF DRIVE ASSIGNED
6014 024776 001406 BEQ 1$ ;BR IF NOT
6015 025000 006304 ASL R4 ;MAKE DRIVE ADDR INTO WORD INDEX
6016 025002 016400 001740 MOV BLKADR(R4),R0 ;ADDR OF BLOCK
6017 025006 004737 022764 JSR PC,TYPEST ;TYPE DRIVE STATISTICS
6018 025012 000504 BR 9$ ;EXIT
6019 025014 012737 052477 026534 1$: MOV #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MSG
6020 025022 004737 026514 JSR PC,ASNERR ;TYPE ERROR MESSAGE
6021 025026 000476 BR 9$ ;EXIT
6022 025030 012703 000010 2$: MOV #8.,R3 ;DRIVE COUNT
6023 025034 136437 033416 001462 3$: BITB ATABIT(R4),ASNLS ;SEE IF DRIVE ASSIGNED
6024 025042 001004 BNE 4$ ;BR IF YES
6025 025044 005204 INC R4 ;INCREMENT DRIVE ADDRESS
6026 025046 005303 DEC R3 ;DECREMENT COUNTER
6027 025050 001371 BNE 3$ ;MORE TO CHECK
6028 025052 000464 BR 9$ ;NONE ASSIGNED, RETURN
6029 025054 004737 022702 4$: JSR PC,STATPR ;TYPE ALL STATISTICS
  
```



```

6030 025060 105737 001220      TSTB  DATE          ;SEE IF 'DATE' ENTERED
6031 025064 001404              BEQ   11$           ;BR IF NOT
6032 025066 104401 053501      TYPE  ,DATEIS      ;'DATE:'
6033 025072 104401 001220      TYPE  ,DATE        ;THE OPERATOR ENTERED DATE
6034 025076 105737 001232      11$: TSTB  OPERID    ;SEE IF OPERATOR I.D. ENTERED
6035 025102 001404              BEQ   12$           ;BR IF NOT
6036 025104 104401 053512      TYPE  ,IDIS        ;'OPERATOR I.D.:'
6037 025110 104401 001232      TYPE  ,OPERID      ;THE OPERATOR I.D.
6038 025114 104401 053534      12$: TYPE  ,HEDLIN   ;HEADER LINE
6039 025120 012737 042262 025174 MOV   #DRIVE0+$DRVID,6$ ;DRIVE I.D. FIELD ADDRESS
6040 025126 005004              CLR   R4           ;DRIVE ADDRESS
6041 025130 012703 000010      MOV   #8.,R3       ;COUNTER
6042 025134 136437 033416 001462 5$: BITB  ATABIT(R4),ASNLST ;SEE IF DRIVE ASSIGNED
6043 025142 001417              BEQ   7$           ;BR IF NOT ASSIGNED
6044 025144 010446              MOV   R4,-(SP)     ;SAVE R4 FOR TYPEOUT
6045                                ;TYPE DRIVE NUMBER
6046 025146 104403              TYPOS                ;GO TYPE--OCTAL ASCII
6047 025150          002          .BYTE  2           ;TYPE 2 DIGIT(S)
6048 025151          000          .BYTE  0           ;SUPPRESS LEADING ZEROS
6049 025152 104401 052321      TYPE  ,LIN4SP      ;4 SPACES
6050 025156 105777 000012      TSTB  @6$          ;SEE IF DRIVE I.D. ENTERED
6051 025162 001003              BNE  10$          ;BR IF DRIVE I.D. PRESENT
6052 025164 104401 053557      TYPE  ,NONE        ;TYPE 'NONE'
6053 025170 000404              BR   7$           ;CONTINUE
6054 025172 104401              10$: TYPE  THE DRIVE I.D.
6055 025174 000000      6$: .WORD  0       ;ADDRESS OF DRIVE I.D. FIELD HERE
6056 025176 104401 001165      TYPE  ,%CRLF       ;CR-LF
6057 025202 005303      7$: DEC   R3       ;DECREMENT THE COUNTER
6058 025204 001405              BEQ   8$           ;BR IF AT END
6059 025206 062737 000304 025174 ADD   #RPEC2+2,6$   ;INCREMENT THE MESSAGE FIELD ADDRESS
6060 025214 005204              INC   R4           ;INCREMENT DRIVE ADDRESS
6061 025216 000746              BR   5$           ;CONTINUE
6062 025220 104401 001165      8$: TYPE  ,%CRLF   ;CR-LF
6063 025224 000207      9$: RTS   PC
6064
6065      ;'W' COMMAND (ROUTINE TO WRITE A DATA PACK)
6066
6067 025226 012737 177777 001216 DATAPK: MOV   #-1,PACK ;SET THE 'W' COMMAND INDICATOR
6068 025234 000137 024274              JMP   ASSIGN      ;ASSIGN REQUESTED DRIVE
6069
6070
6071      ;'R' COMMAND (ROUTINE TO READ A DATA PACK)
6072
6073 025240 012737 000001 001216 REDAPK: MOV   #1,PACK ;SET THE 'READ' INDICATOR
6074 025246 000137 024274              JMP   ASSIGN      ;ASSIGN THE REQUESTED DRIVE
6075
6076      ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
6077      ;CALL:
6078      ;     MOV   #DPB,R0 ;DPB ADDRESS
6079      ;     JSR   PC,CLRDPB
6080      ;     RETURN
6081
6082      CLRDPB:
6083 025252 010346              MOV   R3,-(SP)    ;PUSH R3 ON STACK
6084 025254 010446              MOV   R4,-(SP)    ;PUSH R4 ON STACK
6085 025256 010546              MOV   R5,-(SP)    ;PUSH R5 ON STACK
  
```



```

6086 025260 016046 000262      MOV      $RPDT(R0),-(SP) ;SAVE DRIVE TYPE (REVD)
6087 025264 010004      MOV      R0,R4 ;GET THE DPB ADDRESS
6088 025266 062704 000002      ADD      #2,R4 ;ADDRESS OF FIRST LOCN TO BE CLEARED
6089 025272 012703 000005      MOV      #5,R3 ;NUMBER OF LOCNS TO BE CLEARED
6090 025276 005024      1$: CLR      (R4)+ ;CLEAR THE LOCATION
6091 025300 005303      DEC      R3 ;DECREMENT THE COUNTER
6092 025302 001375      BNE      1$ ;BR IF NOT FINISHED
6093 025304 062704 000002      ADD      #2,R4 ;MOVE THE ADDRESS PAST THE 'REG' ADDR
6094 025310 012703 000070      MOV      #S$NEXT-$REG,R3 ;NUMBER OF LOCNS TO BE CLEARED
6095 025314 005024      2$: CLR      (R4)+ ;CLEAR
6096 025316 162703 000002      SUB      #2,R3 ;DECREMENT THE LOCN COUNTER
6097 025322 001374      BNE      2$ ;BR IF NOT FINISHED
6098 025324 062704 000014      ADD      #12.,R4 ;MOVE PAST ADDRESS LIMITS
6099 025330 012703 000162      MOV      #SRPEC2-MINSEC,R3 ;NUMBER OF LOCNS TO BE CLEARED
6100 025334 005024      3$: CLR      (R4)+ ;CLEAR A LOCATION
6101 025336 162703 000002      SUB      #2,R3 ;DECREMENT THE COUNTER
6102 025342 001374      BNE      3$ ;BR IF NOT DONE
6103 025344 113760 001434 000024      MOVB     BEGCOD,$CODE(R0) ;INITIAL COMMAND CODE
6104 025352 013701 001434      MOV      BEGCOD,R1 ;GET THE ACTUAL OP CODE
6105 025356 116160 001760 000002      MOVB     COMTBL(R1),$COMND(R0) ;OPERATION CODE
6106 025364 113760 001432 000030      MOVB     BEGPAT,$PATT(R0) ;PATTERN CODE
6107 025372 106360 000030      ASLB     $PATT(R0) ;CONVERT CODE TO A TABLE INDEX
6108 025376 013760 001436 000020      MOV      BEGSIZ,$WRDL(R0) ;BEGINNING RECORD SIZE
6109 025404 013760 001436 000004      MOV      BEGSIZ,$WRDM(R0) ;VALUE FOR DATA TRANSFER
6110 025412 005460 000004      NEG      $WRDM(R0) ;MAKE IT INTO 2'S COMPLEMENT
6111 025416 012760 000400 000022      MOV      #256.,$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
6112 025424 132760 000001 000024      BITB     #1,$CODE(R0) ;HEADER ORDER ?
6113 025432 001403      BEQ      4$ ;BR IF NOT
6114 025434 062760 000004 000022      ADD      #4,$SSEC(R0) ;ADD HEADER SIZE TO SECTOR SIZE
6115 025442 012660 000262      4$: MOV      (SP)+,$RPDT(R0) ;RESTORE DRIVE TYPE (REVD)
6116 025446 012605      MOV      (SP)+,R5 ;POP STACK INTO R5
6117 025450 012604      MOV      (SP)+,R4 ;POP STACK INTO R4
6118 025452 012603      MOV      (SP)+,R3 ;POP STACK INTO R3
6119 025454 000207      RTS      PC ;RETURN

```

;ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR

```

6120
6121
6122
6123 025456 010346      DRVPRM: MOV      R3,-(SP) ;SAVE R3
6124 025460 010446      MOV      R4,-(SP) ;SAVE R4
6125 025462 005737 000042      TST      42 ;RUNNING UNDER MONITOR CONTROL
6126 025466 001035      BNE      3$ ;BR IF YES
6127 025470 104401 053375      TYPE     ,ENTLMT ;'ENTER ADDRESSES'
6128 025474 006204      ASR      R4 ;CONVERT INDEX TO DRIVE NUMBER
6129 025476 010446      MOV      R4,-(SP) ;SAVE R4 FOR TYPEOUT
6130 ;TYPE DRIVE NUMBER
6131 025500 104403      TYPOS   ;GO TYPE--OCTAL ASCII
6132 025502 002 ;TYPE 2 DIGIT(S)
6133 025503 000 ;SUPPRESS LEADING ZEROS
6134 025504 104401 054066      TYPE     ,SLASH ; '/'
6135 025510 012737 052654 025554      MOV      #RP04B,2$ ;ADDRESS OF 'RP04' MESSAGE
6136 025516 132764 000001 033312      BITB     #BIT00,DRVTYP(R4) ;RP04 ?
6137 025524 001012      BNE      1$ ;BR IF YES
6138 025526 012737 052661 025554      MOV      #RP05,2$ ;ADDRESS OF 'RP05' MESSAGE
6139 025534 132764 000002 033312      BITB     #BIT01,DRVTYP(R4) ;RP05 ?
6140 025542 001003      BNE      1$ ;BR IF YES
6141 025544 012737 052666 025554      MOV      #RP06,2$ ;ADDRESS OF 'RP06' MESSAGE

```



```

6142 025552 104401          1$:  TYPE                ;TYPE THE MESSAGE WHICH FOLLOWS
6143 025554 000000          2$:  .WORD 0             ;MESSAGE ADDRESS
6144 025556 104401 001165   TYPE      , $CRLF      ;CR-LF
6145 025562 012737 000632 001350 3$:  MOV #410, CYLIMT      ;ASSUME AN RP04/5
6146 025570 132764 000004 033312 BITB #BIT02, DRVTYP(R4) ;SEE IF RP06
6147 025576 001403          BEQ 4$                ;BR IF NOT
6148 025600 012737 001456 001350 MOV #814, CYLIMT      ;CHANGE LIMIT TO 814
6149 025606 062760 177777 000122 4$:  ADD #-1, $FIRST(RO) ;SEE IF FIRST TIME STARTED
6150 025614 103417          BCS 5$                ;BR IF NOT
6151 025616 013760 001350 000106 MOV CYLIMT, MAXCYL(RO) ;LOAD MAXIMUM CYLINDER
6152 025624 005060 000110 CLR MINCYL(RO)        ;CLEAR MINIMUM CYLINDER
6153 025630 013760 001346 000112 MOV TRKLMT, MAXTRK(RO) ;LOAD MAXIMUM TRACK
6154 025636 005060 000114 CLR MINTRK(RO)        ;CLEAR MINIMUM TRACK
6155 025642 013760 001344 000116 MOV SECLMT, MAXSEC(RO) ;LOAD MAXIMUM SECTOR
6156 025650 005060 000120 CLR MINSEC(RO)        ;CLEAR MINIMUM SECTOR
6157 025654 006304          5$:  ASL R4                 ;SETUP TO ADDRESS WORDS
6158 025656 016403 054246   MOV TABLE(R4), R3    ;PARAMETER TABLE ADDRESS
6159 025662 013763 001350 000002 MOV CYLIMT, 2(R3)     ;LOAD CYLINDER LIMIT FOR LAST CYLINDER
6160 025670 013763 001350 000010 MOV CYLIMT, 10(R3)    ;LOAD CYLINDER LIMIT FOR STARTING CYLINDER
6161 025676 005737 000042   TST 42                ;UNDER MONITOR CONTROL ?
6162 025702 001002          BNE 6$                ;BR IF YES
6163 025704 004737 026370   JSR PC, PARENT        ;GET THE DRIVE'S PARAMETERS
6164 025710 116060 000120 000010 6$:  MOVB MINSEC(RO), $SEC(RO) ;INITIAL SECTOR VALUE
6165 025716 116060 000114 000011 MOVB MINTRK(RO), $TRK(RO) ;INITIAL TRACK VALUE
6166 025724 016060 000110 000012 MOV MINCYL(RO), $CYL(RO) ;INITIAL CYLINDER VALUE
6167 025732 012604          MOV (SP)+, R4          ;RESTORE R4
6168 025734 012603          MOV (SP)+, R3          ;RESTORE R3
6169 025736 000207          RTS PC                 ;RETURN
6170
6171 ;ROUTINE TO GET THE DRIVE I.D. FROM THE OPERATOR
6172
6173 025740 010546          GETID: MOV R5, -(SP)      ;SAVE R5
6174 025742 005737 000042   TST 42                ;UNDER MONITOR CONTROL ?
6175 025746 001036          BNE 2$                ;BR IF NOT
6176 025750 005037 001262 1$:  CLR CFLAG            ;CLEAR THE 'CONTROL C' FLAG
6177 025754 104401 053350   TYPE , ENTDRV        ;'ENTER DRV I.D.:'
6178 025760 005046          CLR -(SP)            ;CLEAR THE STACK
6179 025762 111016          MOVB (RO), (SP)     ;PUT THE DRIVE NUMBER ON THE STACK
6180 025764 104403          TYPOS                ;TYPE THE DRIVE NUMBER
6181 025766 002           .BYTE 2                ;TYPE 2 DIGITS
6182 025767 000           .BYTE 0                ;SUPPRESS LEADING ZEROS
6183 025770 104401 001165   TYPE      , $CRLF      ;CR-LF
6184 025774 104411          RDLIN                ;READ THE ENTRY
6185 025776 012605          MOV (SP)+, R5        ;GET THE ENTRY ADDRESS
6186 026000 005737 001262   TST CFLAG            ;'CONTROL C' ENTERED ?
6187 026004 001361          BNE 1$              ;BR IF IT WAS
6188 026006 121527 000056   CMPB (R5), #'.'      ;PERIOD ENTERED ?
6189 026012 001414          BEQ 2$              ;BR IF YES
6190 026014 112560 000224   MOVB (R5)+, $DRVID(RO) ;STORE THE I.D.
6191 026020 112560 000225   MOVB (R5)+, $DRVID+1(RO) ;STORE THE I.D.
6192 026024 112560 000226   MOVB (R5)+, $DRVID+2(RO) ;STORE THE I.D.
6193 026030 112560 000227   MOVB (R5)+, $DRVID+3(RO) ;STORE THE I.D.
6194 026034 112560 000230   MOVB (R5)+, $DRVID+4(RO) ;STORE THE I.D.
6195 026040 112560 000231   MOVB (R5)+, $DRVID+5(RO) ;STORE THE I.D.
6196 026044 012605          2$:  MOV (SP)+, R5        ;RESTORE R5
6197 026046 000207          RTS PC                 ;RETURN

```



```
6198  
6199 ;ROUTINE TO GET THE ADDRESSES OF ANY BAD SECTORS (UP TO A MAX OF 16)  
6200  
6201 026050 GETADR:  
6202 026050 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK  
6203 026052 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK  
6204 026054 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK  
6205 026056 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK  
6206 026060 005737 000042 TST 42 ;UNDER MONITOR CONTROL ?  
6207 026064 001012 BNE 15$ ;BR IF YES  
6208 026066 005037 001262 14$: CLR CFLAG ;CLEAR 'CONTROL C' FLAG  
6209 026072 104401 053436 TYPE ,ENTADR ;ENTER SECTOR ADDRESSES  
6210 026076 005046 CLR -(SP) ;CLEAR THE STACK  
6211 026100 111016 MOV B (R0),(SP) ;PUT THE DRIVE NUMBER ON THE STACK  
6212 026102 104403 TYPOS ;TYPE THE DRIVE NUMBER  
6213 026104 002 .BYTE 2 ;TYPE 2 DIGITS  
6214 026105 000 .BYTE 0 ;SUPPRESS LEADING ZEROS  
6215 026106 104401 001165 TYPE ,SCRLF ;CR-LF  
6216 026112 012703 000040 15$: MOV #32.,R3 ;NUMBER OF LOCATIONS IN THE TABLE TO PRESET  
6217 026116 012704 000124 MOV #BDSEC,R4 ;TABLE INCREMENT  
6218 026122 060004 ADD R0,R4 ;BLOCK STARTING ADDRESS  
6219 026124 012724 177777 1$: MOV #-1,(R4)+ ;SET LOCATION TO 1'S  
6220 026130 005303 DEC R3 ;DECREMENT TABLE SIZE COUNT  
6221 026132 001374 BNE 1$ ;BR IF NOT FINISHED WITH TABLE  
6222 026134 005737 000042 TST 42 ;UNDER MONITOR CONTROL ?  
6223 026140 001106 BNE 13$ ;BR IF YES  
6224 026142 162704 000100 SUB #64.,R4 ;SET POINTER TO BEGINNING OF TABLE  
6225 026146 012703 000020 MOV #16.,R3 ;NUMBER OF ADDRESSES IN TABLE  
6226 026152 104411 2$: RDLIN ;GET ADDRESS FROM OPERATOR  
6227 026154 012601 MOV (SP)+,R1 ;TEXT POINTER  
6228 026156 005737 001262 TST CFLAG ;'CONTROL C' ENTERED ?  
6229 026162 001341 BNE 14$ ;BR IF IT WAS  
6230 026164 013702 001350 MOV CYLMT,R2 ;UPPER LIMIT OF INPUT  
6231 026170 004537 027730 JSR R5,CK.DIG ;CHECK THE DIGIT(S)  
6232 026174 026336 12$ ;CARRIAGE RETURN ONLY ENTERED  
6233 026176 026356 13$ ;PERIOD ONLY ENTERED  
6234 026200 026336 12$ ;ILLEGAL INPUT  
6235 026202 026214 4$ ;TERMINATED WITH A CARRIAGE RETURN  
6236 026204 026220 5$ ;TERMINATED WITH A ".  
6237 026206 026210 3$ ;TERMINATED WITH A ".  
6238 026210 010214 3$: MOV R2,(R4) ;CYLINDER ADDRESS  
6239 026212 000461 BR 13$ ;EXIT, PERIOD ENTERED  
6240 026214 010214 4$: MOV R2,(R4) ;CYLINDER ADDRESS  
6241 026216 000442 BR 11$ ;FINISHED WITH THIS ADDRESS, 'CR' ENTERED  
6242 026220 010214 5$: MOV R2,(R4) ;CYLINDER ADDRESS  
6243 026222 013702 001346 MOV TRKLMT,R2 ;UPPER LIMIT OF INPUT  
6244 026226 004537 027730 JSR R5,CK.DIG ;CHECK THE DIGIT(S)  
6245 026232 026336 12$ ;CARRIAGE RETURN ONLY ENTERED  
6246 026234 026356 13$ ;PERIOD ONLY ENTERED  
6247 026236 026336 12$ ;ILLEGAL INPUT  
6248 026240 026254 7$ ;TERMINATED WITH A CARRIAGE RETURN  
6249 026242 026262 8$ ;TERMINATED WITH A ".  
6250 026244 026246 6$ ;TERMINATED WITH A ".  
6251 026246 110264 000003 6$: MOV B R2,3(R4) ;TRACK ADDRESS  
6252 026252 000441 BR 13$ ;EXIT, ENTRY TERMINATED BY PERIOD  
6253 026254 110264 000003 7$: MOV B R2,3(R4) ;TRACK ADDRESS
```


6254	026260	000421			BR	11\$:FINISHED WITH THIS ADDRESS, 'CR' ENTERED
6255	026262	110264	000003		8\$:	MOVB	R2,3(R4)	:TRACK ADDRESS
6256	026266	013702	001344			MOV	SECLMT,R2	:UPPER LIMIT OF INPUT
6257	026272	004537	027730			JSR	R5,CK.DIG	:CHECK THE DIGIT(S)
6258	026276	026336				12\$:CARRIAGE RETURN ONLY ENTERED
6259	026300	026356				13\$:PERIOD ONLY ENTERED
6260	026302	026336				12\$:ILLEGAL INPUT
6261	026304	026320				10\$:TERMINATED WITH A CARRIAGE RETURN
6262	026306	026336				12\$:TERMINATED WITH A ''
6263	026310	026312				9\$:TERMINATED WITH A ''
6264	026312	110264	000002		9\$:	MOVB	R2,2(R4)	:SECTOR ADDRESS
6265	026316	000417				BR	13\$:EXIT, ENTRY TERMINATED BY PERIOD
6266	026320	110264	000002		10\$:	MOVB	R2,2(R4)	:SECTOR ADDRESS
6267	026324	005303			11\$:	DEC	R3	:MORE ENTRIES ?
6268	026326	001413				BEQ	13\$:BR IF NOT
6269	026330	062704	000004			ADD	#4,R4	:INCREMENT THE TABLE POINTER
6270	026334	000706				BR	2\$:CONTINUE
6271	026336	012714	177777		12\$:	MOV	#-1,(R4)	:CLEAR PRESENT TABLE ENTRY
6272	026342	012764	177777	000002		MOV	#-1,2(R4)	:CLEAR PRESENT TABLE ENTRY
6273	026350	104401	053566			TYPE	,BADENT	: 'INVALID ENTRY'
6274	026354	000676				BR	2\$:TRY AGAIN
6275	026356				13\$:			
6276	026356	012604				MOV	(SP)+,R4	::POP STACK INTO R4
6277	026360	012603				MOV	(SP)+,R3	::POP STACK INTO R3
6278	026362	012602				MOV	(SP)+,R2	::POP STACK INTO R2
6279	026364	012601				MOV	(SP)+,R1	::POP STACK INTO R1
6280	026366	000207				RTS	PC	:RETURN
6281								
6282								
6283								
6284								
6285								
6286								
6287	026370	010346						
6288	026372	005037	001262					
6289	026376	012337	026406					
6290	026402	001442						
6291	026404	104401						
6292	026406	000000						
6293	026410	012302						
6294	026412	012305						
6295	026414	011546						
6296	026416	104405						
6297	026420	104401	054066					
6298	026424	104411						
6299	026426	012601						
6300	026430	005737	001262					
6301	026434	001021						
6302	026436	004537	027730					
6303	026442	026376						
6304	026444	026510						
6305	026446	026462						
6306	026450	026456						
6307	026452	026462						
6308	026454	026474						
6309	026456	010215			5\$:	MOV	R2,(R5)	:MOVE NEW VALUE TO PARAMETER LOCATION

:PARAMETER ENTRY ROUTINE

:CALL
: MOV #ADR,R3
: JSR PC,PARENT

PARENT: MOV R3,-(SP)

1\$: CLR CFLAG
MOV (R3)+,3\$
BEQ 9\$

3\$: TYPE .WORD 0
MOV (R3)+,R2

MOV (R3)+,R5
MOV (R5),-(SP)

TYPDS
TYPE ,SLASH
RD LIN

MOV (SP)+,R1
TST CFLAG

BNE 8\$
JSR R5,CK.DIG

1\$
9\$
6\$
5\$
6\$
7\$

:PARAMETER TABLE ADDRESS
:GET THE PARAMETERS

:SAVE THE PARAMETER TABLE ADDRESS
:CLEAR THE 'CONTROL C' FLAG

:ADDRESS OF PARAMETER NAME
:BR IF AT END OF TABLE

:TYPE THE PARAMETER NAME
:ADDRESS OF PARAMETER NAME TEXT

:MAXIMUM PARAMETER VALUE
:ADDRESS OF PARAMETER

:CURRENT VALUE OF PARAMETER
:TYPE THE CURRENT VALUE OF THE PARAMETER

: ' / '
:READ THE KEYBOARD
:INPUT ASCII STRING ADDRESS

: 'CONTROL C' ENTERED ?
:BR IF IT WAS

:CHECK THE DIGIT(S)
:CARRIAGE RETURN ONLY ENTERED

:PERIOD ONLY ENTERED
:ILLEGAL INPUT
:TERMINATED WITH A CARRIAGE RETURN

:TERMINATED WITH A ''
:TERMINATED WITH A ''


```

6310 026460 000746
6311 026462 104401 053566
6312 026466 162703 000006
6313 026472 000741
6314 026474 010215
6315 026476 000404
6316 026500 005037 001262
6317 026504 011603
6318 026506 000733
6319 026510 005726
6320 026512 000207
6321
6322
6323
6324
6325
6326
6327
6328 026514 104401 053277
6329 026520 104401 052450
6330 026524 010446
6331
6332 026526 104403
6333 026530 002
6334 026531 000
6335 026532 104401
6336 026534 000000
6337 026536 104401 001165
6338 026542 000207
6339
6340
6341
6342
6343
6344
6345 026544 005004
6346 026546 111004
6347 026550 146437 033416 001462
6348 026556 006304
6349 026560 010064 001464
6350 026564 104401 001165
6351 026570 104401 001165
6352 026574 104401 053052
6353 026600 104401 053163
6354 026604 006204
6355 026606 010446
6356
6357 026610 104403
6358 026612 002
6359 026613 000
6360 026614 104401 001165
6361 026620 000207
6362
6363
6364
6365 026622 032777 000020 152310

```

```

BR 1$ ;GET MORE PARAMETERS
6$: TYPE ,BADENT ;'BAD ENTRY'
SUB #6,R3 ;DECREMENT THE TABLE POINTER
BR 1$ ;TRY AGAIN
7$: MOV R2,(R5) ;NEW VALUE
BR 9$ ;EXIT
8$: CLR CFLAG ;CLEAR THE 'CONTROL C' FLAG
MOV (SP),R3 ;RELOAD THE PARAMETER TABLE ADDRESS
BR 1$ ;TRY AGAIN
9$: TST (SP)+ ;CORRECT THE STACK POINTER
RTS PC ;RETURN

;TYPEOUT ASSIGN/DEASSIGN ERROR MESSAGE
;CALL
; MOV #MESADR,ASNMSG ;ERROR MESSAGE ADDRESS
; JSR PC,ASNERR
; RETURN

ASNERR: TYPE ,QUES ;QUESTION MARK
TYPE ,UNTMSG ;TYPE 'DRIVE'
MOV R4,-(SP) ;;SAVE R4 FOR TYPEOUT
;;TYPE DRIVE NUMBER
;;GO TYPE--OCTAL ASCII
;;TYPE 2 DIGIT(S)
;;SUPPRESS LEADING ZEROS
ASNMSG: TYPE ,SPECMSG ;TYPE SPECIFIC MESSAGE
;WORD 0 ;MESSAGE ADDRESS
TYPE ,%CRLF
RTS PC

;DEASSIGN DRIVE IF A FATAL ERROR OCCURS
;CALL
; JSR PC,DROP
; RETURN

DROP: CLR R4 ;CLEAR R4 FOR DRIVE NUMBER
MOVB (R0),R4 ;MOVE DRIVE NUMBER TO R4
BICB ATABIT(R4),ASNLIST ;REMOVE DRIVE FROM ASSIGNED LIST
ASL R4 ;MAKE DRIVE NUMBER INTO A TABLE INDEX
MOV R0,DUNIT(R4) ;PUT DRIVE IN DROP LIST
TYPE ,%CRLF
TYPE ,%CRLF
TYPE ,DROPNG ;TYPE 'DROPPING DRIVE'
TYPE ,DRNUM ;'DRIVE #'
ASR R4 ;DRIVE NUMBER
MOV R4,-(SP) ;;SAVE R4 FOR TYPEOUT
;;TYPE DRIVE NUMBER
;;GO TYPE--OCTAL ASCII
;;TYPE 2 DIGIT(S)
;;SUPPRESS LEADING ZEROS
TYPE ,%CRLF
RTS PC

;ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE
ABNRML: BIT #SW04,@SWR ;SEE IF SWITCH 4 SET

```



```

6366 026630 001006          BNE 1$ ;BR IF IT'S SET
6367 026632 023760 001406 000056  CMP MAXER,$TOTAL(RO) ;CHECK TOTAL ERROR VALUE
6368 026640 103002          BHIS 1$ ;BR IF ERRORS DONOT EXCEED MAX
6369 026642 000137 026544  JMP DROP ;DEASSING THE DRIVE
6370 026646 000207          1$: RTS PC ;RETURN
6371
6372 ;ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
6373
6374 026650 005737 001430  EOP: TST ENDET ;END OF PASS DETERMINED BY SEEKS OR WORDS ?
6375 026654 001412          BEQ EOP1 ;BR IF SEEKS
6376 026656 026037 000054 001374  CMP $READ+2(RO),ENDCON+2 ;CHECK MSW OF WORDS READ COUNT
6377 026664 101017          BHI EOP2 ;BR IF MSW GREATER THAN LIMIT
6378 026666 103405          BLO EOP1 ;BR IF MSW LESS THAN LIMIT
6379 026670 026037 000052 001372  CMP $READ(RO),ENDCON ;CHECK LSW AGAINST LIMIT
6380 026676 103012          BHIS EOP2 ;BR IF EQUAL OR GREATER
6381 026700 000510          BR EOPX ;EXIT
6382 026702 026037 000044 001400  EOP1: CMP $POSIT+2(RO),ENDSEK+2 ;CHECK MSW OF SEEK COUNT
6383 026710 101005          BHI EOP2 ;BR IF MSW GREATER THAN LIMIT
6384 026712 103503          BLO EOPX ;EXIT IF MSW LESS THAN LIMIT
6385 026714 026037 000042 001376  CMP $POSIT(RO),ENDSEK ;CHECK LSW OF SEEK COUNT
6386 026722 103477          BLO EOPX ;EXIT IF LSW LESS THAN LIMIT
6387 026724 104401 001165  EOP2: TYPE , $CRLF ;CR-LF
6388 026730 104401 053106  TYPE , ENDPAS ;END OF PASS FOR THE DRIVE
6389 026734 016046 000070  MOV $PASSC(RO),-(SP) ;PUT PASS COUNT ON THE STACK
6390 026740 104405          TYPDS ;CONVERT PASS COUNT TO DECIMAL AND TYPE IT
6391 026742 111037 001246  MOVB (RO),UNIT ;STORE THE DRIVE NUMBER
6392 026746 032777 000020 152164  BIT #SW04,@SWR ;SWITCH 4 SET ?
6393 026754 001017          BNE 1$ ;BR IF SET
6394 026756 026037 000070 001402  CMP $PASSC(RO),PASCNT ;SEE IF AT END OF TEST
6395 026764 103413          BLO 1$ ;BR IF NOT
6396 026766 104401 053122  TYPE ,ENDTST ;TYPE 'END OF TEST'
6397 026772 104401 053163  TYPE ,DRNUM ;'DRIVE #'
6398 026776 013746 001246  MOV UNIT,-(SP) ;:SAVE UNIT FOR TYPEOUT
6399 ;:TYPE DRIVE NUMBER
6400 027002 104403          TYPOS ;:GO TYPE--OCTAL ASCII
6401 027004 002          .BYTE 2 ;:TYPE 2 DIGIT(S)
6402 027005 000          .BYTE 0 ;:SUPPRESS LEADING ZEROS
6403 027006 104401 001165  TYPE , $CRLF ;CR-LF
6404 027012 000431          BR 3$ ;DEASSIGN THE DRIVE
6405 027014 104401 053163  1$: TYPE ,DRNUM ;'DRIVE #'
6406 027020 013746 001246  MOV UNIT,-(SP) ;:SAVE UNIT FOR TYPEOUT
6407 ;:TYPE DRIVE NUMBER
6408 027024 104403          TYPOS ;:GO TYPE--OCTAL ASCII
6409 027026 002          .BYTE 2 ;:TYPE 2 DIGIT(S)
6410 027027 000          .BYTE 0 ;:SUPPRESS LEADING ZEROS
6411 027030 104401 001165  TYPE , $CRLF ;CR-LF
6412 027034 004737 022764  JSR PC,TYPEST ;TYPE THE DRIVE'S STATISTICS
6413 027040 010346          MOV R3,-(SP) ;SAVE R3
6414 027042 010446          MOV R4,-(SP) ;SAVE R4
6415 027044 010004          MOV R0,R4 ;DRIVE'S BLOCK ADDRESS
6416 027046 062704 000036  ADD #SOPERC,R4 ;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
6417 027052 012703 000010  MOV #8.,R3 ;NUMBER OF LOCNS TO BE CLEARED
6418 ;(ERROR COUNTERS NOT CLEARED)
6419 027056 005024          2$: CLR (R4)+ ;CLEAR THE LOCN
6420 027060 005303          DEC R3 ;DECREMENT THE LOCATION COUNTER
6421 027062 001375          BNE 2$ ;BR IF MORE TO GO
    
```



```
6422 027064 012604      MOV      (SP)+,R4      ;RESTORE R4
6423 027066 012603      MOV      (SP)+,R3      ;RESTORE R3
6424 027070 005260 000070  INC      $PASSC(R0)    ;INCREMENT THE PASS COUNT
6425 027074 000412      BR       EOPX          ;EXIT
6426 027076 104401 001165 3$:      TYPE      ,%CRLF
6427 027102 005004      CLR      R4            ;CLEAR R4 FOR DRIVE NUMBER
6428 027104 111004      MOV      (R0),R4       ;MOVE DRIVE NUMBER
6429 027106 146437 033416 001462  BIC      ATABIT(R4),ASNLST ;DELETE DRIVE FROM ASSIGNED LIST
6430 027114 006304      ASL      R4            ;MAKE DRIVE NUMBER INTO TABLE INDEX
6431 027116 010064 001464  MOV      R0,DUNIT(R4)  ;PUT BLOCK ADDRESS INTO DROP LIST
6432 027122 000207  EOPX:    RTS          PC      ;RETURN
6433
6434      ;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
6435      ;CALL
6436      ;      MOV      NUMBER,R5      ;DIVISOR INTO R5
6437      ;      JSR      PC,GETREM
6438      ;      RETURN
6439      ;REMAINDER IS IN R5
6440 027124 013746 032562  GETREM:  MOV      $LONUM,-(SP)  ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
6441 027130 013746 032560  MOV      $HINUM,-(SP)  ;UPPER PART
6442 027134 010546      MOV      R5,-(SP)      ;PUT THE DIVISOR ONTO THE STACK
6443 027136 004737 027152  JSR      PC,LINKDV     ;DIVIDE THE RANDOM NUMBERS
6444 027142 012605      MOV      (SP)+,R5      ;PUT THE REMAINDER INTO R5
6445 027144 005726      TST      (SP)+         ;ADJUST THE STACK POINTER
6446 027146 000240      NOP
6447 027150 000207      RTS          PC      ;FOR DEBUGGING HALT
6448
6449      ;LINK ROUTINE TO THE DIVISION UTILITY SUBROUTINE
6450      ;      THIS ROUTINE ALLOWS THE 'SYSMAC' DIVIDE ROUTINE
6451      ;      CALLING SEQUENCE TO BE USED
6452
6453 027152 104412  LINKDV:  SAVREG      ;STORE R0 - R5
6454 027154 016605 000026  MOV      26(SP),R5     ;DIVISOR
6455 027160 005004      CLR      R4            ;OTHER DIVISOR WORD
6456 027162 016602 000030  MOV      30(SP),R2     ;UPPER DIVIDEND WORD
6457 027166 016603 000032  MOV      32(SP),R3     ;LOWER DIVIDEND WORD
6458 027172 005000      CLR      R0            ;CLEAR OTHER DIVIDEND REGISTERS
6459 027174 005001      CLR      R1
6460 027176 004737 027220  JSR      PC,M.DPID     ;GO TO THE DIVIDE ROUTINE
6461 027202 010166 000030  MOV      R1,30(SP)     ;REMAINDER ON THE STACK
6462 027206 010366 000032  MOV      R3,32(SP)     ;QUOTIENT ON THE STACK
6463 027212 104413      RESREG      ;RESTORE R0 - R5
6464 027214 012616      MOV      (SP)+,(SP)   ;MOVE RETURN UP THE STACK
6465 027216 000207      RTS          PC
6466
6467      ;      DIVISION UTILITY SUBROUTINE
6468      ;      R0-R1-R2-R3=DIVIDEND
6469      ;      R4-R5=DIVISOR
6470      ;      R0-R1=REMAINDER AFTER DIVISION
6471      ;      R2-R3=QUOTIENT AFTER DIVISION
6472      ;      ENTER WITH JSR PC,M.DPID
6473
6474 027220 012746 000040  M.DPID:  MOV      #40,-(SP)    ;COUNTER FOR DIVISION CYCLES
6475 027224 010446      MOV      R4,-(SP)     ;HIGH ORDER
6476 027226 010546      MOV      R5,-(SP)     ;LOW ORDER DIVISOR TO THE STACK
6477 027230 005466 000002  NEG      2(SP)         ;FORM NEGATIVE
```



```
6478 027234 005416          NEG @SP          ;VERSION OF THE DIVISOR
6479 027236 005666 000002   SBC 2(SP)
6480 027242 061601          ADD @SP,R1
6481 027244 005500          ADC R0          ;PERFORM THE INITIAL SUBTRACTION
6482 027246 066600 000002   ADD 2(SP),R0
6483 027252 103445          BCS M.DP50      ;IF CARRY THEN OVERFLOW HAS OCCURRED
6484 027254 005046          CLR -(SP)      ;THIS IS A LONGER LASTING CARRY BIT
6485 027256 006103          M.DP40: ROL R3
6486 027260 006102          ROL R2
6487 027262 006101          ROL R1
6488 027264 006100          ROL R0
6489 027266 005716          TST @SP        ;TEST "CARRY" INDICATOR
6490 027270 001410          BEQ M.DP41     ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
6491 027272 005016          CLR @SP        ;CLEAR UP FOR NEXT TIME
6492 027274 066601 000002   ADD 2(SP),R1
6493 027300 005500          ADC R0          ;ADD -(DIVISOR)
6494 027302 005516          ADC @SP        ;SET "CARRY"
6495 027304 066600 000004   ADD 4(SP),R0 ;<-
6496 027310 000404          BR M.DP42
6497 027312 060501          M.DP41: ADD R5,R1
6498 027314 005500          ADC R0          ;ADD +(DIVISOR)
6499 027316 005516          ADC @SP        ;SET "CARRY"
6500 027320 060400          ADD R4,R0 ;<-
6501 027322 005516          M.DP42: ADC @SP
6502 027324 005716          TST @SP        ;SET "CARRY"
6503 027326 001401          BEQ .+4 ;-> ;TEST THE UPDATE INDICATOR
6504 027330 005203          INC R3 ;> ;IF ZERO FORGET IT
6505 027332 005366 000006   DEC 6(SP) ;<- ;NO CARRY POSSIBLE HERE
6506 027336 003347          BGT M.DP40    ;DECREMENT COUNTER
6507 027340 006003          ROR R3        ;BRANCH IF MORE TO DO
6508 027342 103404          BCS M.DP44
6509 027344 060501          ADD R5,R1
6510 027346 005500          ADC R0
6511 027350 060400          ADD R4,R0
6512 027352 000241          CLC
6513 027354 006103          M.DP44: ROL R3
6514 027356 062706 000010   ADD #10,SP     ;ADJUST STACK BY 4 WORDS
6515 027362 000242          CLV
6516 027364 000207          RTS PC
6517 027366 062706 000006   M.DP50: ADD #6,SP
6518 027372 000262          SEV
6519 027374 000207          RTS PC
6520
6521
6522          ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
6523          ;CALL
6524          ;      MOV #ADR, -(SP) ;ADDRESS OF NUMBER (IN ASCII)
6525          ;      JSR R5, REPLZ
6526          ;      .WORD N ;'N' IS NUMBER OF DIGITS TO BE TYPED
6527
6528 027376 010046          REPLZ: MOV R0, -(SP) ;SAVE R0
6529 027400 012746 000012   MOV #10, -(SP) ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
6530 027404 162516          SUB (R5)+, (SP) ;SUBTRACT DIGITS TO FORM INDEX
6531 027406 016600 000006   MOV 6(SP), R0 ;ADDRESS OF NUMBER TO R0
6532 027412 122710 000060   1$: CMPB #'0', (R0) ;BYTE EQUAL TO ASCII '0' ?
6533 027416 001004          BNE 2$ ;BR IF NOT
```



```
6534 027420 112710 000040          MOVB   #40,(R0)          ;REPLACE THE ZERO WITH A SPACE
6535 027424 005200          INC    R0              ;INCREMENT THE BYTE ADDRESS
6536 027426 000771          BR     1$             ;GO BACK AND LOOK FOR MORE LEADING ZEROS
6537 027430 105710          2$:   TSTB   (R0)       ;SEE IF ZERO BYTE TERMINATOR
6538 027432 001003          BNE   3$             ;BR IF NOT
6539 027434 005300          DEC   R0              ;BACKUP STRING POINTER
6540 027436 112710 000060          MOVB   #'0,(R0)       ;PUT A ZERO BACK IN
6541 027442 016637 000006 027456 3$:   MOV    6(SP),4$       ;PUT ADDRESS IN LOCATION FOR TYPEOUT
6542 027450 062637 027456          ADD   (SP)+,4$       ;BEGINNING OF SIGNIFICANT DIGITS
6543 027454 104401          TYPE          ;TYPE THE NUMBER
6544 027456 000000          4$:   .WORD  0          ;ADDRESS OF NUMBER
6545 027460 012600          MOV   (SP)+,R0       ;RESTORE R0
6546 027462 012616          MOV   (SP)+,(SP)    ;MOVE RETURN ADDRESS
6547 027464 000205          RTS    R5            ;RETURN
6548
6549          ;TYPE NUMERICAL ASCIZ STRING SUPRESS LEADING ZEROS
6550
6551          ;CALL
6552          ;   MOV   #NUMADR,-(SP) ;FIRST ADDRESS OF ASCIZ STRING
6553          ;   JSR   PC,$SUPRS
6554
6555 027466 010046          $SUPRS: MOV   R0,-(SP)    ;SAVE R0
6556 027470 016600 000004          MOV   4(SP),R0      ;PICKUP THE POINTER
6557 027474 105710          1$:   TSTB   (R0)       ;TERMINATOR ?
6558 027476 001403          BEQ   2$             ;BR IF YES
6559 027500 122720 000060          CMPB  #'0,(R0)+     ;IS THIS AN ASCII '0' ?
6560 027504 001773          BEQ   1$             ;BR IF YES
6561 027506 005300          2$:   DEC   R0        ;BACKUP BY '1'
6562 027510 010037 027516          MOV   R0,3$         ;SAVE FOR TYPING
6563 027514 104414          3$:   DISPLY          ;GO PRINT
6564 027516 000000          .WORD  0            ;ASCIZ POINTER GOES HERE
6565 027520 012600          MOV   (SP)+,R0      ;RESTORE R0
6566 027522 012616          MOV   (SP)+,(SP)    ;RESTORE THE STACK
6567 027524 000207          RTS    PC           ;RETURN
6568
6569          ;ROUTINE TO TYPE AT PRIORITY 4
6570
6571 027526 013746 177776          TYPRI4: MOV   @#PS,-(SP) ;SAVE THE PRESENT STATUS
6572 027532 012737 000200 177776          MOV   #200,@#PS    ;CHANGE THE PRIORITY TO 4
6573 027540 012537 027550          MOV   (R5)+,1$     ;MESSAGE ADDRESS
6574 027544 004737 031132          JSR   PC,$TYPE     ;TYPE THE MESSAGE
6575 027550 000000          1$:   .WORD  0        ;MESSAGE ADDRESS GOES HERE
6576 027552 000205          RTS    R5          ;RETURN
6577
6578          ;ROUTINE TO TYPE ERRORS
6579          ;CALL
6580          ;   DISPLY          ;MUST DEFINED IN 'TRAP' TABLE
6581          ;   MESADR          ;ADDRESS OF MESSAGE
6582          ;   RETURN
6583
6584 027554 032777 020000 151356 $DSPLY: BIT   #BIT13,@SWR ;INHIBIT ERROR TYPEOUT ?
6585 027562 001004          BNE   1$             ;BR IF YES
6586 027564 005037 177776          CLR   @#PS          ;SET PRIORITY TO ZERO
6587 027570 000137 031132          JMP   $TYPE         ;TYPE THE MESSAGE
6588 027574 062716 000002          1$:   ADD   #2,(SP)   ;INCREMENT THE RETURN
6589 027600 000002          RTI                    ;RETURN
```



```

6590
6591 ;THIS ROUTINE IS USED TO CHECK IF AN
6592 ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
6593 ;CALL
6594 ;      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
6595 ;      JSR      R5,CK.OCT    ;CHECK THE CHARACTER
6596 ;      RETURN1   ;CHARACTER IS NOT BETWEEN 0-7
6597 ;      RETURN2   ;CHARACTER IS IN R2 AS A
6598 ;                      ;OCTAL DIGIT
6599
6600 027602 121127 000060 CK.OCT: CMPB      (R1),#'0      ;LESS THAN ZERO?
6601 027606 103407      BLO      1$      ;YES -- BRANCH
6602 027610 121127 000067      CMPB      (R1),#'7      ;GREATER THAN SEVEN?
6603 027614 101004      BHI      1$      ;YES -- BRANCH
6604 027616 111102      MOVB      (R1),R2    ;GET THE CHARACTER
6605 027620 042702 177770      BIC      #'C7,R2    ;STRIP AWAY THE ASCII
6606 027624 005725      TST      (R5)+    ;ADJUST FOR RETURN
6607 027626 000205      1$:      RTS      R5      ;RETURN
6608
6609 ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
6610 ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
6611 ;CALL
6612 ;      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
6613 ;      JSR      R5,CK.DEC    ;CHECK THE CHARACTER
6614 ;      RETURN1   ;NOT BETWEEN 0 AND 9
6615 ;      RETURN2   ;BETWEEN 0 AND 9
6616 ;                      ;R2 = DIGIT
6617
6618 027630 121127 000060 CK.DEC: CMPB      (R1),#'0      ;LESS THAN ZERO?
6619 027634 103407      BLO      1$      ;YES -- BRANCH
6620 027636 121127 000071      CMPB      (R1),#'9      ;GREATER THAN NINE?
6621 027642 101004      BHI      1$      ;YES -- BRANCH
6622 027644 111102      MOVB      (R1),R2    ;GET THE CHARACTER
6623 027646 042702 000060      BIC      #'0,R2    ;STRIP AWAY THE ASCII
6624 027652 005725      TST      (R5)+    ;ADJUST FOR RETURN
6625 027654 000205      1$:      RTS      R5      ;RETURN
6626
6627 ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
6628 ;DETERMINE WHAT IT IS.
6629 ;CALL
6630 ;      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
6631 ;      JSR      R5,CK.CHR    ;CHECK CHARACTER
6632 ;      RETURN   ADR1        ;UNKNOWN CHARACTER
6633 ;      RETURN   ADR2        ;CARRIAGE RETURN * (R1)=ADR+1
6634 ;      RETURN   ADR3        ;COMMA * (R1)=ADR+1
6635 ;      RETURN   ADR4        ;PERIOD * (R1)=ADR+1
6636 ;      RETURN   ADR5        ;DIGIT BETWEEN 0 AND 7.
6637 ;      RETURN   ADR6        ;DIGIT BETWEEN 8 AND 9.
6638 ;                      ;R2 = DIGIT * (R1)=ADR+1
6639
6640 027656 105711      CK.CHR: TSTB      (R1)        ;"CARRIAGE RETURN"?
6641 027660 001417      BEQ      3$      ;YES -- BRANCH
6642 027662 121127 000054      CMPB      (R1),#',      ;"COMMA"?
6643 027666 001413      BEQ      2$      ;YES -- BRANCH
6644 027670 121127 000056      CMPB      (R1),#'.      ;"PERIOD"?
6645 027674 001407      BEQ      1$      ;YES -- BRANCH

```


CZRJDDO, RP04/5/6 MLT-DR LGC
CZRJDD.P11 06-JUN-79 08:32

MACY11 30A(1052) 03-JUL-79 10:41 # 10 PAGE 131
GENERAL SUPPORT SUBROUTINES

SEQ 0129

6646 027676 004537 027630
6647 027702 000410
6648 027704 004537 027602
6649 027710 005725
6650 027712 005725
6651 027714 005725

JSR R5,CK.DEC ;'DIGIT'?
BR 4\$;NO -- BRANCH
JSR R5,CK.OCT ;OCTAL ?
TST (R5)+ ;DIGIT BETWEEN 8-9
TST (R5)+ ;DIGIT BETWEEN 0-7
1\$: TST (R5)+ ;PERIOD


```

6652 027716 005725      2$:   TST      (R5)+      ;COMMA
6653 027720 005725      3$:   TST      (R5)+      ;CARRIAGE RETURN
6654 027722 005201          INC      R1          ;MOVE POINTER TO NEXT CHARACTER
6655 027724 011505      4$:   MOV      (R5),R5     ;UNKNOWN CHARACTER
6656 027726 000205          RTS      R5          ;RETURN
6657
6658 ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
6659 ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
6660 ;CALL
6661 ;       MOV      #ADR,R1      ;ADDRESS OF ASCII STRING
6662 ;       MOV      #NUM,R2      ;MAX. MAGNITUDE OF INPUT NUMBER
6663 ;       JSR      R5,CK.DIG     ;CHECK DIGITS
6664 ;       RETURN   ADR1         ;"CR" ONLY ENTERED -- R2=0
6665 ;       RETURN   ADR2         ;"PERIOD" ONLY ENTERED -- R2=0
6666 ;       RETURN   ADR3         ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
6667 ;       RETURN   ADR4         ;"CR" -- R2 = NUMBER
6668 ;       RETURN   ADR5         ;"COMMA" -- R2 = NUMBER
6669 ;       RETURN   ADR6         ;"PERIOD" -- R2 = NUMBER
6670
6671 027730 010446      CK.DIG: MOV      R4,-(SP)      ;SAVE R4
6672 027732 010346          MOV      R3,-(SP)      ;SAVE R3
6673 027734 010246          MOV      R2,-(SP)      ;SAVE THE MAX. SIZE ON THE STACK
6674 027736 005002          CLR      R2          ;START WITH 0
6675 027740 005003          CLR      R3
6676 027742 005004          CLR      R4
6677 027744 004537 027656      JSR      R5,CK.CHR     ;CHECK ONE CHARACTER
6678 027750 030044          6$:          ;ILLEGAL CHARACTER
6679 027752 030052          9$:          ;CARRIAGE RETURN
6680 027754 030044          6$:          ;
6681 027756 030046          7$:          ;
6682 027760 027764          1$:          ;DIGIT 0-7
6683 027762 027764          1$:          ;DIGIT 8-9
6684 027764 062705 000004      1$:   ADD      #4,R5      ;STEP RETURN. POINTER PAST "CR" & "PERIOD" RETURNS
6685 027770 006303      2$:   ASL      R3          ;INPUT NUMBER *2
6686 027772 010346          MOV      R3,-(SP)      ;SAVE *2
6687 027774 006303          ASL      R3          ;*4
6688 027776 006303          ASL      R3          ;*8
6689 030000 062603          ADD      (SP)+,R3      ;(*2)+(*8) = *10
6690 030002 060203          ADD      R2,R3        ;UPDATE THE INPUT NUMBER
6691 030004 004537 027656      JSR      R5,CK.CHR     ;CHECK ONE CHARACTER
6692 030010 030050          8$:          ;ILLEGAL CHARACTER
6693 030012 030034          5$:          ;CARRIAGE RETURN
6694 030014 030032          4$:          ;
6695 030016 030024          3$:          ;
6696 030020 027770          2$:          ;DIGIT 0-7
6697 030022 027770          2$:          ;DIGIT 8-9
6698 030024 105711      3$:   TSTB     (R1)        ;DOES A "CR" FOLLOW THE "PERIOD"
6699 030026 001010          BNE      8$          ;BR IF NOT
6700 030030 005724          TST      (R4)+        ;INCREMENT THE RETURN
6701 030032 005724      4$:   TST      (R4)+        ;INCREMENT THE RETURN
6702 030034 005724      5$:   TST      (R4)+        ;INCREMENT THE RETURN
6703 030036 020316          CMP      R3,(SP)      ;CHECK THE MAGNITUDE OF THE NUMBER
6704 030040 101004          BHI      9$          ;BR IF ENTERED NUMBER TOO LARGE
6705 030042 000402          BR       8$          ;BYPASS INCREMENT
6706 030044 005725      6$:   TST      (R5)+        ;INCREMENT RETURN PAST INVALID RETURN
6707 030046 005725      7$:   TST      (R5)+        ;INCREMENT RETURN

```



```
6708 030050 060405      8$:   ADD     R4,R5      ;SETUP RETURN POINTER
6709 030052 010302      9$:   MOV     R3,R2      ;ENTERED VALUE
6710 030054 005726          TST     (SP)+          ;CLEAN MAX. SIZE OFF OF STACK
6711 030056 012603          MOV     (SP)+,R3      ;RESTORE R3
6712 030060 012604          MOV     (SP)+,R4      ;RESTORE R4
6713 030062 011505          MOV     (R5),R5      ;GET RETURN ADDRESS
6714 030064 000205          RTS      R5          ;RETURN
6715
6716 ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
6717 ;UNSIGNED DECIMAL ASCII NUMBER.
6718 ;CALL
6719 ;       MOV     NUMBER,-(SP) ;PUT THE NUMBER ON THE STACK
6720 ;       JSR     PC,$SB2D     ;CALL
6721 ;       RETURN                    ;ADDRESS OF THE 1ST ASCII CHAR IS ON THE STACK
6722
6723 ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
6724 ;       THE SYSMAC LIBRARY, REV C AND LATER
6725
6726 030066 016637 000002 030112 $SB2D: MOV     2(SP),1$      ;SAVE THE BINARY NUMBER
6727 030074 012746 030112          MOV     #1$,-(SP)    ;SET THE POINTER
6728 030100 004737 032660          JSR     PC,$DB2D     ;CALL THE DOUBLE LENGTH CONVERT
6729 030104 012666 000002          MOV     (SP)+,2(SP)  ;PICKUP THE POINTER
6730 030110 000207          RTS      PC          ;RETURN
6731 030112 000000 000000      1$:   .WORD  0,0
6732
6733 ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
6734 ;UNSIGNED OCTAL ASCII NUMBER.
6735 ;CALL
6736 ;       MOV     NUMBER,-(SP) ;PUT THE NUMBER ON THE STACK
6737 ;       JSR     PC,$SB20     ;CALL
6738 ;       RETURN                    ;ADDRESS OF THE 1ST ASCII CHAR IS ON THE STACK
6739
6740 ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB20', NOT THE VERSION ON
6741 ;       THE SYSMAC LIBRARY, REV C AND LATER
6742
6743 030116 016637 000002 030142 $SB20: MOV     2(SP),1$      ;SAVE THE BINARY NUMBER
6744 030124 012746 030142          MOV     #1$,-(SP)    ;SET THE POINTER
6745 030130 004737 033054          JSR     PC,$DB20     ;CALL THE DOUBLE LENGTH CONVERT
6746 030134 012666 000002          MOV     (SP)+,2(SP)  ;PICKUP THE POINTER
6747 030140 000207          RTS      PC          ;RETURN
6748 030142 000000 000000      1$:   .WORD  0,0
6749
6750 ;KEYBOARD INTERRUPT INITIALIZATION ROUTINE
6751 ;CALL
6752 ;       JSR     PC,$TKINT    ;
6753 ;       RETURN
6754
6755 030146 012737 030176 000060 $TKINT: MOV     #$TKSRV,TKVEC ;SETUP VECTOR
6756 030154 012737 000240 000062          MOV     #PR5,TKVEC+2 ;PRIORITY TO 5
6757 030162 005777 150760          TST     @TKB         ;CLEAR THE BUFFER
6758 030166 012777 000100 150750          MOV     #BIT06,@TKS  ;SET INTERRUPT ENABLE
6759 030174 000207          RTS      PC          ;RETURN
6760
6761 ;KEYBOARD INTERRUPT SERVICE ROUTINE
6762 ;CALL
6763 ;       ENTER VIA INTERRUPT
```



```

6764
6765 030176 104410          $TKSRV: RDCHR          ;READ THE KEYBOARD
6766 030200 112637 030326  MOVB      (SP)+,5$      ;GET THE CHARACTER
6767 030204 023727 030326 000003  CMP      5$,#3          ;'CONTROL C' ?
6768 030212 001012          BNE      1$            ;BR IF NOT
6769 030214 104401 001165    TYPE     ,%CRLF        ;CR-LF
6770 030220 104401 030620    TYPE     ,%CNTLC      ;'AC'
6771 030224 012737 177777 001262  MOV      #-1,CFLAG     ;SET THE 'CONTROL C' FLAG
6772 030232 005077 150706    CLR      @%TKS        ;CLEAR THE TTY INTERRUPT
6773 030236 000432          BR       4$            ;EXIT
6774 030240 023727 001140 000176 1$:  CMP      SWR,%SWREG    ;SOFTWARE SWITCH REGISTER IN USE ?
6775 030246 001024          BNE      3$            ;BR IF NOT
6776 030250 023727 030326 000007  CMP      5$,#7          ;'CONTROL G' ?
6777 030256 001020          BNE      3$            ;BR IF NOT
6778 030260 104401 001165    TYPE     ,%CRLF        ;CR-LF
6779 030264 104401 032433    TYPE     ,%CNTLG      ;'AG'
6780 030270 013746 177776    MOV      PS,-(SP)      ;PUT THE STATUS WORD ON THE STACK
6781 030274 012746 030310    MOV      #2$,-(SP)    ;RETURN ADDRESS
6782 030300 005077 150640    CLR      @%TKS        ;CLEAR THE TTY INTERRUPT ENABLE
6783 030304 000137 032074    JMP      %GTSWR       ;GET THE SWITCH REGISTER ENTRY
6784 030310 012777 000100 150626 2$:  MOV      #100,@%TKS    ;ENABLE TTY KEYBOARD INTERRUPT
6785 030316 000402          BR       4$            ;EXIT
6786 030320 104401 030326 3$:  TYPE     ,5$          ;ECHO THE CHARACTER
6787 030324 000002          4$:  RTI              ;RETURN
6788
6789 030326 000000          5$:  .WORD    0          ;ENTERED CHARACTER
6790
6791          ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
6792          ;CALL:
6793          ;      RDLIN          ;:INPUT A STRING FROM THE TTY
6794          ;      RETURN HERE   ;:ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
6795          ;                  ;:TERMINATOR WILL BE A BYTE OF ALL 0'S
6796
6797 030330 010346          $RDLIN: MOV      R3,-(SP) ;SAVE R3
6798 030332 005046          CLR      -(SP)        ;CLEAR THE RUBOUT KEY
6799 030334 012703 030606 1$:  MOV      %$TTYIN,R3   ;GET ADDRESS
6800 030340 022703 030620 2$:  CMP      %$TTYIN+10.,R3 ;BUFFER FULL?
6801 030344 101467          BLOS     4$            ;BR IF YES
6802 030346 104410          RDCHR    ;GO READ ONE CHARACTER FROM THE TTY
6803 030350 112613          MOVB     (SP)+,(R3)   ;GET CHARACTER
6804 030352 122713 000177  CMPB     #177,(R3)    ;IS IT A RUBOUT
6805 030356 001022          BNE      5$            ;BR IF NO
6806 030360 005716          TST     (SP)         ;IS THIS THE FIRST RUBOUT?
6807 030362 001007          BNE      6$            ;BR IF NO
6808 030364 112737 000134 030604  MOVB     #'\",9$      ;TYPE A BACK SLASH
6809 030372 104401 030604          TYPE     ,9$
6810 030376 012716 177777          MOV      #-1,(SP)    ;SET THE RUBOUT KEY
6811 030402 005303          6$:  DEC      R3          ;BACKUP BY ONE
6812 030404 020327 030606          CMP      R3,%$TTYIN  ;STACK EMPTY?
6813 030410 103445          BLO     4$            ;BR IF YES
6814 030412 111337 030604          MOVB     (R3),9$     ;SETUP TO TYPEOUT THE DELETED CHAR.
6815 030416 104401 030604          TYPE     ,9$
6816 030422 000746          BR       2$            ;GO READ ANOTHER CHAR.
6817 030424 005716          5$:  TST     (SP)         ;RUBOUT KEY SET?
6818 030426 001406          BEQ     7$            ;BR IF NO
6819 030430 112737 000134 030604  MOVB     #'\",9$      ;TYPE A BACK SLASH

```



```
6820 030436 104401 030604          TYPE      ,9$
6821 030442 005016                   CLR      (SP)          ;CLEAR THE RUBOUT KEY
6822 030444 122713 000025          7$:      CMPB     #25,(R3) ;IS CHARACTER A CTRL U?
6823 030450 001003                   BNE      10$          ;BR IF NO
6824 030452 104401 032426          TYPE      ,%CNTLU    ;TYPE A CONTROL "U"
6825 030456 000726                   BR       1$          ;GO START OVER
6826 030460 122713 000003          10$:     CMPB     #3,(R3) ;IS CHARACTER A CTRL C ?
6827 030464 001006                   BNE      8$          ;BR IF NOT
6828 030466 012737 177777          001262   MOV      #-1,CFLAG    ;SET CNTRL C FLAG
6829 030474 104401 030620          TYPE      ,%CNTLC    ;ECHO IT
6830 030500 000427                   BR       11$         ;EXIT
6831 030502 122713 000012          8$:      CMPB     #12,(R3) ;IS CHARACTER A "LF"?
6832 030506 001011                   BNE      3$          ;BRANCH IF NO
6833 030510 105013                   CLRB     (R3)        ;CLEAR THE CHARACTER
6834 030512 104401 001165          TYPE      ,%SCLRF    ;TYPE A "CR" & "LF"
6835 030516 104401 030606          TYPE      ,%TTYIN    ;TYPE THE INPUT STRING
6836 030522 000706                   BR       2$          ;GO PICKUP ANOTHER CHACTER
6837 030524 104401 001164          4$:      TYPE      ,%QUES ;TYPE A '?'
6838 030530 000701                   BR       1$          ;CLEAR THE BUFFER AND LOOP
6839 030532 111337 030604          3$:      MOV      (R3),9$ ;ECHO THE CHARACTER
6840 030536 104401 030604          TYPE      ,9$
6841 030542 122723 000015          CMPB     #15,(R3)+  ;CHECK FOR RETURN
6842 030546 001274                   BNE      2$          ;LOOP IF NOT RETURN
6843 030550 105063 177777          CLRB     -1(R3)     ;CLEAR RETURN (THE 15)
6844 030554 104401 001166          TYPE      ,%SLF      ;TYPE A LINE FEED
6845 030560 005726                   11$:     TST      (SP)+  ;CLEAN RUBOUT KEY FROM THE STACK
6846 030562 012603                   MOV      (SP)+,R3   ;RESTORE R3
6847 030564 011646                   MOV      (SP),-(SP) ;ADJUST THE STACK AND PUT ADDRESS OF THE
6848 030566 016666 000004          000002   MOV      4(SP),2(SP) ;FIRST ASCII CHARACTER ON IT
6849 030574 012766 030606          000004   MOV      #$TTYIN,4(SP)
6850 030602 000002                   RTI
6851 030604 000          9$:      .BYTE    0          ;STORAGE FOR ASCII CHAR. TO TYPE
6852 030605 000          .BYTE    0          ;TERMINATOR
6853 030606 000012          $TTYIN:  .BLKB   10.   ;RESERVE 10 BYTES FOR TTY INPUT
6854 030620 041536 005015          000     $CNTLC:  .ASCIZ  / ^C / <CR> <LF> ;CONTROL "C"
6855
6856          030626          .EVEN
6857
6858          ;:*****
6859
6860          .SBTTL  MACRO ROUTINES
6861
6862          .SBTTL  ERROR HANDLER ROUTINE
6863
6864          ;:*****
6865          ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
6866          ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
6867          ;*AND GO TO $ERRTYP ON ERROR
6868          ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
6869          ;*SW15=1          HALT ON ERROR
6870          ;*SW13=1          INHIBIT ERROR TYPEOUTS
6871          ;*SW10=1         BELL ON ERROR
6872          ;*CALL
6873          ;*      ERROR    N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
6874
6875          $ERROR:
```



```
6876 030626 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
6877 030630 010337 001244    MOV           R3,ATTN          ;;SAVE THE ATTENTION REGISTER CONTENTS
6878 030634 010137 001242    MOV           R1,DRIVE         ;;DRIVE NUMBER
6879 030640 032777 020000 150272  BIT           #SW13,@SWR       ;;INHIBIT PRINTOUTS ?
6880 030646 001002          BNE           .+6              ;;BR IF YES
6881 030650 004737 023536    JSR           PC,$TIME         ;;TYPE THE TIME
6882 030654 105237 001103    7$: INCB      $ERFLG           ;;SET THE ERROR FLAG
6883 030660 001775          BEQ           7$              ;;DON'T LET THE FLAG GO TO ZERO
6884 030662 013777 001102 150252  MOV           $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
6885 030670 032777 002000 150242  BIT           #BIT10,@SWR      ;;BELL ON ERROR?
6886 030676 001402          BEQ           1$              ;;NO - SKIP
6887 030700 104401 001160    TYPE         ,SBELL           ;;RING BELL
6888 030704 005237 001112    1$: INC       $ERTTL           ;;COUNT THE NUMBER OF ERRORS
6889 030710 011637 001116    MOV           (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
6890 030714 162737 000002 001116  SUB           #2,$ERRPC
6891 030722 117737 150170 001114  MOVB          @$ERRPC,$ITEMB   ;;STRIP AND SAVE THE ERROR ITEM CODE
6892 030730 032777 020000 150202  BIT           #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
6893 030736 001004          BNE           20$            ;;SKIP TYPEOUTS
6894 030740 004737 030776    JSR           PC,$ERRTP       ;;GO TO USER ERROR ROUTINE
6895 030744 104401 001165    TYPE         ,SCRLF
6896 030750          20$:
6897 030750 005777 150164    2$: TST       @SWR             ;;HALT ON ERROR
6898 030754 100002          BPL           3$             ;;SKIP IF CONTINUE
6899 030756 000000          HALT
6900 030760 104407          CKSWR             ;;TEST FOR CHANGE IN SOFT-SWR
6901 030762          3$:
6902 030762 023737 000042 000046  CMP           @#42,@#46        ;ARE WE IN ACT-11 AUTO MODE?
6903 030770 001001          BNE           .+4            ;BRANCH IF NOT
6904 030772 000000          HALT
6905 030774 000002          RTI                 ;RETURN
```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

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

\$ERRTP:

```
6914 030776          TYPE         ,SCRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
6915 030776 104401 001165    MOV           RO,-(SP)         ;;SAVE RO
6916 031002 010046          CLR           RO              ;;PICKUP THE ITEM INDEX
6917 031004 005000          BISB          @#$ITEMB,RO
6918 031006 153700 001114    BNE           1$              ;;IF ITEM NUMBER IS ZERO, JUST
6919 031012 001004          MOV           $ERRPC,-(SP)   ;;TYPE THE PC OF THE ERROR
6920          ;;SAVE $ERRPC FOR TYPEOUT
6921 031014 013746 001116    TYPE         ,ERRADR         ;;ERROR ADDRESS
6922          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
6923 031020 104402          BR           6$              ;;GET OUT
6924 031022 000426    1$: DEC       RO              ;;ADJUST THE INDEX SO THAT IT WILL
6925 031024 005300          ASL           RO              ;;WORK FOR THE ERROR TABLE
6926 031026 006300          ASL           RO
6927 031030 006300          ASL           RO
6928 031032 006300          ASL           RO
6929 031034 062700 004026    ADD           #$ERRTB,RO      ;;FORM TABLE POINTER
6930 031040 012037 031050    MOV           (RO)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
6931 031044 001404          BEQ           3$              ;;SKIP TYPEOUT IF NO POINTER
```



```

6932 031046 104401          TYPE                                ;;TYPE THE "ERROR MESSAGE"
6933 031050 000000          2$: .WORD 0                                ;;"ERROR MESSAGE" POINTER GOES HERE
6934 031052 104401 001165  TYPE , $CRLF                                ;;"CARRIAGE RETURN" & "LINE FEED"
6935 031056 012037 031066  3$: MOV (R0)+,4$                                ;;PICKUP "DATA HEADER" POINTER
6936 031062 001404          BEQ 5$                                ;;SKIP TYPEOUT IF 0
6937 031064 104401          TYPE                                ;;TYPE THE "DATA HEADER"
6938 031066 000000          4$: .WORD 0                                ;;"DATA HEADER" POINTER GOES HERE
6939 031070 104401 001165  TYPE , $CRLF                                ;;"CARRIAGE RETURN" & "LINE FEED"
6940 031074 011000          5$: MOV (R0),R0                                ;;PICKUP "DATA TABLE" POINTER
6941 031076 001004          BNE 7$                                ;;GO TYPE THE DATA
6942 031100 012600          6$: MOV (SP)+,R0                                ;;RESTORE R0
6943 031102 104401 001165  TYPE , $CRLF                                ;;"CARRIAGE RETURN" & "LINE FEED"
6944 031106 000207          RTS PC                                ;;RETURN
6945 031110          7$:
6946 031110 013046          MOV @ (R0)+, -(SP)                ;;SAVE @ (R0)+ FOR TYPEOUT
6947 031112 104402          TYPOC                                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
6948 031114 005710          TST (R0)                                ;;IS THERE ANOTHER NUMBER?
6949 031116 001770          BEQ 6$                                ;;BR IF NO
6950 031120 104401 031126  TYPE ,8$                                ;;TYPE TWO(2) SPACES
6951 031124 000771          BR 7$                                ;;LOOP
6952 031126 020040 000          8$: .ASCIZ / /                                ;;TWO(2) SPACES
6953 031132 031132          .EVEN
6954
6955          .SBTTL TYPE ROUTINE
6956
6957          ;;*****
6958          ;;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
6959          ;;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
6960          ;;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
6961          ;;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
6962          ;;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
6963          ;;*
6964          ;;*CALL:
6965          ;;*1) USING A TRAP INSTRUCTION
6966          ;;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
6967          ;;*OR
6968          ;;* TYPE
6969          ;;* MESADR
6970          ;;*
6971
6972 031132 105737 001157  $TYPE: TSTB $TPFLG                ;;IS THERE A TERMINAL?
6973 031136 100002          BPL 1$                                ;;BR IF YES
6974 031140 000000          HALT                                ;;HALT HERE IF NO TERMINAL
6975 031142 000407          BR 3$                                ;;LEAVE
6976 031144 010046          1$: MOV R0, -(SP)                                ;;SAVE R0
6977 031146 017600 000002  MOV @2(SP),R0                                ;;GET ADDRESS OF ASCIZ STRING
6978 031152 112046          2$: MOVB (R0)+, -(SP)                ;;PUSH CHARACTER TO BE TYPED ONTO STACK
6979 031154 001005          BNE 4$                                ;;BR IF IT ISN'T THE TERMINATOR
6980 031156 005726          TST (SP)+                                ;;IF TERMINATOR POP IT OFF THE STACK
6981 031160 012600          60$: MOV (SP)+,R0                                ;;RESTORE R0
6982 031162 062716 000002  3$: ADD #2, (SP)                                ;;ADJUST RETURN PC
6983 031166 000002          RTI                                ;;RETURN
6984 031170 122716 000011  4$: CMPB #HT, (SP)                                ;;BRANCH IF <HT>
6985 031174 001430          BEQ 8$                                ;;BRANCH IF NOT <CR LF>
6986 031176 122716 000200  CMPB #CRLF, (SP)
6987 031202 001006          BNE 5$

```



```
6988 031204 005726          TST      (SP)+          ;;POP <CR><LF> EQUIV
6989 031206 104401          TYPE
6990 031210 001165          $CRLF
6991 031212 105037 031346   CLR      $CHARCNT      ;;CLEAR CHARACTER COUNT
6992 031216 000755          BR       2$            ;;GET NEXT CHARACTER
6993 031220 004737 031302   5$:     JSR      PC,$TYPEC  ;;GO TYPE THIS CHARACTER
6994 031224 123726 001156   6$:     CMPB     $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
6995 031230 001350          BNE     2$            ;;IF NO GO GET NEXT CHAR.
6996 031232 013746 001154   MOV     $NULL,-(SP)    ;;GET # OF FILLER CHARS. NEEDED
6997                                ;;AND THE NULL CHAR.
6998 031236 105366 000001   7$:     DECB    1(SP)    ;;DOES A NULL NEED TO BE TYPED?
6999 031242 002770          BLT     6$            ;;BR IF NO--GO POP THE NULL OFF OF STACK
7000 031244 004737 031302   JSR     PC,$TYPEC      ;;GO TYPE A NULL
7001 031250 105337 031346   DECB    $CHARCNT      ;;DO NOT COUNT AS A COUNT
7002 031254 000770          BR      7$            ;;LOOP
```

;HORIZONTAL TAB PROCESSOR

```
7003
7004
7005
7006 031256 112716 000040   8$:     MOV     #' ,(SP)  ;;REPLACE TAB WITH SPACE
7007 031262 004737 031302   9$:     JSR     PC,$TYPEC  ;;TYPE A SPACE
7008 031266 132737 000007 031346   BITB    #7,$CHARCNT    ;;BRANCH IF NOT AT
7009 031274 001372          BNE     9$            ;;TAB STOP
7010 031276 005726          TST     (SP)+         ;;POP SPACE OFF STACK
7011 031300 000724          BR      2$            ;;GET NEXT CHARACTER
7012 031302 105777 147642   $TYPEC: TSTB    @ $TPS   ;;WAIT UNTIL PRINTER IS READY
7013 031306 100375          BPL     $TYPEC
7014 031310 116677 000002 147634   MOV     2(SP),@ $TPB   ;;LOAD CHAR TO BE TYPED INTO DATA REG.
7015 031316 122766 000015 000002   CMPB    #CR,2(SP)     ;;IS CHARACTER A CARRIAGE RETURN?
7016 031324 001003          BNE     1$            ;;BRANCH IF NO
7017 031326 105037 031346   CLR     $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
7018 031332 000406          BR     $TYPEX         ;;EXIT
7019 031334 122766 000012 000002 1$:     CMPB    #LF,2(SP)    ;;IS CHARACTER A LINE FEED?
7020 031342 001402          BEQ    $TYPEX         ;;BRANCH IF YES
7021 031344 105227          INCB   (PC)+         ;;COUNT THE CHARACTER
7022 031346 000000          $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
7023 031350 000207          $TYPEX: RTS      PC
```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```
7024
7025
7026
7027
7028
7029
7030
7031
7032
7033
7034
7035
7036
7037
7038
7039
7040
7041
7042
7043
```

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



```
7100 031574 000 $OCNT: .BYTE 0 ;;OCTAL DIGIT COUNTER
7101 031575 000 $OFILL: .BYTE 0 ;;ZERO FILL SWITCH
7102 031576 000000 $OMODE: .WORD 0 ;;NUMBER OF DIGITS TO TYPE
7103
7104 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
7105
7106 ::*****
7107 ::*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
7108 ::*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
7109 ::*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
7110 ::*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
7111 ::*REPLACED WITH SPACES.
7112 ::*CALL:
7113 ::* MOV NUM,-(SP) ;;PUT THE BINARY NUMBER ON THE STACK
7114 ::* TYPDS ;;GO TO THE ROUTINE
7115
7116 $TYPDS:
7117 031600 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
7118 031602 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
7119 031604 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
7120 031606 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
7121 031610 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
7122 031612 012746 020200 MOV #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
7123 031616 016605 000020 MOV 20(SP),R5 ;;GET THE INPUT NUMBER
7124 031622 100004 BPL 1$ ;;BR IF INPUT IS POS.
7125 031624 005405 NEG R5 ;;MAKE THE BINARY NUMBER POS.
7126 031626 112766 000055 000001 MOVB #'-,1(SP) ;;MAKE THE ASCII NUMBER NEG.
7127 031634 005000 1$: CLR R0 ;;ZERO THE CONSTANTS INDEX
7128 031636 012703 032014 MOV #DBLK,R3 ;;SETUP THE OUTPUT POINTER
7129 031642 112723 000040 MOVB #' ,(R3)+ ;;SET THE FIRST CHARACTER TO A BLANK
7130 031646 005002 2$: CLR R2 ;;CLEAR THE BCD NUMBER
7131 031650 016001 032004 MOV $DTBL(R0),R1 ;;GET THE CONSTANT
7132 031654 160105 3$: SUB R1,R5 ;;FORM THIS BCD DIGIT
7133 031656 002402 BLT 4$ ;;BR IF DONE
7134 031660 005202 INC R2 ;;INCREASE THE BCD DIGIT BY 1
7135 031662 000774 BR 3$
7136 031664 060105 4$: ADD R1,R5 ;;ADD BACK THE CONSTANT
7137 031666 005702 TST R2 ;;CHECK IF BCD DIGIT=0
7138 031670 001002 BNE 5$ ;;FALL THROUGH IF 0
7139 031672 105716 TSTB (SP) ;;STILL DOING LEADING 0'S?
7140 031674 100407 BMI 7$ ;;BR IF YES
7141 031676 106316 5$: ASLB (SP) ;;MSD?
7142 031700 103003 BCC 6$ ;;BR IF NO
7143 031702 116663 000001 177777 MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
7144 031710 052702 000060 6$: BIS #'0,R2 ;;MAKE THE BCD DIGIT ASCII
7145 031714 052702 000040 7$: BIS #' ,R2 ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
7146 031720 110223 MOVB R2,(R3)+ ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
7147 031722 005720 TST (R0)+ ;;JUST INCREMENTING
7148 031724 020027 000010 CMP R0,#10 ;;CHECK THE TABLE INDEX
7149 031730 002746 BLT 2$ ;;GO DO THE NEXT DIGIT
7150 031732 003002 BGT 8$ ;;GO TO EXIT
7151 031734 010502 MOV R5,R2 ;;GET THE LSD
7152 031736 000764 BR 6$ ;;GO CHANGE TO ASCII
7153 031740 105726 8$: TSTB (SP)+ ;;WAS THE LSD THE FIRST NON-ZERO?
7154 031742 100003 BPL 9$ ;;BR IF NO
7155 031744 116663 177777 177776 MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
```



```
7156 031752 105013          9$:  CLRB   (R3)           ;;SET THE TERMINATOR
7157 031754 012605          MOV    (SP)+,R5         ;;POP STACK INTO R5
7158 031756 012603          MOV    (SP)+,R3         ;;POP STACK INTO R3
7159 031760 012602          MOV    (SP)+,R2         ;;POP STACK INTO R2
7160 031762 012601          MOV    (SP)+,R1         ;;POP STACK INTO R1
7161 031764 012600          MOV    (SP)+,R0         ;;POP STACK INTO R0
7162 031766 104401 032014    TYPE    $DBLK           ;;NOW TYPE THE NUMBER
7163 031772 016666 000002 000004  MOV    2(SP),4(SP)      ;;ADJUST THE STACK
7164 032000 012616          MOV    (SP)+,(SP)
7165 032002 000002          RTI                    ;;RETURN TO USER
7166 032004 023420          $DTBL: 10000.
7167 032006 001750          1000.
7168 032010 000144          100.
7169 032012 000012          10.
7170 032014 000004          $DBLK: .BLKW 4
7171
7172          .SBTTL  TTY INPUT ROUTINE
7173
7174          ;:*****
7175          .ENABL  LSB
7176
7177          ;:*****
7178          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
7179          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
7180          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
7181          ;*WHEN OPERATING IN TTY FLAG MODE.
7182 032024 022737 000176 001140  $CKSWR: CMP    #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
7183 032032 001074          BNE    15$              ;;BRANCH IF NO
7184 032034 105777 147104          TSTB   @STKS            ;;CHAR THERE?
7185 032040 100071          BPL    15$              ;;IF NO, DON'T WAIT AROUND
7186 032042 117746 147100          MOVB   @STKB,-(SP)      ;;SAVE THE CHAR
7187 032046 042716 177600          BIC    #^C177,(SP)     ;;STRIP-OFF THE ASCII
7188 032052 022726 000007          CMP    #7,(SP)+        ;;IS IT A CONTROL G?
7189 032056 001062          BNE    15$              ;;NO, RETURN TO USER
7190 032060 123727 001134 000001  CMPB   $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
7191 032066 001456          BEQ    15$              ;;BRANCH IF YES
7192
7193 032070 104401 032433          $GTSWR: TYPE    $CNTLG      ;;ECHO THE CONTROL-G (^G)
7194 032074 104401 032440          TYPE    $MSWR          ;;TYPE CURRENT CONTENTS
7195 032100 013746 000176          MOV    SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
7196 032104 104402          TYPOC           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7197 032106 104401 032451          TYPE    $MNEW          ;;PROMPT FOR NEW SWR
7198 032112 005046          19$:  CLR    -(SP)         ;;CLEAR COUNTER
7199 032114 005046          CLR    -(SP)         ;;THE NEW SWR
7200 032116 105777 147022          7$:  TSTB   @STKS            ;;CHAR THERE?
7201 032122 100375          BPL    7$             ;;IF NOT TRY AGAIN
7202
7203 032124 117746 147016          MOVB   @STKB,-(SP)     ;;PICK UP CHAR
7204 032130 042716 177600          BIC    #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
7205
7206
7207
7208 032134 021627 000025          9$:  CMP    (SP),#25       ;;IS IT A CONTROL-U?
7209 032140 001005          BNE    10$            ;;BRANCH IF NOT
7210 032142 104401 032426          TYPE    $CNTLU         ;;YES, ECHO CONTROL-U (^U)
7211 032146 062706 000006          20$: ADD    #6,SP        ;;IGNORE PREVIOUS INPUT
```



```
7212 032152 000757 BR 19$ ::LET'S TRY IT AGAIN
7213
7214
7215 032154 021627 000015 10$: CMP (SP),#15 ::IS IT A <CR>?
7216 032160 001022 BNE 16$ ::BRANCH IF NO
7217 032162 005766 000004 TST 4(SP) ::YES, IS IT THE FIRST CHAR?
7218 032166 001403 BEQ 11$ ::BRANCH IF YES
7219 032170 016677 000002 146742 MOV 2(SP),@SWR ::SAVE NEW SWR
7220 032176 062706 000006 11$: ADD #6,SP ::CLEAR UP STACK
7221 032202 104401 001165 14$: TYPE $CRLF ::ECHO <CR> AND <LF>
7222 032206 123727 001135 000001 CMPB $INTAG,#1 ::RE-ENABLE TTY KBD INTERRUPTS?
7223 032214 001003 BNE 15$ ::BRANCH IF NOT
7224 032216 012777 000100 146720 MOV #100,@$TKS ::RE-ENABLE TTY KBD INTERRUPTS
7225 032224 000002 15$: RTI ::RETURN
7226 032226 004737 031302 16$: JSR PC,$TYPEC ::ECHO CHAR
7227 032232 021627 000060 CMP (SP),#60 ::CHAR < 0?
7228 032236 002420 BLT 18$ ::BRANCH IF YES
7229 032240 021627 000067 CMP (SP),#67 ::CHAR > 7?
7230 032244 003015 BGT 18$ ::BRANCH IF YES
7231 032246 042726 000060 BIC #60,(SP)+ ::STRIP-OFF ASCII
7232 032252 005766 000002 TST 2(SP) ::IS THIS THE FIRST CHAR
7233 032256 001403 BEQ 17$ ::BRANCH IF YES
7234 032260 006316 ASL (SP) ::NO, SHIFT PRESENT
7235 032262 006316 ASL (SP) :: CHAR OVER TO MAKE
7236 032264 006316 ASL (SP) :: ROOM FOR NEW ONE.
7237 032266 005266 000002 17$: INC 2(SP) ::KEEP COUNT OF CHAR
7238 032272 056616 177776 BIS -2(SP),(SP) ::SET IN NEW CHAR
7239 032276 000707 BR 7$ ::GET THE NEXT ONE
7240 032300 104401 001164 18$: TYPE $QUES ::TYPE ?<CR><LF>
7241 032304 000720 BR 20$ ::SIMULATE CONTROL-U
7242 .DSABL LSB
```

7243
7244
7245

```
::*****
::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
::
```

```
7252
7253 032306 011646 $RDCHR: MOV (SP),-(SP) ::PUSH DOWN THE PC
7254 032310 016666 000004 000002 MOV 4(SP),2(SP) ::SAVE THE PS
7255 032316 105777 146622 1$: TSTB @$TKS ::WAIT FOR
7256 032322 100375 BPL 1$ ::A CHARACTER
7257 032324 117766 146616 000004 MOVB @$TKB,4(SP) ::READ THE TTY
7258 032332 042766 177600 000004 BIC #^C<177>,4(SP) ::GET RID OF JUNK IF ANY
7259 032340 026627 000004 000023 CMP 4(SP),#23 ::IS IT A CONTROL-S?
7260 032346 001013 BNE 3$ ::BRANCH IF NO
7261 032350 105777 146570 2$: TSTB @$TKS ::WAIT FOR A CHARACTER
7262 032354 100375 BPL 2$ ::LOOP UNTIL ITS THERE
7263 032356 117746 146564 MOVB @$TKB,-(SP) ::GET CHARACTER
7264 032362 042716 177600 BIC #^C177,(SP) ::MAKE IT 7-BIT ASCII
7265 032366 022627 000021 CMP (SP)+,#21 ::IS IT A CONTROL-Q?
7266 032372 001366 BNE 2$ ::IF NOT DISCARD IT
7267 032374 000750 BR 1$ ::YES, RESUME
```



```

7268 032376 026627 000004 000140 3$:    CMP      4(SP),#140      ;; IS IT UPPER CASE?
7269 032404 002407                BLT      4$              ;; BRANCH IF YES
7270 032406 026627 000004 000175        CMP      4(SP),#175      ;; IS IT A SPECIAL CHAR?
7271 032414 003003                BGT      4$              ;; BRANCH IF YES
7272 032416 042766 000040 000004        BIC      #40,4(SP)       ;; MAKE IT UPPER CASE
7273 032424 000002                4$:    RTI                ;; GO BACK TO USER
7274 032426 052536 005015      000  $CNTLU: .ASCIZ / ^U / <15><12>  ;; CONTROL "U"
7275 032433      136 006507 000012  $CNTLG: .ASCIZ / ^G / <15><12>  ;; CONTROL "G"
7276 032440 005015 053523 020122  $MSWR:  .ASCIZ <15><12> / SWR = /
7277 032446 020075      000
7278 032451      040 047040 053505  $MNEW:  .ASCIZ / NEW = /
7279 032456 036440 000040
7280
7281      .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
7282
7283      ;; *****
7284      ;; *THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
7285      ;; *WITH A RANGE OF 0 TO 2(+33)-1.
7286      ;; *CALL:
7287      ;; *      JSR      PC,$RAND      ;; CALL THE ROUTINE
7288      ;; *      RETURN      ;; RETURN HERE THE RANDOM
7289      ;; *      ;; NUMBER WILL BE IN
7290      ;; *      ;; $HINUM,$LONUM
7291
7292      $RAND:
7293      MOV      R0,-(SP)      ;; PUSH R0 ON STACK
7294      MOV      R1,-(SP)      ;; PUSH R1 ON STACK
7295      MOV      R2,-(SP)      ;; PUSH R2 ON STACK
7296      MOV      $LONUM,R0     ;; SET R0 WITH LOW
7297      MOV      $HINUM,R1     ;; SET R1 WITH HIGH
7298      MOV      #-7,R2        ;; SET SHIFT COUNT
7299      1$:    ASL      R0      ;; SHIFT R0 LEFT AND
7300      ROL      R1      ;; ROTATE CARRY INTO R1 AND
7301      INC      R2      ;; CHECK FOR DONE
7302      BNE     1$        ;; CONTINUE SHIFT LOOP
7303      ADD     $LONUM,R0     ;; ADD NUMBER TO MAKE X 129
7304      ADC     R1          ;; PROPOGATE CARRY
7305      ADD     $HINUM,R1     ;; ADD NUMBER TO MAKE X 129
7306      ADD     #1057,R0     ;; ADD LOW CONSTANT
7307      ADC     R1          ;; PROPOGATE CARRY
7308      ADD     #47401,R1     ;; ADD HIGH CONSTANT
7309      MOV     R0,$LONUM     ;; SAVE R0
7310      MOV     R1,$HINUM     ;; SAVE R1
7311      MOV     (SP)+,R2      ;; POP STACK INTO R2
7312      MOV     (SP)+,R1      ;; POP STACK INTO R1
7313      MOV     (SP)+,R0      ;; POP STACK INTO R0
7314      RTS     PC           ;; RETURN
7315      $HINUM: .WORD 176543
7316      $LONUM: .WORD 123456
7317
7318      .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
7319
7320      ;; *****
7321      ;; *SAVE R0-R5
7322      ;; *CALL:
7323      ;; *      SAVREG

```



```
7324 ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
7325 ;*
7326 ;*TOP---(+16)
7327 ;* +2---(+18)
7328 ;* +4---R5
7329 ;* +6---R4
7330 ;* +8---R3
7331 ;*+10---R2
7332 ;*+12---R1
7333 ;*+14---R0
7334
7335 032564 $SAVREG:
7336 032564 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
7337 032566 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
7338 032570 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
7339 032572 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
7340 032574 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
7341 032576 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
7342 032600 016646 000022 MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
7343 032604 016646 000022 MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
7344 032610 016646 000022 MOV 22(SP),-(SP) ;;SAVE PS OF CALL
7345 032614 016646 000022 MOV 22(SP),-(SP) ;;SAVE PC OF CALL
7346 032620 000002 RTI
7347
7348 ;*RESTORE R0-R5
7349 ;*CALL:
7350 ;* RESREG
7351 032622 $RESREG:
7352 032622 012666 000022 MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
7353 032626 012666 000022 MOV (SP)+,22(SP) ;;RESTORE PS OF CALL
7354 032632 012666 000022 MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
7355 032636 012666 000022 MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW
7356 032642 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
7357 032644 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
7358 032646 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
7359 032650 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
7360 032652 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
7361 032654 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
7362 032656 000002 RTI
7363
7364 .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
7365
7366 ;*****
7367 ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
7368 ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
7369 ;*POSITIVE.
7370 ;*CALL
7371 ;* MOV #PNTR,-(SP) ;;POINTER TO LOW WORD OF BINARY NUMBER
7372 ;* JSR PC,@#$DB2D
7373 ;* RETURN ;;THE FIRST ADDRESS OF ASCII
7374 ;* ;;IS ON THE STACK
7375
7376
7377 032660 104412 $DB2D: SAVREG ;;SAVE REGISTERS
7378 032662 016602 000002 MOV 2(SP),R2 ;;PICKUP THE DATA POINTER
7379 032666 012700 033040 MOV #$DECVL,R0 ;;GET ADDRESS OF "$DECVL" STRING
```



```
7380 032672 010066 000002      MOV      R0,2(SP)      ;;PUT ADDRESS OF ASCIZ STRING ON STACK
7381 032676 012201      MOV      (R2)+,R1     ;;PICKUP THE BINARY NUMBER
7382 032700 012202      MOV      (R2)+,R2
7383 032702 012737 000012 032756  MOV      #10.,4$      ;;SET UP TO DO 10 CONVERSIONS
7384 032710 012704 032770  MOV      #$TNPWR,R4   ;;ADDRESS OF TEN POWER
7385 032714 012705 032772  MOV      #$TNPWR+2,R5
7386 032720 005003      1$: CLR      R3        ;;CLEAR PARTIAL
7387 032722 161401      2$: SUB      (R4),R1   ;;SUBTRACT TEN POWER
7388 032724 005602      SBC      R2
7389 032726 161502      SUB      (R5),R2
7390 032730 002402      BLT      3$          ;;BR IF TEN POWER TO LARGE
7391 032732 005203      INC      R3          ;;ADD 1 TO PARTIAL
7392 032734 000772      BR       2$          ;;LOOP
7393 032736 062401      3$: ADD      (R4)+,R1 ;;RESTORE SUBTRACTED VALUE
7394 032740 005502      ADC      R2
7395 032742 062402      ADD      (R4)+,R2
7396 032744 022525      CMP      (R5)+,(R5)+ ;;MOVE TO NEXT TEN POWER
7397 032746 052703 000060  BIS      #'0,R3      ;;CHANGE PARTIAL TO ASCII
7398 032752 110320      MOV      R3,(R0)+   ;;SAVE IT
7399 032754 005327      DEC      (PC)+     ;;DONE?
7400 032756 000000      4$: .WORD    0
7401 032760 001357      BNE      1$          ;;BR IF NO
7402 032762 105020      CLRB    (R0)+     ;;TERMINATOR
7403 032764 104413      RESREG   ;;RESTORE REGISTERS
7404 032766 000207      RTS      PC        ;;RETURN
7405 032770 145000      $TNPWR: 145000     ;;1.0E09
7406 032772 035632      35632
7407 032774 160400      160400           ;;1.0E08
7408 032776 002765      2765
7409 033000 113200      113200           ;;1.0E07
7410 033002 000230      230
7411 033004 041100      041100           ;;1.0E06
7412 033006 000017      17
7413 033010 103240      103240           ;;1.0E05
7414 033012 000001      1
7415 033014 023420      23420           ;;1.0E04
7416 033016 000000      0
7417 033020 001750      1750            ;;1.0E03
7418 033022 000000      0
7419 033024 000144      144             ;;1.0E02
7420 033026 000000      0
7421 033030 000012      12             ;;1.0E01
7422 033032 000000      0
7423 033034 000001      1             ;;1.0E00
7424 033036 000000      0
7425 033040 000014      $DECVL: .BLKB 12. ;;RESERVE STORAGE FOR ASCIZ STRING
7426
7427      .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
7428
7429      ;*****
7430      ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
7431      ;*UNSIGNED OCTAL ASCII NUMBER.
7432      ;*CALL
7433      ;*      MOV      #PNTR,-(SP)  ;;POINTER TO LOW WORD OF BINARY NUMBER
7434      ;*      JSR      PC,@#$DB20  ;;CALL THE ROUTINE
7435      ;*      RETURN  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
```



```
7436
7437
7438 033054 104412
7439 033056 016601 000002
7440 033062 012705 033173
7441 033066 012704 000014
7442 033072 012703 177770
7443 033076 012100
7444 033100 012101
7445 033102 005002
7446 033104 110245
7447 033106 010002
7448 033110 005304
7449 033112 003007
7450 033114 001405
7451 033116 005205
7452 033120 010566 000002
7453 033124 104413
7454 033126 000207
7455 033130 006203
7456 033132 006001
7457 033134 006000
7458 033136 006001
7459 033140 006000
7460 033142 006001
7461 033144 006000
7462 033146 040302
7463 033150 062702 000060
7464 033154 000753
7465 033156 000016
7466
7467
7468
7469
7470
7471
7472
7473
7474
7475 033174 010046
7476 033176 016600 000002
7477 033202 005740
7478 033204 111000
7479 033206 006300
7480 033210 016000 033230
7481 033214 000200
7482
7483
7484
7485
7486 033216 011646
7487 033220 016666 000004 000002
7488 033226 000002
7489
7490
7491

$DB20: SAVREG          ;;SAVE ALL REGISTERS
MOV      2(SP),R1      ;;PICKUP THE POINTER TO LOW WORD
MOV      #SOCTVL+13.,R5 ;;POINTER TO DATA TABLE
MOV      #12.,R4       ;;DO ELEVEN CHARACTERS
MOV      #^C7,R3       ;;MASK
MOV      (R1)+,R0      ;;LOWER WORD
MOV      (R1)+,R1      ;;HIGH WORD
CLR      R2            ;;TERMINATOR
1$:      MOVB          R2,-(R5) ;;PUT CHARACTER IN DATA TABLE
MOV      R0,R2        ;;GET THIS DIGIT
DEC      R4           ;;COUNT THIS CHARACTER
BGT      3$          ;;BR IF NOT THE LAST DIGIT
BEQ      2$          ;;BR IF IT IS THE LAST DIGIT
INC      R5           ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
MOV      R5,2(SP)    ;;ASCIZ CHAR. & PUT IT ON THE STACK
RESREG
RTS      PC          ;;RESTORE ALL REGISTERS
2$:      ASR          R3      ;;RETURN TO USER
3$:      ROR          R3      ;;POSITION THE MASK FOR THE LAST DIGIT
ROR      R1          ;;POSITION THE BINARY NUMBER FOR
ROR      R0          ;;THE NEXT OCTAL DIGIT
ROR      R1
ROR      R0
ROR      R1
ROR      R0
BIC      R3,R2       ;;MASK OUT ALL JUNK
ADD      #'0,R2      ;;MAKE THIS CHAR. ASCII
BR       1$          ;;GO PUT IT IN THE DATA TABLE
$OCTVL: .BLKB        14.   ;;RESERVE DATA TABLE

.SBTTL TRAP DECODER

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

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

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

$TRAP2: MOV      (SP),-(SP) ;;MOVE THE PC DOWN
MOV      4(SP),2(SP)    ;;MOVE THE PSW DOWN
RTI
.SBTTL TRAP TABLE
```


7492
7493
7494
7495
7496
7497 033230 033216
7498 033232 031132
7499 033234 031376
7500 033236 031352
7501 033240 031412
7502 033242 031600
7503
7504 033244 032074
7505
7506 033246 032024
7507 033250 032306
7508 033252 030330
7509 033254 032564
7510 033256 032622
7511 033260 027554
7512 000032
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532 033262 000000 000000 000000
7533 033270 000000
7534
7535
7536
7537
7538
7539
7540 033272 000
7541 033273 000
7542 033274 000
7543 033275 000
7544 033276 000
7545 033277 000
7546 033300 000
7547 033301 000

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

```
ROUTINE
-----
$TRPAD: .WORD $TRAP2
$TYPE      ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
$TYPOC     ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS     ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
$TYPON     ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS     ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

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

$CKSWR     ;;CALL=CKSWR     TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
$RDCHR     ;;CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
$RDLIN     ;;CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
$SAVREG    ;;CALL=SAVREG    TRAP+12(104412) SAVE R0-R5 ROUTINE
$RESREG    ;;CALL=RESREG    TRAP+13(104413) RESTORE R0-R5 ROUTINE
$DSPLY     ;;CALL=DISPLY    TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES

$TERM=.-$TRPAD
```

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

;COPYRIGHT (C) 1976,1979
;DIGITAL EQUIPMENT CORP.
;MAYNARD, MA 01754

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

RPERRS: .WORD 0,0,0,0

;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

```
DRVACT: .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
```


7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7600
7601
7602
7603

033302 000
033303 000
033304 000
033305 000
033306 000
033307 000
033310 000
033311 000

033312 000
033313 000
033314 000
033315 000
033316 000
033317 000
033320 000
033321 000

033322 000
033323 000
033324 000
033325 000
033326 000
033327 000
033330 000
033331 000

033332 000
033333 000
033334 000
033335 000
033336 000
033337 000
033340 000
033341 000

;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
;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
.BYTE 0 ;DRIVE 7


```
7604  
7605 ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)  
7606 ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF  
7607 ;"DPB" OF THE I/O OPERATION.  
7608  
7609 033342 000000 TRNSWT: .WORD 0  
7610  
7611 ;SEARCH WAIT KEYS (SRCHWT=1 WORD)  
7612 ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF  
7613 ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O  
7614 ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.  
7615 ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.  
7616  
7617 033344 000000 SRCHWT: .WORD 0  
7618  
7619  
7620 ;RP04/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)  
7621 ;ACTDRV=0 IF DRIVER IS INACTIVE  
7622 ;ACTDRV>0 IF DRIVER IS ACTIVE  
7623  
7624 033346 000 ACTDRV: .BYTE 0  
7625  
7626 ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)  
7627 ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE  
7628 ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE  
7629  
7630 033347 000 ACTSTR: .BYTE 0  
7631  
7632 ;UNLOAD FLAG (ULDFLG=8 BYTES)  
7633 ;ULDFLG=0 IF NO UNLOAD COMMAND  
7634 ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS  
7635 ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE  
7636  
7637 033350 000 ULDFLG: .BYTE 0 ;DRIVE 0  
7638 033351 000 .BYTE 0 ;DRIVE 1  
7639 033352 000 .BYTE 0 ;DRIVE 2  
7640 033353 000 .BYTE 0 ;DRIVE 3  
7641 033354 000 .BYTE 0 ;DRIVE 4  
7642 033355 000 .BYTE 0 ;DRIVE 5  
7643 033356 000 .BYTE 0 ;DRIVE 6  
7644 033357 000 .BYTE 0 ;DRIVE 7  
7645  
7646 ;LOOK AHEAD COUNT (LACNT=8 BYTES)  
7647 ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED  
7648  
7649 033360 000 LACNT: .BYTE 0 ;DRIVE 0  
7650 033361 000 .BYTE 0 ;DRIVE 1  
7651 033362 000 .BYTE 0 ;DRIVE 2  
7652 033363 000 .BYTE 0 ;DRIVE 3  
7653 033364 000 .BYTE 0 ;DRIVE 4  
7654 033365 000 .BYTE 0 ;DRIVE 5  
7655 033366 000 .BYTE 0 ;DRIVE 6  
7656 033367 000 .BYTE 0 ;DRIVE 7  
7657  
7658 ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)  
7659 ;SAVEFG <0 IF SAVE THE RH11/RP04/5/6 REGISTERS WHEN THE
```



```
7660 ;OPERATION IS COMPLETED AS PER (DPB+14).
7661 ;SAVEFG=0 IF SAVE THE RH11/RP04/5/6 REGISTERS, AS PER
7662 ;(DPB+14), AFTER AN ERROR.
7663
7664 033370 000000 SAVEFG: .WORD 0
7665
7666 ;SEEK FLAG (SEEKFG=1 WORD)
7667 ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
7668 ;FOR A DATA TRANSFER START A SEARCH COMMAND
7669 ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
7670 ;DISREGARD THE WINDOW
7671
7672 033372 000000 SEEKFG: .WORD 0
7673
7674 ;TIMEOUT TABLE (TIMER=8 WORDS)
7675 ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
7676
7677 033374 177777 TIMER: .WORD -1 ;DRIVE 0
7678 033376 177777 .WORD -1 ;DRIVE 1
7679 033400 177777 .WORD -1 ;DRIVE 2
7680 033402 177777 .WORD -1 ;DRIVE 3
7681 033404 177777 .WORD -1 ;DRIVE 4
7682 033406 177777 .WORD -1 ;DRIVE 5
7683 033410 177777 .WORD -1 ;DRIVE 6
7684 033412 177777 .WORD -1 ;DRIVE 7
7685
7686 ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
7687 ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
7688 ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
7689
7690 033414 177777 DTUW: .WORD -1
7691
7692 ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
7693 ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
7694 ;ATTENTION BIT
7695
7696 033416 001 ATABIT: .BYTE 1 ;DRIVE 0
7697 033417 002 .BYTE 2 ;DRIVE 1
7698 033420 004 .BYTE 4 ;DRIVE 2
7699 033421 010 .BYTE 10 ;DRIVE 3
7700 033422 020 .BYTE 20 ;DRIVE 4
7701 033423 040 .BYTE 40 ;DRIVE 5
7702 033424 100 .BYTE 100 ;DRIVE 6
7703 033425 200 .BYTE 200 ;DRIVE 7
7704
7705 ;RP04/5/6 TO RH11 'MASSBUS CONTROL BUS PARITY ERRORS' (MCPE) ALLOWED BEFORE
7706 ;CALLING IT FATAL (MCPEMX=1 WORD)
7707
7708 033426 000003 MCPEMX: .WORD 3
7709
7710 ;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RP04/5/6),
7711 ;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).
7712
7713 033430 176700 RPADR: .WORD 176700
7714 033432 000254 000240 RPVEC: .WORD 254,5*32.
7715
```



```
7716 ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
7717
7718 033436 000004 MXLACT: .WORD 4
7719 ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
7720
7721 033440 001000 MXDLTA: .WORD 8.*64.
7722 ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
7723
7724 033442 000200 MNDLTA: .WORD 2*64.
7725 ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
7726
7727 033444 000005 MXWNDW: .WORD 5
7728
7729 ;DEFINITIONS OF THE RH11/RP04/5/6 ADDRESS INDEXES
7730
7731 000000 RPCS1=0 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
7732 000002 RPWC=2 ;WORD COUNT REGISTER (NOT A DRIVE REG)
7733 000004 RPBA=4 ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
7734 000006 RPDA=6 ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
7735 000010 RPCS2=10 ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
7736 000012 RPDS1=12 ;DRIVE STATUS REGISTER (DRIVE REG 01)
7737 000014 RPER1=14 ;ERROR REGISTER #1 (DRIVE REG. 02)
7738 000016 RPAS=16 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
7739 000020 RPLA=20 ;LOOK AHEAD REGISTER (DRIVE REG. 07)
7740 000022 RPDB=22 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
7741 000024 RPMR=24 ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
7742 000026 RPDT=26 ;DRIVE TYPE REGISTER (DRIVE REG. 06)
7743 000030 RPSN=30 ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
7744 000032 RPOF=32 ;OFFSET REGISTER (DRIVE REG. 11)
7745 000034 RPCA=34 ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
7746 000036 RPCC=36 ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
7747 000040 RPER2=40 ;ERROR REGISTER #2 (DRIVE REG. 14)
7748 000042 RPER3=42 ;ERROR REGISTER #3 (DRIVE REG. 15)
7749 000044 RPEC1=44 ;ECC POSITION REGISTER (DRIVE REG. 16)
7750 000046 RPEC2=46 ;ECC PATTERN REGISTER (DRIVE REG. 17)
7751
7752 ;RH11/RP04/5/6 DRIVER INITIALIZATION CODE
7753 ;THIS ROUTINE WILL DETERMINE WHICH RP04/5/6 DRIVES ARE
7754 ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
7755 ;TO THE PROPER STATE FOR EACH DRIVE.
7756 ;NOTE: THIS ROUTINE CALLS DRVINT
7757
7758 ;CALL
7759
7760 ; JSR PC,RPINIT
7761 ; RETURN
7762
7763 ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
7764
7765 033446 104412 RPINIT: SAVREG ;SAVE R0 - R5
7766 033450 013746 177776 MOV @#PS,-(SP) ;SAVE THE PRESENT PROCESSOR STATUS
7767 033454 012737 000240 177776 MOV #<5*32.>,@#PS ;CHANGE THE PRIORITY TO 5
7768 033462 004737 041600 JSR PC,CLRQUE ;CLEAR ALL REQUEST QUEUES
7769 033466 012701 033262 MOV #RPERRS,R1 ;FIRST ADDRESS TO BE CLEARED
7770 033472 012702 033372 MOV #SEEKFG,R2 ;LAST ADDRESS TO BE CLEARED
7771 033476 005021 1$: CLR (R1)+ ;CLEAR
```



```
7772 033500 020102          CMP      R1,R2          ;ARE WE DONE?
7773 033502 101775          BLOS     1$             ;BRANCH IF NO
7774 033504 012702 033414     MOV      #DTUW,R2       ;LAST ADDRESS
7775 033510 012721 177777     2$:     MOV      #-1,(R1)+    ;INITIALIZE
7776 033514 020102          CMP      R1,R2          ;DONE?
7777 033516 101774          BLOS     2$             ;LOOP IF NO
7778 033520 005037 033302     CLR      DRVSTA         ;SET ALL DRIVES TO OFFLINE
7779 033524 005037 033304     CLR      DRVSTA+2
7780 033530 005037 033306     CLR      DRVSTA+4
7781 033534 005037 033310     CLR      DRVSTA+6
7782 033540 013703 033432     MOV      RPVEC,R3       ;SETUP THE RH11/RP04/5/6 VECTOR
7783 033544 012723 036362     MOV      #ISR,(R3)+
7784 033550 013713 033434     MOV      RPVEC+2,(R3)
7785 033554 013704 033430     MOV      RPADR,R4       ;FIRST ADDRESS OF RH11/RP04
7786 033560 012764 000040 000010  MOV      #BIT05,RPCS2(R4) ;MASSBUS INIT
7787 033566 005001          CLR      R1             ;START WITH DRIVE 0
7788 033570 004037 033660     3$:     JSR      RO,DRVINT      ;INIT THE DRIVE
7789 033574 000401          BR       4$             ;'DVA' NOT SET OR PARITY ERROR
7790 033576 000402          BR       5$             ;NORMAL RETURN
7791 033600 105061 033302     4$:     CLR      DRVSTA(R1)    ;SET DRIVE STATUS TO OFFLINE
7792 033604 005201 5$:     INC      R1             ;GO TO NEXT DRIVE
7793 033606 042701 177770     BIC      #^C7,R1        ;MASK OUT UNUSED BITS
7794 033612 001366          BNE      3$             ;BR IF MORE DRIVES TO GO
7795 033614 012701 000007     MOV      #7,R1         ;START WITH DRIVE 7
7796 033620 005037 177776     CLR      @#PS           ;CLEAR THE PROCESSOR STATUS
7797 033624 105761 033322     6$:     TSTB     DPINT(R1)     ;WAITING FOR DRIVE TO SWITCH PORTS ?
7798 033630 001405          BEQ      8$             ;BR NOT WAITING
7799 033632 004737 041234     JSR      PC,SET.IE      ;SET INTERRUPT
7800 033636 105761 033322     7$:     TSTB     DPINT(R1)     ;DRIVE SWITCHED PORTS ?
7801 033642 001375          BNE      7$             ;BR IF NOT
7802 033644 005301 8$:     DEC      R1             ;GO TO THE NEXT DRIVE
7803 033646 100366          BPL      6$             ;CHECK NEXT DRIVE
7804 033650 012637 177776     MOV      (SP)+,@#PS     ;RESTORE THE PROCESSOR STATUS
7805 033654 104413          RESREG          ;RESTORE R0 - R5
7806 033656 000207          RTS      PC             ;BYE-BYE
7807
7808 ;DRIVE INITIALIZATION ROUTINE
7809 ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
7810 ;AN RP04/5/6. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
7811 ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
7812 ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
7813 ;DRVSTA IS SET TO THE PROPER CONDITION.
7814 ;CALL
7815 ;
7816 ;     MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
7817 ;     MOV      RPADR,R4       ;UNIBUS ADDRESS OF RH11/RP04/5/6 (RPCS1)
7818 ;     JSR      RO,DRVINT      ;CALLED BY A JSR
7819 ;     RETURN1    ;ERROR OCCURRED (PARITY)
7820 ;     RETURN2    ;NORMAL RETURN
7821 ;
7822 033660 010546 822:     MOV      R5,-(SP)      ;SAVE R5
7823 033662 105061 033302     CLR      DRVSTA(R1)    ;START DRIVE STATUS AS OFFLINE
7824 033666 105061 033312     CLR      DRVSTYP(R1)   ;CLEAR THE DRIVE TYPE INDICATOR
7825 033672 105061 033350     CLR      ULDFLG(R1)    ;CLEAR THE UNLOAD FLAG
7826 033676 010164 000010     MOV      R1,RPCS2(R4)  ;SELECT A DRIVE
7827 033702 112764 000111 000000  MOV      #111,RPCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
```


7828	033710	032764	010000	000010	BIT	#BIT12,RPCS2(R4)	:NONEXISTENT DRIVE?	
7829	033716	001403			BEQ	1\$:NO---BRANCH	
7830	033720	004737	041234		JSR	PC,SET.IE	:GO SET "IE" WITHOUT A "TRE"	
7831	033724	000543			BR	6\$:LEAVE THIS ROUTINE	
7832	033726	105061	033302		1\$: CLRB	DRVSTA(R1)	:SET DRIVE STATUS TO OFFLINE	
7833	033732	032764	004000	000000	BIT	#BIT11,RPCS1(R4)	:SEE IF DRIVE AVAILABLE	
7834	033740	001537			BEQ	7\$:BR IF DRIVE NOT AVAILABLE	
7835	033742	004037	040554		JSR	RO,RD.RP	:READ THE DRIVE TYPE REG.	
7836	033746	000026			RPDT			
7837	033750	034260			8\$:ERROR RETURN ADDRESS	
7838	033752	012605			MOV	(SP)+,R5	:PUT DRIVE TYPE IN R5	
7839	033754	112761	000001	033312	MOV	#1,DRVTP(R1)	:SET RP04 INDICATOR	
7840	033762	022705	020020		CMP	#20020,R5	:IS IT A SINGLE PORT RP04?	
7841	033766	001431			BEQ	2\$:BRANCH IF YES	
7842	033770	022705	024020		CMP	#24020,R5	:IS IT A DUAL PORT RP04?	
7843	033774	001426			BEQ	2\$:BR IF YES	
7844	033776	112761	000002	033312	MOV	#2,DRVTP(R1)	:SET RP05 INDICATOR	
7845	034004	022705	020021		CMP	#20021,R5	:SINGLE PORT RP05 ?	
7846	034010	001420			BEQ	2\$:BR IF YES	
7847	034012	022705	024021		CMP	#24021,R5	:DUAL PORT RP05 ?	
7848	034016	001415			BEQ	2\$:BR IF YES	
7849	034020	112761	000004	033312	MOV	#4,DRVTP(R1)	:SET RP06 INDICATOR	
7850	034026	022705	020022		CMP	#20022,R5	:SINGLE PORT RP06 ?	
7851	034032	001407			BEQ	2\$:BR IF YES	
7852	034034	022705	024022		CMP	#24022,R5	:DUAL PORT RP06 ?	
7853	034040	001404			BEQ	2\$:BR IF YES	
7854	034042	112761	177777	033312	MOV	#-1,DRVTP(R1)	:SET INDICATOR TO 'OTHER'	
7855	034050	000471			BR	6\$:EXIT	
7856	034052	005737	034264		2\$: TST	TSTPGM	:INHIBIT PROGRAMMABLE DRIVE?	
7857	034056	001010			BNE	9\$:BRANCH IF NO	
7858	034060	032764	001000	000012	BIT	#BIT09,RPDS1(R4)	:IS DRIVE PROGRAMMABLE?	
7859	034066	001404			BEQ	9\$:BRANCH IF NO	
7860	034070	152761	000010	033312	BISB	#BIT03,DRVTP(R1)	:SET INDICATOR	
7861	034076	000456			BR	6\$:EXIT	
7862	034100	010346			9\$: MOV	R3,-(SP)	:SAVE R3	(REV D)
7863	034102	006301			ASL	R1	:CREATE WORD INDEX	(REV D)
7864	034104	016103	001740		MOV	BLKADR(R1),R3	:GET DPB START ADDRESS	(REV D)
7865	034110	010563	000262		MOV	R5,\$RPDT(R3)	:STORE DRIVE'S TYPE	(REV D)
7866	034114	006201			ASR	R1	:RESTORE DRIVE NUMBER	(REV D)
7867	034116	012603			MOV	(SP)+,R3	:RESTORE R3	(REV D)
7868	034120	012746	000121		MOV	#121,-(SP)	:DO A "READ-IN-PRESET"	
7869	034124	004037	040730		JSR	RO,WRT.RP		
7870	034130	000000			RPCS1			
7871	034132	034260			8\$			
7872	034134	012746	010000		MOV	#BIT12,-(SP)	:SET FMT22=1	
7873	034140	004037	040730		JSR	RO,WRT.RP		
7874	034144	000032			RPOF			
7875	034146	034260			8\$			
7876	034150	004037	040554		JSR	RO,RD.RP	:READ RPDS1	
7877	034154	000012			RPDS1			
7878	034156	034260			8\$			
7879	034160	012605			MOV	(SP)+,R5	:AND SAVE IT IN R5	
7880	034162	100015			BPL	4\$:BRANCH IF ATA=0	
7881	034164	116164	033416	000016	MOV	ATABIT(R1),RPAS(R4)	:CLEAR ATTENTION BIT	
7882	034172	004037	040554		JSR	RO,RD.RP	:FIND OUT WHY ATA=1	
7883	034176	000014			RPER1			


```
7884 034200 034260      8$
7885 034202 006126      ROL      (SP)+      ;IS IT UNSAFE?
7886 034204 100004      BPL      4$      ;BR IF NOT
7887 034206 112761 177777 033302  MOVB     #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
7888 034214 000407      BR      6$      ;EXIT
7889 034216 005105      4$:     COM      R5      ;CHECK MOL, DPR, DRY, AND VV
7890 034220 042705 167077  BIC      #^C<BIT12!BIT08!BIT07!BIT06>,R5
7891 034224 001003      BNE     6$      ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
7892 034226 112761 000001 033302  MOVB     #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
7893 034234 005720      6$:     TST      (R0)+   ;STEP OVER THE ERROR RETURN
7894 034236 000410      BR      8$      ;EXIT
7895 034240 006301      7$:     ASL      R1      ;CHANGE INDEX TO ADDRESS WORDS
7896 034242 012761 003720 033374  MOV      #2000.,TIMER(R1) ;START 2 SEC TIMER
7897 034250 006201      ASR      R1      ;RESTORE R1
7898 034252 105161 033322  COMB     DPINT(R1)     ;SET PORT INITIALIZE INIDICATOR
7899 034256 005720      TST      (R0)+
7900 034260 012605      8$:     MOV      (SP)+,R5   ;RESTORE R5
7901 034262 000200      RTS      R0      ;EXIT
7902
7903      ;TEST PROGRAMMABLE DRIVE FLAG (TSTPGM=1 WORD)
7904      ;THE FLAG WILL BE SET BY THE PROGRAM UNDER
7905      ;MANUFACTURING CONDITIONS (ACT,APT) AND
7906      ;CLEARED UNDER FIELD CONDITIONS(XXDP CHAIN,
7907      ;STANDALONE) WITH STARTING ADDRESS 200.
7908      ;THE FLAG WILL BE SET UNDER ALL CONDITIONS
7909      ;WITH STARTING ADDRESS 220.
7910
7911 034264 000000      TSTPGM: .WORD 0      ;=1 TEST PROGRAMMABLE DRIVE
7912
7913      ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
7914      ;
7915      ;CALL
7916      ;
7917      ; JSR      R0,@#RP04      ;CALL THE RP04/5/6 DRIVER
7918      ; PNTADR     ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
7919      ; RETURN1    ;RETURN HERE IF QUEUE IS FULL
7920      ; RETURN2    ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
7921      ; IS AN ERROR CONDITION
7922
7923 034266 013746 177776      RP04:   MOV      @#PS,-(SP)     ;SAVE THE CALLING STATUS
7924 034272 013737 033434 177776  MOV      RPVEC+2,@#PS   ;DON'T ALLOW ANY RP04/5/6 INTERRUPTS
7925 034300 112737 000001 033346  MOVB     #1,ACTDRV     ;SET "ACTIVE DRIVER" FLAG
7926 034306 104412      SAVREG     ;SAVE R0 - R5
7927 034310 011002      MOV      (R0),R2      ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
7928 034312 005062 000016  CLR      16(R2)      ;CLEAR THE STATUS/ERROR INDICATOR
7929 034316 111201      MOVB     (R2),R1      ;PICKUP THE DRIVE NUMBER
7930 034320 013704 033430  MOV      RPADR,R4      ;UNIBUS ADDRESS OF RPCS1
7931 034324 105761 033302  TSTB     DRVSTA(R1)    ;CHECK DRIVES STATUS
7932 034330 003014      BGT      1$      ;BRANCH IF ONLINE
7933 034332 105761 033350  TSTB     ULDFLG(R1)    ;UNLOAD COMMAND IN QUEUE?
7934 034336 001036      BNE     3$      ;BRANCH IF YES
7935 034340 105761 033322  TSTB     DPINT(R1)     ;TRYING TO INIT THE DRIVE
7936 034344 001042      BNE     5$      ;BR IF YES
7937 034346 004037 033660  JSR      R0,DRVINT     ;GO INIT. THE DRIVE
7938 034352 000434      BR      4$      ;ERROR RETURN
7939 034354 105761 033302  TSTB     DRVSTA(R1)    ;IS DRIVE STATUS ONLINE?
```



```
7940 034360 003445          BLE      6$          ;BR IF NOT
7941 034362 105761 033332 1$:      TSTB     DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
7942 034366 001031          BNE      5$          ;BR IF YES
7943 034370 010164 000010      MOV      R1,RPCS2(R4) ;SELECT THE DRIVE
7944 034374 004037 041676      JSR      R0,DRVQUE  ;PUT THIS REQUEST IN QUEUE
7945 034400 000460          BR       9$          ;QUEUE IS FULL
7946 034402 122762 000103 000002      CMPB     #103,2(R2) ;IS THIS REQ. FOR AN UNLOAD?
7947 034410 001003          BNE      2$          ;BR IF NO
7948 034412 112761 177777 033350      MOVB     #-1,ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG
7949 034420 105761 033272 2$:      TSTB     DRVACT(R1) ;IS THIS DRIVE ACTIVE?
7950 034424 001043          BNE      8$          ;BR IF YES
7951 034426 004737 034560      JSR      PC,OPT     ;CALL THE OPTIMIZER
7952 034432 000440          BR       8$
7953 034434 012762 120000 000016 3$:      MOV      #BIT15:BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
7954 034442 000434          BR       8$          ;EXIT
7955 034444 004737 035670 4$:      JSR      PC,C17     ;GO HANDLE THE PARITY ERROR
7956 034450 000431          BR       8$
7957 034452 004037 041676 5$:      JSR      R0,DRVQUE  ;PUT REQUEST IN QUEUE
7958 034456 000431          BR       9$          ;QUEUE IS FULL
7959 034460 032714 000100      BIT      #BIT06,(R4) ;IS 'IE' SET ALREADY ?
7960 034464 001023          BNE      8$          ;BR IF IT IS
7961 034466 004737 041234      JSR      PC,SET.IE  ;SET INTERRUPT
7962 034472 000420          BR       8$          ;RETURN, REQUEST IN QUEUE
7963 034474 105761 033302 6$:      TSTB     DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
7964 034500 002412          BLT      7$          ;BR IF UNSAFE
7965 034502 012762 140000 000016      MOV      #BIT15:BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
7966 034510 105761 033312      TSTB     DRVTYP(R1) ;SEE IF OFFLINE OR NONEXISTENT
7967 034514 001007          BNE      8$          ;BR IF OFFLINE
7968 034516 012762 100002 000016      MOV      #BIT15:BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
7969 034524 000403          BR       8$          ;GO TO EXIT
7970 034526 012762 110000 000016 7$:      MOV      #BIT15:BIT12,16(R2) ;DRIVE IS UNSAFE
7971 034534 104413          8$:      RESREG   ;RESTORE R0 - R5
7972 034536 005720          TST      (R0)+      ;SETUP FOR NORMAL RETURN
7973 034540 000401          BR       10$         ;FINISH UP, THEN EXIT
7974 034542 104413          9$:      RESREG   ;RESTORE R0 - R5
7975 034544 005720          10$:     TST      (R0)+      ;CORRECT THE RETURN ADDRESS
7976 034546 105037 033346      CLR      ACTDRV    ;CLEAR "ACTIVE DRIVER" FLAG
7977 034552 012637 177776      MOV      (SP)+,@#PS ;RETURN "PS" TO USER LEVEL
7978 034556 000200          RTS      R0        ;RETURN TO CALLER
```

```
7979
7980          ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
7981          ;CALL
7982          ;
7983          ;
7984          ;
7985          ;
7986 034560 104412          OPT:     SAVREG   ;SAVE R0 - R5
7987 034562 013746 177776      MOV      @#PS,-(SP) ;SAVE PROC. STATUS
7988 034566 146137 033416 033344      BICB     ATABIT(R1),SRCHWT ;CLEAR "SEARCH WAIT" KEY
7989 034574 004737 041752      JSR      PC,GETREQ  ;GET "DPB" POINTER OF REQUEST
7990 034600 005702          TST      R2        ;IS THERE A REQUEST IN QUEUE?
7991 034602 001505          BEQ      7$          ;NO--BRANCH TO EXIT
7992 034604 032764 004000 000000      BIT      #BIT11,RPCS1(R4) ;IS DVA SET?
7993 034612 001407          BEQ      10$         ;BRANCH IF NOT
7994 034614 032764 000100 000012      BIT      #BIT6,RPDS1(R4) ;IS VV SET ?
7995 034622 001003          BNE      10$         ;BR IF IT IS
```



```

7996 034624 004037 033660      9$:   JSR   R0,DRVINT      ;SEE IF DRIVE STILL ONLINE ?
7997 034630 000470                BR   6$                ;PARITY OR 'DVA' NOT SET
7998 034632 105761 033302      10$:  TSTB  DRVSTA(R1)      ;IS DRIVE ONLINE?
7999 034636 003014                BGT   1$                ;YES--BRANCH
8000 034640 004737 041774      JSR   PC,POPQUE        ;NO--REMOVE REQUEST FROM QUEUE
8001 034644 012762 140000 000016  MOV   #BIT15:BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
8002 034652 105761 033302      TSTB  DRVSTA(R1)      ;IS DRIVE UNSAFE ?
8003 034656 100064                BPL   8$                ;BR TO EXIT IF NOT
8004 034660 012762 110000 000016  MOV   #BIT15:BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
8005 034666 000460                BR   8$                ;BRANCH TO EXIT
8006 034670 012746 000111      1$:   MOV   #111,-(SP)     ;LOAD COMMAND ONTO THE STACK
8007 034674 004037 040730      JSR   R0,WRT.RP        ;LOAD THE REGISTER
8008 034700 000000                RPCS1                   ;REGISTER INCREMENT
8009 034702 035012                6$                      ;ERROR RETURN ADDRESS
8010 034704 032714 004000      BIT   #BIT11,(R4)     ;DRIVE AVAILABLE ?
8011 034710 001427                BEQ   5$                ;BR IF NOT
8012 034712 122762 000150 000002  CMPB  #150,2(R2)      ;IS THE REQUEST FOR I/O?
8013 034720 002403                BLT   2$                ;YES--BRANCH
8014 034722 004737 035254      JSR   PC,C14           ;CALL THE COMMAND INITIATOR
8015 034726 000440                BR   8$                ;BRANCH TO EXIT
8016 034730 005737 033414      2$:   TST   DTUW           ;DATA TRANSFER UNDERWAY?
8017 034734 002012                BGE   4$                ;YES--GO START A SEARCH
8018 034736 005737 033372      TST   SEEKFG          ;DO IMPLIED SEEKS?
8019 034742 100404                BMI   3$                ;YES---BRANCH
8020 034744 004037 036224      JSR   R0,LA           ;NO--DO LOOK AHEAD
8021 034750 000427                BR   8$                ;RETURN HERE ON A PARITY ERROR
8022 034752 000403                BR   4$                ;GO START A SEARCH
8023 034754 004737 035040      3$:   JSR   PC,C11         ;START A DATA TRANSFER
8024 034760 000423                BR   8$                ;BRANCH TO EXIT
8025 034762 004737 035146      4$:   JSR   PC,C13         ;START A SEARCH
8026 034766 000420                BR   8$                ;GO TO THE EXIT
8027 034770 112761 177777 033332  5$:   MOVB  #-1,DPRQS(R1)   ;SET PORT REQUEST INDICATOR
8028 034776 010103                MOV   R1,R3           ;SET UP TO ADDRESS WORDS
8029 035000 006303                ASL   R3              ;CONVERT TO WORD INDEX
8030 035002 012763 023420 033374  MOV   #10000.,TIMER(R3) ;START 10 SEC TIMER
8031 035010 000402                BR   7$                ;EXIT
8032 035012 004737 035670      6$:   JSR   PC,C17         ;PROCESS THE PARITY ERROR
8033 035016 032714 000100      7$:   BIT   #BIT06,(R4)   ;SEE IF 'IE' ALREADY SET
8034 035022 001002                BNE   8$                ;BR IF SET
8035 035024 004737 041234      JSR   PC,SET.IE       ;SET 'IE' WITHOUT A 'TRE'
8036 035030 012637 177776      8$:   MOV   (SP)+,@#PS     ;RESTORE PROC. STATUS
8037 035034 104413                RESREG                  ;RESTORE R0 - R5
8038 035036 000207                RTS   PC
8039
8040                ;COMMAND INITIATOR
8041                ;
8042                ;CALL
8043                ;   MOV#  #DRVNUM,R1      ;DRIVE NUMBER
8044                ;   MOV#  #DPB,R2       ;ADDRESS OF DPB
8045                ;   JSR   PC,C1?      ;C1?= C11,C13, OR C14
8046                ;   ;WHERE:
8047                ;   ;C11=DATA TRANSFER
8048                ;   ;C12=SEARCH REQUESTED BY DATA XFER
8049                ;   ;C14=NOT DATA TRANSFER
8050
8051 035040 004737 041774      C11:  JSR   PC,POPQUE    ;REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE

```


8052	035044	010237	033342		MOV	R2,TRNSWT	;PUT REQ. IN TRANSFER WAIT QUEUE
8053	035050	010203			MOV	R2,R3	;DPB ADDRESS TO R3
8054	035052	013704	033430		MOV	RPADR,R4	;RPCS1 ADDRESS
8055	035056	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE
8056	035062	062703	000004		ADD	#4,R3	;DESIRED WORD COUNT
8057	035066	062704	000002		ADD	#2,R4	;RPWC ADDRESS
8058	035072	012324			MOV	(R3)+,(R4)+	;LOAD WORD COUNT
8059	035074	012324			MOV	(R3)+,(R4)+	;LOAD BUFFER ADDRESS
8060	035076	012346			MOV	(R3)+,-(SP)	;LOAD SECTOR AND TRACK
8061	035100	004037	040730		JSR	R0,WRT.RP	;CALL THE LOAD(WRITE) ROUTINE
8062	035104	000006			RPDA		;INDEX OF REGISTER TO LOAD
8063	035106	035670			C17		;ERROR RETURN ADDRESS
8064	035110	012346			MOV	(R3)+,-(SP)	;LOAD CYLINDER ADDRESS
8065	035112	004037	040730		JSR	R0,WRT.RP	
8066	035116	000034			RPCA		
8067	035120	035670			C17		
8068	035122	016246	000002		MOV	2(R2),-(SP)	;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
8069	035126	004037	040730		JSR	R0,WRT.RP	
8070	035132	000000			RPCS1		
8071	035134	035670			C17		
8072	035136	010137	033414		MOV	R1,DTUW	;SET "DATA TRANSFER UNDERWAY"
8073	035142	000137	035632		JMP	C15	
8074	035146	013704	033430		MOV	RPADR,R4	;RPCS1 ADDRESS
8075	035152	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE
8076	035156	016246	000012		MOV	12(R2),-(SP)	;DESIRED CYLINDER ADDRESS
8077	035162	004037	040730		JSR	R0,WRT.RP	
8078	035166	000034			RPCA		
8079	035170	035670			C17		
8080	035172	116203	000010		MOVB	10(R2),R3	;PICKUP SECTOR ADDRESS
8081	035176	163703	033444		SUB	MXWINDW,R3	;BACKUP BY MAX. SEARCH FOR I/O WINDOW
8082	035202	002002			BGE	1\$	
8083	035204	062703	000026		ADD	#22.,R3	
8084	035210	010346			MOV	R3,-(SP)	;COMBINE THE ADJUSTED SECTOR WITH
8085	035212	116266	000011	000001	MOVB	11(R2),1(SP)	;THE DESIRED TRACK
8086	035220	004037	040730		JSR	R0,WRT.RP	;LOAD DESIRED TRACK & SECTOR
8087	035224	000006			RPDA		
8088	035226	035670			C17		
8089	035230	012746	000131		MOV	#131,-(SP)	;START A SEARCH
8090	035234	004037	040730		JSR	R0,WRT.RP	
8091	035240	000000			RPCS1		
8092	035242	035670			C17		
8093	035244	156137	033416	033344	BISB	ATABIT(R1),SRCHWT	;SET "SEARCH WAIT" KEY
8094	035252	000567			BR	C15	
8095	035254	013704	033430		MOV	RPADR,R4	;RPCS1 ADDRESS
8096	035260	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE
8097	035264	116203	000002		MOVB	2(R2),R3	;PICKUP THE REQUESTED COMMAND
8098	035270	122703	000131		CMPB	#131,R3	;IS IT A SEARCH COMMAND?
8099	035274	001007			BNE	1\$;BRANCH IF NO
8100	035276	016246	000010		MOV	10(R2),-(SP)	;LOAD DESIRED TRACK & SECTOR
8101	035302	004037	040730		JSR	R0,WRT.RP	
8102	035306	000006			RPDA		
8103	035310	035670			C17		
8104	035312	000403			BR	2\$;GO LOAD CYLINDER
8105	035314	122703	000105		CMPB	#105,R3	;IS IT A SEEK COMMAND
8106	035320	001007			BNE	3\$;BRANCH IF NO
8107	035322	016246	000012		MOV	12(R2),-(SP)	;LOAD DESIRED CYLINDER

8108	035326	004037	040730		JSR	RO,WRT.RP	
8109	035332	000034			RPCA		
8110	035334	035670			C17		
8111	035336	000546			BR	C16	
8112	035340	122703	000115	3\$:	CMPB	#115,R3	:IS IT AN "OFFSET" COMMAND?
8113	035344	001013			BNE	4\$:BR IF NO
8114	035346	004037	040554		JSR	RO,RD.RP	:MERGE THE OFFSET VALUE INTO RPOF
8115	035352	000032			RPOF		:BUT DON'T CHANGE THE UPPER
8116	035354	035670			C17		
8117	035356	116216	000001		MOVB	1(R2),(SP)	:BYTE WHEN LOADING THE
8118	035362	004037	040730		JSR	RO,WRT.RP	:REGISTER (RPOF)
8119	035366	000032			RPOF		
8120	035370	035670			C17		
8121	035372	000530			BR	C16	:GO START THE COMMAND
8122	035374	122703	000107	4\$:	CMPB	#107,R3	:IS IT A "RECALIBRATE" COMMAND?
8123	035400	001525			BEQ	C16	:BRANCH IF YES
8124	035402	122703	000117		CMPB	#117,R3	:IS IT A RETURN TO CENTER?
8125	035406	001522			BEQ	C16	:BRANCH IF YES
8126	035410	122703	000103		CMPB	#103,R3	:IS IT AN "UNLOAD" COMMAND?
8127	035414	001016			BNE	5\$:BRANCH IF NO
8128	035416	112761	000001	033272	MOVB	#1,DRVACT(R1)	:SET THE DRIVE ACTIVE INDICATOR
8129	035424	105061	033302		CLRB	DRVSTA(R1)	:PUT DRIVE STATUS TO OFFLINE
8130	035430	112761	000001	033350	MOVB	#1,ULDFLG(R1)	:SET "UNLOAD IN PROGRESS" FLAG
8131	035436	010346			MOV	R3,-(SP)	:START THE "UNLOAD" COMMAND
8132	035440	004037	040730		JSR	RO,WRT.RP	
8133	035444	000000			RPCS1		
8134	035446	035670			C17		
8135	035450	000207			RTS	PC	:RETURN TO USER
8136	035452	122703	000143	5\$:	CMPB	#143,R3	:IS IT A "SET FORMAT" COMMAND?
8137	035456	001014			BNE	6\$:BRANCH IF NO
8138	035460	004037	040554		JSR	RO,RD.RP	:READ THE OFFSET REGISTER
8139	035464	000032			RPOF		
8140	035466	035670			C17		
8141	035470	116266	000001	000001	MOVB	1(R2),1(SP)	:COMBINE "FMT22", "ECI", AND "HCI"
8142	035476	004037	040730		JSR	RO,WRT.RP	:LOAD "FMT22", "ECI", AND/OR "HCI".
8143	035502	000032			RPOF		
8144	035504	035670			C17		
8145	035506	000436			BR	12\$	
8146	035510	122703	000141	6\$:	CMPB	#141,R3	:IS IT A "GET REGISTER" COMMAND?
8147	035514	001023			BNE	10\$:BRANCH IF NO
8148	035516	016203	000006	7\$:	MOV	6(R2),R3	:POINTS TO 1ST ADDRESS OF WHERE
8149							:TO PUT THE REGISTER(S)
8150	035522	116237	000010	035540	MOVB	10(R2),9\$:INIT. THE INDEX FOR THE FIRST REG.
8151	035530	116205	000011		MCVB	11(R2),R5	:INDEX OF LAST REG. TO MOVE
8152	035534	004037	040554	8\$:	JSR	RO,RD.RP	:READ RP04/5/6 REGISTER
8153	035540	000000		9\$:	RPCS1		:INDEX OF REG. TO READ
8154	035542	035670			C17		
8155	035544	012623			MOV	(SP)+,(R3)+	:GET THE CONTENTS OF RH11/RP04/5/6 REG.
8156	035546	023705	035540		CMP	9\$,R5	:LAST REG. BEEN READ?
8157	035552	001414			BEQ	12\$:GET OUT IF YES
8158	035554	062737	000002	035540	ADD	#2,9\$:INCREASE THE INDEX BY 2
8159	035562	000764			BR	8\$:LOOP--MORE TO READ
8160	035564	122703	000145	10\$:	CMPB	#145,R3	:IS IT A "SELECT DRIVE" COMMAND?
8161	035570	001405			BEQ	12\$:BRANCH IF YES
8162	035572	010346		11\$:	MOV	R3,-(SP)	:LOAD THE COMMAND
8163	035574	004037	040730		JSR	RO,WRT.RP	

8164	035600	000000				RPCS1	
8165	035602	035670				C17	
8166	035604	004737	041774		12\$:	JSR	PC,POPQUE ;REMOVE REQ. FROM QUEUE
8167	035610	052762	000200	000016		BIS	#BIT07,16(R2) ;SET THE "DONE" BIT
8168	035616	005737	033370			TST	SAVEFG ;SAVE THE RH11/RP04/5/6 REGISTERS?
8169	035622	100002				BPL	13\$;BRANCH IF NO
8170	035624	004737	041116			JSR	PC,SVRH11 ;YES--GO SAVE THE REGISTERS
8171	035630	000207			13\$:	RTS	PC ;RETURN TO USER
8172	035632	006301			C15:	ASL	R1
8173	035634	012761	001750	033374		MOV	#1000.,TIMER(R1) ;SET A ONE SECOND TIMER
8174	035642	006201				ASR	R1
8175	035644	112761	000001	033272		MOV	#1,DRVACT(R1) ;SET THE DRIVE ACTIVE
8176	035652	000207				RTS	PC ;RETURN TO THE USER
8177	035654	010346			C16:	MOV	R3,-(SP) ;LOAD THE COMMAND
8178	035656	004037	040730			JSR	RO,WRT.RP
8179	035662	000000				RPCS1	
8180	035664	035670				C17	
8181	035666	000761				BR	C15
8182	035670	032764	010000	000010	C17:	BIT	#BIT12,RPCS2(R4) ;DRIVE NON-EXISTENT ?
8183	035676	001034				BNE	C18 ;BR IF YES
8184	035700	005702			1\$:	TST	R2 ;ANYTHING IN QUEUE ?
8185	035702	001405				BEQ	C17B ;BR IF NOT
8186	035704	012762	104000	000016		MOV	#BIT15:BIT11,16(R2) ;SET "PARITY" ERROR INDICATOR
8187	035712	004737	041116			JSR	PC,SVRH11 ;GO SAVE THE RH11/RP04/5/6 REGISTERS
8188	035716	012746	000111		C17B:	MOV	#111,-(SP) ;DO A "DRIVE CLEAR"
8189	035722	004037	040730			JSR	RO,WRT.RP
8190	035726	000000				RPCS1	
8191	035730	035770				C18	
8192	035732	004737	041656			JSR	PC,EMPTYQ ;EMPTY THE QUEUE
8193	035736	105061	033350			CLRB	ULDFLG(R1) ;CLEAR THE UNLOAD IN QUEUE FLAG
8194	035742	105061	033272			CLRB	DRVACT(R1) ;DRIVE IS IDLE
8195	035746	020137	033414			CMP	R1,DTUW ;IF THIS DRIVE HAD AN I/O REQUEST
8196	035752	001005				BNE	1\$;IN PROGRESS CLEAR ALL OF THE FLAGS
8197	035754	005037	033342			CLR	TRNSWT
8198	035760	012737	177777	033414		MOV	#-1,DTUW
8199	035766	000207			1\$:	RTS	PC
8200	035770	104412			C18:	SAVREG	;SAVE RO - R5
8201	035772	032764	010000	000010		BIT	#BIT12,RPCS2(R4) ;IS "NED" SET ?
8202	036000	001002				BNE	1\$;BR IF YES
8203	036002	005001				CLR	R1
8204	036004	005003				CLR	R3
8205	036006	105761	033272		1\$:	TSTB	DRVACT(R1) ;DRIVE ACTIVE?
8206	036012	001443				BEQ	5\$;BRANCH IF NO
8207	036014	013702	033342			MOV	TRNSWT,R2 ;GET THE "TRANSFER WAIT" QUEUE
8208	036020	020137	033414			CMP	R1,DTUW ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
8209	036024	001402				BEQ	2\$;BRANCH IF YES
8210	036026	004737	041752			JSR	PC,GETREQ ;GET THE DPB POINTER
8211	036032	005702			2\$:	TST	R2 ;QUEUE ENTRY FOR DRIVE ?
8212	036034	001415				BEQ	4\$;BR IF NOT
8213	036036	032764	010000	000010		BIT	#BIT12,RPCS2(R4) ;"NED" SET ?
8214	036044	001404				BEQ	3\$;BR IF NOT
8215	036046	012762	100002	000016		MOV	#BIT15:BIT01,16(R2) ;SET "DRIVE NON-EXISTENT" INDICATOR
8216	036054	000405				BR	4\$;CONTINUE
8217	036056	012762	102000	000016	3\$:	MOV	#BIT15:BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
8218	036064	004737	041116			JSR	PC,SVRH11 ;SAVE RH11/RP04/5/6 REGISTERS
8219	036070	012763	177777	033374	4\$:	MOV	#-1,TIMER(R3) ;STOP THE TIMER


```
8220 036076 105061 033272          CLRB  DRVACT(R1)      ;SET "DRIVE ACTIVE" TO IDLE
8221 036102 020137 033414          CMP   R1,DTUW        ;IS THIS DRIVE SETUP FOR A TRANSFER
8222 036106 001005                   BNE   5$              ;BR IF NOT
8223 036110 012737 177777 033414    MOV   #-1,DTUW       ;RESET THE INDICATOR
8224 036116 005037 033342          CLR   TRNSWT         ;CLEAR THE TRANSFER QUEUE
8225 036122 105061 033350          5$:  CLRB  ULDFLG(R1)   ;CLEAR UNLOAD FLAG
8226 036126 032764 010000 000010    BIT   #BIT12,RPCS2(R4) ;'NED' SET ?
8227 036134 001021                   BNE   6$              ;BR IF YES
8228 036136 005201                   INC   R1              ;MOVE TO THE NEXT DRIVE
8229 036140 062703 000002          ADD   #2,R3
8230 036144 042701 177770          BIC   #^C7,R1
8231 036150 001316                   BNE   1$              ;BRANCH IF MORE DRIVES
8232 036152 012737 177777 033414    MOV   #-1,DTUW       ;NO DATA TRANSFERS UNDERWAY
8233 036160 005037 033342          CLR   TRNSWT         ;CLEAR THE 'TRANSFER WAIT' QUEUE
8234 036164 004737 041600          JSR   PC,CLRQUE      ;CLEAR ALL OF THE REQUEST QUEUES
8235 036170 012764 000040 000010    MOV   #BIT05,RPCS2(R4) ;DO A MASSBUS INIT.
8236 036176 000406                   BR    7$              ;CONTINUE
8237 036200 004737 041656          6$:  JSR   PC,EMPTYQ   ;CLEAR THE DRIVE'S QUEUE
8238 036204 105061 033302          CLRB  DRVSTA(R1)     ;SET DRIVE TO OFFLINE
8239 036210 105061 033312          CLRB  DRVTYP(R1)    ;CLEAR THE DRIVE TYPE INDICATOR
8240 036214 004737 041234          7$:  JSR   PC,SET.IE  ;SET "IE" WITHOUT "TRE"
8241 036220 104413                   RESREG
8242 036222 000207                   RTS   PC              ;RESTORE R0 - R5
8243                                     ;RETURN
8244                                     ;LOOK AHEAD ROUTINE
8245                                     ;
8246                                     ;CALL
8247                                     ;
8248                                     ;   MOV   #DRVNUM,R1      ;DRIVE NUMBER
8249                                     ;   MOV   #DPB,R2        ;POINT TO DPB
8250                                     ;   JSR   RO,LA         ;GO CHECK THE WINDOW
8251                                     ;   RETURN1          ;ERROR RETURN
8252                                     ;   RETURN2          ;START A SEARCH
8253                                     ;   RETURN3          ;START A DATA TRANSFER
8254 LA:  MOV   RPADR,R4      ;GET RPCS1'S ADDRESS
8255      MOV   R1,RPCS2(R4)  ;SELECT DRIVE
8256      JSR   RO,RD.RP     ;READ CURRENT CYLINDER
8257      RPCC 4$
8258      4$  ;ERROR RETURN ADDRESS
8259      CMP   (SP)+,12(R2)  ;IS CURRENT CYLINDER=DESIRED
8260      ;CYLINDER?
8261      BNE   3$           ;EXIT IF NO
8262      INCB  LACNT(R1)     ;INCREMENT THE LOOK AHEAD COUNT
8263      CMPB  LACNT(R1),MXLACT ;EXCEED MAX?
8264      BGT   2$           ;BRANCH IF YES
8265      MOVB  10(R2),R3    ;GET DESIRED SECTOR ADDRESS AND
8266      SWAB  R3           ;MULT. BY 64--ALIGN WITH
8267      ASR   R3           ;LOOK AHEAD REGISTER
8268      ASR   R3
8269      MOV   #340,@#PS    ;PRIORITY LEVEL "7"
8270      JSR   RO,RD.RP     ;READ LOOK AHEAD REGISTER
8271      RPLA 4$
8272      4$
8273      SUB   (SP)+,R3     ;CALCULATE THE DELTA
8274      BGE   1$
8275      ADD   #<22.*64.>,R3 ;MAKE THE DELTA POSITIVE
```



```
8276 036326 023703 033440      1$:    CMP      MXDLTA,R3      :CHECK THE DELTA TO SEE
8277 036332 002406              BLT      3$                  :IF IT IS WITHIN THE
8278 036334 023703 033442      CMP      MNDLTA,R3          :WINDOW---IF YES, ZERO
8279 036340 002003              BGE      3$                  :THE LOOK AHEAD COUNT
8280 036342 105061 033360      2$:    CLR      LACNT(R1)     :AND TAKE THE I/O EXIT
8281 036346 005720              TST      (RO)+
8282 036350 005720      3$:    TST      (RO)+      :ADJUST THE RETURN ADDRESS
8283 036352 000402              BR       5$                  :EXIT
8284 036354 004737 035670      4$:    JSR      PC,C17        :PROCESS THE ERROR
8285 036360 000200      5$:    RTS      RO          :RETURN
8286
8287 ;INTERRUPT SERVICE ROUTINE
8288
8289 036362 112737 000001 033346 1SR:    MOV      #1,ACTDRV      :SET "ACTIVE DRIVER" FLAG
8290 036370 104412              SAVREG      :SAVE R0 - R5
8291 036372 013704 033430      MOV      RPADR,R4          :ADDRESS OF RHSCS1
8292 036376 013701 033414      MOV      DTUW,R1          :GET "DATA TRANSFER UNDERWAY" INDICATOR
8293 036402 002403              BLT      1$                  :BRANCH IF NO DATA TRANSFER UNDERWAY
8294 036404 004737 036426      JSR      PC,TD            :CALL TRANSFER DONE
8295 036410 000402              BR       2$                  :EXIT
8296 036412 004737 036710      1$:    JSR      PC,SC        :CALL SPECIAL CONDITIONS
8297 036416 104413      2$:    RESREG      :RESTORE R0 - R5
8298 036420 105037 033346      CLR      ACTDRV          :CLEAR "ACTIVE DRIVER" FLAG
8299 036424 000002              RTI                       :RETURN
8300
8301 ;TRANSFER DONE ROUTINE
8302
8303 036426 105061 033272      TD:    CLR      DRVACT(R1)   :SET DRIVE ACTIVE INDICATOR TO IDLE
8304 036432 012737 177777 033414      MOV      #-1,DTUW        :NO DATA TRANSFERS UNDERWAY
8305 036440 006301              ASL      R1
8306 036442 012761 177777 033374      MOV      #-1,TIMER(R1)   :CANCEL TIMEOUT
8307 036450 006201              ASR      R1
8308 036452 013702 033342      MOV      TRNSWT,R2       :GET "DPB" ADDRESS FROM THE
8309 036456 005037 033342      CLR      TRNSWT          :TRANSFER WAIT QUEUE--CLEAR QUEUE
8310 036462 052762 000200 000016      BIS      #BIT07,16(R2)   :SET DONE
8311 036470 010164 000010      MOV      R1,RPCS2(R4)    :SELECT THE DRIVE
8312 036474 004037 040554      JSR      RO,RD.RP        :TRANSFER ERROR(TRE=1)?
8313 036500 000000              RPCS1
8314 036502 035670              C17
8315 036504 006126              ROL      (SP)+
8316 036506 100421              BMI      3$                  :BR IF YES
8317 036510 005737 033370      TST      SAVEFG          :SAVE THE RH11/RP04/5/6 REGISTERS?
8318 036514 100002              BPL      1$                  :BRANCH IF NO
8319 036516 004737 041116      JSR      PC,SVRH11       :YES--SAVE THE REGISTERS
8320 036522 004737 036602      1$:    JSR      PC,WC          :SEE IF WRITE CHECK TO BE PUT IN QUEUE
8321 036526 004737 041752      JSR      PC,GETREQ       :GET DPB POINTER
8322 036532 005702              TST      R2                :ENTRY FOR DRIVE ?
8323 036534 001403              BEQ      2$                  :BR IF NOT
8324 036536 004737 034560      JSR      PC,OPT          :CALL OPTIMIZER
8325 036542 000462              BR       SC                  :CHECK OTHER DRIVES
8326 036544 012714 000113      2$:    MOV      #113,(R4)     :RELEASE THE DRIVE
8327 036550 000457              BR       SC                  :CHECK FOR OTHER DRIVES
8328 036552 052762 100100 000C16 3$:    BIS      #BIT15!BIT06,16(R2) :SET DATA ERROR FLAG
8329 036560 004737 041656      JSR      PC,EMPTYQ       :EMPTY THE "DRIVE'S WAIT" QUEUE
8330 036564 004737 041116      JSR      PC,SVRH11       :SAVE THE RH11/RP04/5/6 REGISTERS
8331 036570 012714 040111      MOV      #40111,(R4)     :ISSUE A "DRIVE CLEAR"
```



```

8332 036574 012714 000113          MOV    #113,(R4)      ;ISSUE A RELEASE TO THE DRIVE
8333 036600 000443          BR     SC             ;CHECK FOR OTHER DRIVES
8334
8335          ;FORCED WRITE CHECK ROUTINE
8336
8337 036602 005737 001424          WC:   TST    AUTOCK      ;AUTOMATIC WRITE CHECKS ?
8338 036606 001437          BEQ    2$            ;BR IF NOT
8339 036610 122762 000002 000024          CMPB  #2,$CODE(R2)   ;LAST OPERATION WRITE DATA ?
8340 036616 001404          BEQ    1$            ;BR IF IT WAS
8341 036620 122762 000003 000024          CMPB  #3,$CODE(R2)   ;LAST OPERATION WRITE HEADER & DATA ?
8342 036626 001027          BNE    2$            ;BR IF NOT
8343 036630 004037 041676          1$:   JSR    R0,DRVQUE ;PUT THE OPERATION IN THE QUEUE
8344 036634 000424          BR     2$            ;QUEUE IS FULL
8345 036636 005062 000016          CLR    16(R2)        ;CLEAR 'DONE' BIT IN DPB
8346 036642 116262 000234 000027          MOVB  $RPCS1(R2),$PREV0(R2) ;SAVE WRITE OPERATION CODE
8347 036650 016262 000012 000034          MOV   $CYL(R2),$PREVA+2(R2) ;SAVE CYLINDER
8348 036656 016262 000010 000032          MOV   $SEC(R2),$PREVA(R2)  ;SAVE SECTOR AND TRACK ADDRESSES
8349 036664 142762 000002 000024          BICB  #2,$CODE(R2)   ;CHANGE WRITE TO CHECK
8350 036672 142762 000020 000002          BICB  #20,$COMND(R2)  ;CHANGE DRIVER CODE TO WRITE CHECK
8351 036700 152762 000010 000002          BISB  #10,$COMND(R2)  ;FINISH CHANGING CODE TO WRITE CHECK
8352 036706 000207          2$:   RTS    PC       ;EXIT
8353
8354          ;SPECIAL CONDITION ROUTINE
8355
8356 036710 116403 000016          SC:   MOVB  RPAS(R4),R3 ;READ 'RPAS'
8357 036714 001012          BNE    2$            ;BRANCH IF ANY 'ATA' BITS SET
8358 036716 004037 040554          JSR    R0,RD.RP     ;READ CONTROL AND STATUS REGISTER
8359 036722 000000          RPCS1
8360 036724 035770          C18
8361 036726 106126          ROLB  (SP)+         ;IS 'IE'=1?
8362 036730 100403          BMI   1$            ;YES, NO DRIVES TO CHECK
8363 036732 104001          ERROR 1             ;REPORT AN ILLEGAL INTERRUPT
8364 036734 004737 041234          JSR    PC,SET.IE   ;SET INTERRUPT ENABLE
8365 036740 000207          1$:   RTS    PC       ;RETURN
8366 036742 005046          2$:   CLR  -(SP)      ;PROCESS ALL DRIVES THAT HAVE
8367 036744 110316          MOVB  R3,(SP)      ;AN 'ATA'=1
8368 036746 012703 000001          MOV   #1,R3
8369 036752 005001          CLR  R1
8370 036754 030316          SC3:  BIT   R3,(SP)  ;ATA=1?
8371 036756 001005          BNE   SC5           ;YES--BRANCH
8372 036760 005201          SC4:  INC  R1        ;MOVE TO THE NEXT DRIVE
8373 036762 106303          ASLB  R3
8374 036764 001373          BNE   SC3           ;BRANCH IF MORE TO CHECK?
8375 036766 005726          TST  (SP)+         ;CLEAN OFF THE STACK
8376 036770 000207          RTS  PC            ;RETURN TO USER
8377 036772 105761 033322          SC5:  TSTB  DPINT(R1) ;INITIALIZING THE DRIVE ?
8378 036776 001402          BEQ   1$            ;BR IF NOT
8379 037000 000137 037666          JMP   SC13          ;PROCESS THE DRIVE
8380 037004 105761 033332          1$:   TSTB  DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
8381 037010 001402          BEQ   2$            ;BR IF NOT
8382 037012 000137 037666          JMP   SC13          ;START THE OUTSTANDING COMMAND
8383 037016 105761 033302          2$:   TSTB  DRVSTA(R1) ;CHECK THE DRIVE STATUS
8384 037022 003025          BGT   5$            ;BRANCH IF ONLINE
8385 037024 105761 033350          TSTB  ULDFLG(R1)   ;UNLOAD IN PROGRESS?
8386 037030 003422          BLE   5$            ;BRANCH IF NOT
8387 037032 004737 041752          JSR   PC,GETREQ    ;GET DPB POINTER

```



```

8388 037036 004737 041116      JSR    PC,SVRH11      ;SAVE THE RH11/RP04/5/6 REGISTERS
8389 037042 004737 037616      JSR    PC,SC12       ;SAVE RPDS1, RPER1, RPER2, AND RPER3
8390                               ;ALSO DO A DRIVE INIT (DRVINT)
8391 037046 105761 033302      TSTB   DRVSTA(R1)    ;DID DRIVE COME ONLINE?
8392 037052 003416             BLE    6$            ;NO---BRANCH
8393 037054 032737 040000 033262 BIT    #BIT14,RPERRS ;WAS THERE AN ERROR?
8394 037062 001002             BNE    3$            ;BR IF ERROR
8395 037064 000137 037526             JMP    SC11          ;NO ERROR
8396 037070 013705 033264      3$:   MOV    RPERRS+2,R5 ;YES -- PICKUP RPER1 AND
8397 037074 000476             BR     SC6A          ;GO PROCESS THE ERROR
8398 037076 105761 033272      5$:   TSTB   DRVACT(R1) ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
8399 037102 001027             BNE    SC6           ;BR IF EITHER
8400 037104 004737 037616      JSR    PC,SC12       ;SAVE RPDS1, RPER1, RPER2, AND RPER3
8401                               ;ALSO DO A DRVINT
8402 037110 105761 033322      6$:   TSTB   DPINT(R1)  ;TRYING TO INIT THE DRIVE ?
8403 037114 001321             BNE    SC4           ;BR IF YES, CHECK ON MORE DRIVES
8404 037116 105761 033302      TSTB   DRVSTA(R1)    ;CHECK ON DRIVE'S STATUS
8405 037122 100412             BMI    7$            ;BR IF UNSAFE
8406 037124 032737 020000 033270 BIT    #BIT13,RPERRS+6 ;ADDRESS PLUG CHANGED ?
8407 037132 001011             BNE    8$            ;BR IF YES
8408 037134 012746 000113      MOV    #113,-(SP)    ;RELEASE COMMAND
8409 037140 004037 040730      JSR    RO,WRT.RP     ;WRITE THE COMMAND INTO RPCS1
8410 037144 000000             RPCS1 ;REGISTER INDEX
8411 037146 037476             SC8   ;PARITY EXIT ADDRESS
8412 037150 011605      7$:   MOV    (SP),R5    ;PICKUP (RPAS) BEFORE THE ERROR CALL
8413 037152 104002             ERROR 2             ;REPORT THE UNEXPECTED ATTENTION
8414 037154 000701             BR     SC4           ;GO CHECK FOR MORE ATA'S
8415 037156             8$:
8416 037156 104005      ERROR 5             ;REPORT THE ADDRESS PLUG CHANGE
8417 037160 000677      BR     SC4           ;CHECK FOR MORE DRIVES
8418 037162 006301      5C6:  ASL    R1           ;SETUP TO ADDRESS WORDS
8419 037164 012761 177777 033374 MOV    #-1,TIMER(R1) ;STOP THE TIMER
8420 037172 006201      ASR    R1           ;RESTORE THE DRIVE ADDRESS
8421 037174 004737 041752      JSR    PC,GETREQ     ;GET THE DPB POINTER FROM THE QUEUE
8422 037200 010164 000010      MOV    R1,RPCS2(R4) ;SELECT DRIVE
8423 037204 004037 040554      JSR    RO,RD.RP     ;READ THE RP04'S STATUS REG.
8424 037210 000012      RPDS1
8425 037212 037476             SC8
8426 037214 011605      MOV    (SP),R5      ;AND PUT IT IN R5
8427 037216 006126      ROL    (SP)+        ;WAS THERE AN ERROR?
8428 037220 100407      BMI    1$           ;BR IF ERROR
8429 037222 105761 033272      TSTB   DRVACT(R1)    ;CHECK DRIVE'S STATE
8430 037226 003137             BGT    SC11          ;BR IF DRIVE ACTIVE WITH ORDER
8431 037230 052762 100210 000016 BIS    #BIT15:BIT07:BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
8432 037236 000470             BR     SC7           ;
8433 037240 004037 040554      1$:   JSR    RO,RD.RP     ;READ ERROR REGISTER #1
8434 037244 000014      RPER1
8435 037246 037476             SC8
8436 037250 012605      MOV    (SP)+,R5     ;AND SAVE IT IN R5
8437 037252 004737 041116      JSR    PC,SVRH11     ;SAVE RH11/RP04/5/6 REGISTERS
8438 037256 012746 000111      MOV    #111,-(SP)   ;ISSUE A DRIVE CLEAR
8439 037262 004037 040730      JSR    RO,WRT.RP
8440 037266 000000      RPCS1
8441 037270 037476             SC8
8442 037272 006105      5C6A: ROL    R5           ;WAS "UNSAFE" CONDITION =1?
8443 037274 100406             BMI    1$           ;BRANCH IF YES

```



```

8444 037276 005702          TST      R2          ;ANYTHING IN QUEUE ?
8445 037300 001447          BEQ      SC7         ;BR IF NOT
8446 037302 052762 100240 000016  BIS     #BIT15:BIT07:BIT05,16(R2) ;INFORM USER OF ERROR
8447 037310 000443          BR       SC7
8448 037312 004037 040554      1$:     JSR     RO,RD.RP    ;READ DRIVE STATUS REG. #1
8449 037316 000012          RPDS1
8450 037320 037476          SC8
8451 037322 011605          MOV     (SP),R5     ;SAVE RPDS1 IN R5
8452 037324 006126          ROL     (SP)+       ;"ERR"=1?
8453 037326 100011          BPL     2$         ;BR IF NO--UNSAFE CLEARED
8454 037330 112761 177777 033302  MOVB   #-1,DRVSTA(R1) ;DRIVE IS UNSAFE
8455 037336 004737 041116 000016  JSR     PC,SVRH11   ;SAVE RH11/RP04/5/6 REGISTERS
8456 037342 052762 110000          BIS     #BIT15:BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
8457 037350 000423          BR       SC7
8458 037352 032705 010000      2$:     BIT     #BIT12,R5     ;"MOL" = 1 ?
8459 037356 001015          BNE     3$         ;BR IF YES
8460 037360 112761 177777 033272  MOVB   #-1,DRVACT(R1) ;ACTIVE ERROR RECOVER
8461 037366 112761 000001 033302  MOVB   #1,DRVSTA(R1) ;ONLINE
8462 037374 006301          ASL     R1
8463 037376 012761 072460 033374  MOV     #30000.,TIMER(R1) ;START 30 SECOND TIMER
8464 037404 006201          ASR     R1
8465 037406 000137 036760          JMP     SC4
8466 037412 052762 100220 000016  3$:     BIS     #BIT15:BIT07:BIT04,16(R2) ;INFORM USER OF ERROR
8467 037420 105061 033272  SC7:    CLRB   DRVACT(R1)   ;DRIVE IS IDLE
8468 037424 004737 041656          JSR     PC,EMPTYQ   ;DUMP THE QUEUE
8469 037430 105761 033350          TSTB   ULDFLG(R1)  ;UNLOAD IN PROGRESS OR QUEUE?
8470 037434 003002          BGT     1$         ;BR IF NOT
8471 037436 105061 033350          CLRB   ULDFLG(R1)  ;CLEAR UNLOAD FLAG
8472 037442 116164 033416 000016  1$:     MOVB   ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
8473 037450 105761 033302          TSTB   DRVSTA(R1)  ;IS THE DRIVE UNSAFE ?
8474 037454 100406          BMI     2$         ;BR IF IT IS
8475 037456 012746 000113          MOV     #113,-(SP) ;RELEASE COMMAND
8476 037462 004037 040730          JSR     RO,WRT.RP   ;WRITE THE COMMAND INTO RPCS1
8477 037466 000000          RPCS1
8478 037470 037476          SC8
8479 037472 000137 036760      2$:     JMP     SC4         ;CHECK FOR MORE DRIVES
8480 037476 105761 033272  SC8:    TSTB   DRVACT(R1)  ;IS DRIVE IDLE?
8481 037502 001405          BEQ     1$         ;YES--BRANCH
8482 037504 004737 041752          JSR     PC,GETREQ   ;GET DPB POINTER
8483 037510 004737 035670          JSR     PC,C17      ;PROCESS THE PARITY ERROR
8484 037514 000402          BR      2$         ;CONTINUE
8485 037516 004737 035716      1$:     JSR     PC,C17B     ;PROCESS THE UNCORRECTABLE PARITY ERROR
8486 037522 000137 036760      2$:     JMP     SC4         ;CHECK MORE DRIVES
8487 037526 105761 033350  SC11:   TSTB   ULDFLG(R1)  ;"UNLOAD IN PROGRESS"?
8488 037532 003402          BLE     1$         ;BRANCH IF NO
8489 037534 105061 033350          CLRB   ULDFLG(R1)  ;CLEAR UNLOAD FLAG
8490 037540 105061 033272      1$:     CLRB   DRVACT(R1)  ;SET DRIVE IDLE
8491 037544 136137 033416 033344  BITB   ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
8492                                     ;AN I/O COMMAND?
8493 037552 001012          BNE     2$         ;BRANCH IF YES
8494 037554 004737 041774          JSR     PC,POPQUE   ;REMOVE REQUEST FROM QUEUE
8495 037560 052762 000200 000016  BIS     #BIT07,16(R2) ;SET "DONE" BIT
8496 037566 005737 033370          TST    SAVEFG      ;SAVE THE REGISTERS?
8497 037572 100002          BPL     2$         ;BRANCH IF NO
8498 037574 004737 041116          JSR     PC,SVRH11   ;YES--SAVE ALL OF THE RH11/RP04/5/6 REG'S
8499 037600 116164 033416 000016  2$:     MOVB   ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT

```



```

8500 037606 004737 034560 JSR PC,OPT ;START A REQUEST
8501 037612 000137 036760 JMP SC4 ;CHECK FOR MORE DRIVES
8502 037616 010164 000010 SC12: MOV R1,RPCS2(R4) ;SELECT DRIVE
8503 037622 016437 000012 033262 MOV RPDS1(R4),RPERRS ;SAVE THE FOUR REGISTERS THAT
8504 037630 016437 000014 033264 MOV RPER1(R4),RPERRS+2 ;WILL TELL US SOMETHING
8505
8506 037636 016437 000040 033266 MOV RPER2(R4),RPERRS+4
8507
8508 037644 016437 000042 033270 MOV RPER3(R4),RPERRS+6
8509 037652 004037 033660 JSR RO,DRVINT ;INIT. THE STATE OF THE DRIVE
8510 037656 000401 BR 1$ ;TAKE ERROR EXIT
8511 037660 000207 RTS PC ;RETURN
8512 037662 005726 1$: TST (SP)+ ;POP PC OFF OF THE STACK
8513 037664 000704 BR SC8 ;PROCESS THE PARITY ERROR
8514 037666 006301 SC13: ASL R1 ;SETUP TO ADDRESS WORDS
8515 037670 012761 177777 033374 MOV #-1,TIMER(R1) ;STOP THE TIMER
8516 037676 006201 ASR R1 ;
8517 037700 010164 000010 MOV R1,RPCS2(R4) ;SELECT THE DRIVE
8518 037704 116164 033416 000016 MOV B ATABIT(R1),RPAS(R4) ;CLEAR THE ATTENTION BIT
8519 037712 032714 004000 BIT #BIT11,(R4) ;DRIVE AVAILABLE ?
8520 037716 001006 BNE 1$ ;BR IF AVAILABLE
8521 037720 006301 ASL R1 ;
8522
8523 037722 012761 023420 033374 MOV #10000.,TIMER(R1) ;START 10 SEC TIMER AGAIN
8524
8525 037730 006201 ASR R1 ;
8526 037732 000433 BR 3$ ;EXIT
8527 037734 105761 033322 1$: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
8528 037740 001424 BEQ 2$ ;BR IF NOT
8529 037742 105061 033322 CLRB DPINT(R1) ;CLEAR THE INIT INDICATOR
8530 037746 004037 033660 JSR RO,DRVINT ;GO INIT THE DRIVE
8531 037752 000240 NOP ;DUMMY PARITY ERROR RETURN
8532 037754 105761 033302 TSTB DRVSTA(R1) ;DRIVE ONLINE ?
8533 037760 003014 BGT 2$ ;BR IF YES -- START ORDER
8534 037762 005702 TST R2 ;QUEUE ENTRY FOR THE DRIVE
8535 037764 001416 BEQ 3$ ;BR IF NOT
8536 037766 004737 041752 JSR PC,GETREQ ;GET DPB ADDRESS
8537 037772 052762 140000 000016 BIS #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
8538 040000 004737 041116 JSR PC,SVRH11 ;SAVE THE REGISTERS
8539 040004 004737 041656 JSR PC,EMPTYQ ;EMPTY THE REQUEST QUEUE
8540 040010 000404 BR 3$ ;
8541 040012 105061 033332 2$: CLRB DPRQS(R1) ;CLEAR THE PORT REQUEST INDICATOR
8542 040016 004737 034560 JSR PC,OPT ;START THE PENDING REQUEST
8543 040022 000137 036760 3$: JMP SC4 ;PROCESS OTHER DRIVES
8544
8545 ;RP04/5/6 TIMER ROUTINE
8546 ;CALL
8547 ; MOV #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
8548 ; JSR PC,RPTMR ;CALL RP04/5/6 TIME ROUTINE
8549
8550 040026 005737 033346 RPTMR: TST ACTDRV ;CHECK "ACTDRV & ACTSTR"
8551 040032 001030 BNE 4$ ;IF NON ZERO EXIT
8552 040034 112737 000001 033347 MOV B #1,ACTSTR ;SET "ACTSTR"
8553 040042 104412 SAVREG ;SAVE R0 - R5
8554 040044 005001 CLR R1 ;START WITH DRIVE 0
8555 040046 005003 CLR R3 ;

```



```
8556 040050 005763 033374 1$: TST TIMER(R3) ;IS THE TIMER RUNNING?
8557 040054 002407 BLT 2$ ;BRANCH IF NO
8558 040056 166663 000002 033374 SUB 2(SP),TIMER(R3) ;COUNT THE INTERVAL
8559 040064 003003 BGT 2$ ;BR IF NO SOFTWARE TIMEOUT
8560 040066 004737 040120 JSR PC,STO ;CALL SOFTWARE TIMEOUT ROUTINE
8561 040072 000405 BR 3$ ;GO TO THE EXIT
8562 040074 005201 2$: INC R1 ;MOVE TO NEXT DRIVE
8563 040076 005723 TST (R3)+
8564 040100 022701 000010 CMP #8.,R1 ;OUT OF DRIVES?
8565 040104 003361 BGT 1$ ;BRANCH IF NO
8566 040106 104413 3$: RESREG ;RESTORE R0 - R5
8567 040110 105037 033347 CLRB ACTSTR ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
8568 040114 012616 4$: MOV (SP)+,(SP) ;ADJUST THE STACK
8569 040116 000207 RTS PC ;RETURN
8570
8571 ;SOFTWARE TIMEOUT ROUTINE
8572 ;
8573 ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
8574 ; OR GREATER
8575 ;
8576 ;CALL: STO
8577 ; MOV #DRVNUM,R1 ;DRIVE NUMBER
8578 ; JSR PC,STO ;CALL
8579 ; RETURN
8580
8581 040120 010146 STO: MOV R1,-(SP) ;SAVE R1
8582 040122 010346 MOV R3,-(SP) ;SAVE R3
8583 040124 013704 033430 MOV RPADR,R4 ;GET ADDRESS OF "RPCS1"
8584 040130 010164 000010 MOV R1,RPCS2(R4) ;SELECT THE DRIVE
8585 040134 004037 040554 JSR R0,RD.RP ;READ "DRIVE STATUS REG"
8586 040140 000012 RPDS1
8587 040142 040442 ST05
8588 040144 105726 TSTB (SP)+ ;IS "DRY"=1?
8589 040146 100477 BMI ST02 ;BR IF YES
8590 040150 105761 033322 ST01: TSTB DPINT(R1) ;TRYING TO INTIALIZE THE DRIVE ?
8591 040154 001074 BNE ST02 ;BR IF YES
8592 040156 105761 033332 TSTB DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8593 040162 001071 BNE ST02 ;BR IF YES
8594 040164 013702 033342 MOV TRNSWT,R2 ;PICKUP TRANSFER WAIT QUEUE
8595 040170 020137 033414 CMP R1,DTUW ;TRANSFER UNDERWAY ON THIS DRIVE?
8596 040174 001402 BEQ 1$ ;BRANCH IF YES
8597 040176 004737 041752 JSR PC,GETREQ ;GET DPB ADDRESS
8598 040202 052762 101000 000016 1$: BIS #BIT15:BIT09,16(R2) ;SET THE ERROR FLAGS
8599 040210 004737 041116 JSR PC,SVRH11 ;SAVE RH11/RP04/5/6 REGISTERS
8600 040214 012764 000040 000010 MOV #BIT05,RPCS2(R4) ;"INIT" THE MASS BUS
8601 040222 105061 033272 CLRB DRVACT(R1) ;DRIVE IS IDLE
8602 040226 105061 033350 CLRB ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
8603 040232 005001 CLR R1 ;START WITH DRIVE 0
8604 040234 005003 CLR R3
8605 040236 004037 033660 2$: JSR R0,DRVINT ;INIT. THIS DRIVE
8606 040242 000477 BR ST05 ;PARITY ERROR RETURN
8607 040244 105761 033272 TSTB DRVACT(R1) ;DRIVE IDLE BEFORE THE INIT.?
8608 040250 001414 BEQ 4$ ;YES--BRANCH
8609 040252 013702 033342 MOV TRNSWT,R2 ;GET TRANSFER WAIT QUEUE
8610 040256 023701 033414 CMP DTUW,R1 ;WAS THERE I/O ON THIS DRIVE?
8611 040262 001402 BEQ 3$ ;YES--BRANCH
```



```
8612 040264 004737 041752 JSR PC,GETREQ ;GET THE DPB POINTER FROM QUEUE
8613 040270 052762 100400 000016 3$: BIS #BIT15:BIT08,16(R2) ;INFORM USER OF INIT.
8614 040276 105061 033272 CLR B DRVACT(R1) ;SET DRIVE ACTIVE TO IDLE
8615 040302 105061 033350 4$: CLR B ULDFLG(R1) ;NO UNLOAD
8616 040306 012763 177777 033374 MOV #-1,TIMER(R3) ;STOP THE TIMER
8617 040314 005723 TST (R3)+ ;UPDATE THE INDEX
8618 040316 005201 INC R1 ;INCREMENT THE DRIVE NUMBER
8619 040320 022701 000010 CMP #8.,R1 ;LAST DRIVE BEEN CHECKED?
8620 040324 003344 BGT 2$ ;NO--LOOP
8621 040326 012737 177777 033414 MOV #-1,DTUW ;NO DATA TRANSFERS UNDERWAY
8622 040334 005037 033342 CLR TRNSWT ;CLEAR TRANSFER WAIT QUEUE
8623 040340 004737 041600 JSR PC,CLRQUE ;CLEAR ALL REQUEST QUEUES
8624 040344 000500 BR ST09 ;EXIT
8625 040346 116405 000016 ST02: MOV B RPAS(R4),R5 ;READ ATTENTION REG
8626 040352 136105 033416 BIT B ATABIT(R1),R5 ;IS ATTENTION FOR THIS DRIVE UP ?
8627 040356 001017 BNE ST03 ;YES--BRANCH
8628 040360 105761 033322 TST B DPINT(R1) ;TRYING TO INITIALIZE THE DRIVE ?
8629 040364 001031 BNE ST06 ;BR IF YES - DRIVE NOT ONLINE
8630 040366 105761 033332 TST B DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8631 040372 001045 BNE ST07 ;BR IF YES - NO RESPONSE TO REQUEST
8632 040374 020137 033414 CMP R1,DTUW ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
8633 040400 001263 BNE ST01 ;BR IF NO
8634 040402 004037 040554 JSR R0,RD.RP ;YES--CHECK "RDY"
8635 040406 000000 RPCS1
8636 040410 040442 ST05
8637 040412 105726 TST B (SP)+
8638 040414 100255 BPL ST01 ;BR IF "RDY"=0
8639 040416 105761 033322 ST03: TST B DPINT(R1) ;INITIALIZING THE DRIVE ?
8640 040422 001003 BNE 1$ ;BR IF INIT PENDING
8641 040424 105761 033332 TST B DPRQS(R1) ;PORT REQUEST PENDING ?
8642 040430 001446 BEQ ST09 ;BR IF NOT
8643 040432 012763 177777 033374 1$: MOV #-1,TIMER(R3) ;STOP THE TIMER
8644 040440 000442 BR ST09 ;EXIT
8645 040442 004737 035770 ST05: JSR PC,C18 ;GO HANDLE THE PARITY ERROR
8646 040446 000437 BR ST09
8647 040450 105061 033322 ST06: CLR B DPINT(R1) ;CLEAR THE INITIALIZE INDICATOR
8648 040454 105061 033302 CLR B DRVSTA(R1) ;SET UNIT OFFLINE
8649 040460 012763 177777 033374 MOV #-1,TIMER(R3) ;STOP THE TIMER
8650 040466 004737 041752 JSR PC,GETREQ ;GET THE DPB ADDRESS
8651 040472 005702 TST R2 ;REQUEST IN QUEUE ?
8652 040474 001424 BEQ ST09 ;BR IF NOT
8653 040476 052762 140000 000016 BIS #BIT15:BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
8654 040504 000414 BR ST08 ;FINISH
8655 040506 012763 177777 033374 ST07: MOV #-1,TIMER(R3) ;STOP THE TIMER
8656 040514 105061 033332 CLR B DPRQS(R1) ;CLEAR PORT REQUEST INDICATOR
8657 040520 004737 041752 JSR PC,GETREQ ;GET DPB ADDRESS
8658 040524 005702 TST R2 ;QUEUE ENTRY FOR DRIVE ?
8659 040526 001407 BEQ ST09 ;BR IF NONE
8660 040530 012762 100004 000016 MOV #BIT15:BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
8661 040536 004737 041656 ST08: JSR PC,EMPTYQ ;CLEAR THE QUEUE FOR THE DRIVE
8662 040542 004737 041116 JSR PC,SVRH11 ;SAVE THE REGISTERS
8663 040546 012603 ST09: MOV (SP)+,R3 ;RESTORE R3
8664 040550 012601 MOV (SP)+,R1 ;RESTORE R1
8665 040552 000207 RTS PC ;RETURN
```

8666
8667 ;ROUTINE TO READ A RH11/RP04/5/6 REGISTER


```

8668
8669
8670
8671
8672
8673
8674
8675
8676 040554 013737 033426 040716 RD.RP: MOV MCPEMX, RD.RP2 ;MAX. RETRYS ALLOWED
8677 040562 011646 MOV (SP), -(SP) ;SAVE R0 FOR RETURN
8678 040564 013737 033430 040600 MOV RPADR, RD.ADR ;FORM THE DESIRED ADDRESS
8679 040572 062037 040600 ADD (R0)+, RD.ADR ;USING THE BASE AND THE INDEX
8680 040576 013727 RD.RP1: MOV @ (PC)+, (PC)+ ;READ THE DESIRED REGISTER OF THE RP04
8681 040600 000000 RD.ADR: .WORD 0 ;ADDRESS IS FORMED HERE
8682 040602 000000 RD.WRD: .WORD 0 ;REG. CONTENTS PUT HERE
8683 040604 013766 040602 000002 MOV RD.WRD, 2(SP) ;RETURN IT TO THE USER
8684 040612 013746 033430 MOV RPADR, -(SP) ;PUT THE ADDRESS ON THE STACK
8685 040616 062716 000010 ADD #RPCS2, (SP) ;FORM THE ADDRESS OF RPCS2
8686 040622 032736 010000 BIT #BIT12, @ (SP)+ ;CHECK THE 'NED' BIT
8687 040626 001035 BNE RD.RP3 ;BR IF DRIVE NON-EXISTENT
8688 040630 017746 172574 MOV @RPADR, -(SP) ;READ RPCS1
8689 040634 032716 020000 BIT #BIT13, (SP) ;DID MCPE SET?
8690 040640 001002 BNE 1$ ;BRANCH IF YES
8691 040642 022620 CMP (SP)+, (R0)+ ;ADJUST FOR RETURN
8692 040644 000430 BR RD.RP4 ;EXIT
8693 040646
8694 040646 104003 1$: ERROR 3 ;REPORT 'MCPE' ERROR
8695 040650 005737 033414 TST DTUW ;DATA TRANSFER UNDERWAY?
8696 040654 100405 BMI 2$ ;NO--BRANCH
8697 040656 032716 040000 BIT #BIT14, (SP) ;NO--'TRE'=1?
8698 040662 001402 BEQ 2$ ;NO--BRANCH
8699 040664 005726 TST (SP)+ ;YES--CLEAN OFF THE STACK AND
8700 040666 000415 BR RD.RP3 ;TAKE THE FATAL ERROR EXIT
8701 040670 052716 040000 2$: BIS #BIT14, (SP) ;CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
8702 040674 000316 SWAB (SP) ;POSITION BEFORE WRITING
8703 040676 013737 033430 040712 MOV RPADR, 3$ ;FORM ADDRESS OF HIGH BYTE
8704 040704 005237 040712 INC 3$
8705 040710 112637 MOVB (SP)+, @ (PC)+ ;WRITE THE HIGH BYTE OF RPCS1
8706 040712 000000 3$: .WORD 0 ;ADDRESS STORAGE
8707 040714 005327 DEC (PC)+ ;EXCEEDED MAX. RETRYS
8708 040716 000003 RD.RP2: .WORD 3
8709 040720 002326 BGE RD.RP1 ;BRANCH IF NO
8710 040722 011000 RD.RP3: MOV (R0), R0 ;FATAL ERROR EXIT
8711 040724 012616 MOV (SP)+, (SP)
8712 040726 000200 RD.RP4: RTS R0
8713
8714 ;ROUTINE TO WRITE A REGISTER
8715
8716
8717
8718
8719
8720
8721
8722
8723 040730 013737 033426 041100 WRT.RP: MOV MCPEMX, WRT.R2 ;MAX RETRYS ALLOWED

```



```
8724 040736 016637 000002 041016      MOV      2(SP),WRT.WD      ;SAVE THE WORD TO WRITE
8725 040744 012616                MOV      (SP)+,(SP)      ;ADJUST THE STACK
8726 040746 012037 041020                MOV      (R0)+,WRT.AD    ;GET INDEX OF REGISTER TO BE WRITTEN
8727 040752 001015                BNE      1$              ;BRANCH IF NOT RPCS1
8728 040754 122737 000150 041016      CMPB     #150,WRT.WD     ;IS THE COMMAND FOR DATA TRANSFERS?
8729 040762 002411                BLT      1$              ;YES--DON'T GET THE OLD A16 & A17, & PSEL
8730 040764 004037 040554                JSR      R0,RD.RP        ;NO---COMBINE A16&A17, & PSEL WITH
8731 040770 000000                RPCS1                    ;THE COMMAND BEFORE SENDING IT TO
8732 040772 041106                WRT.R3                    ;THE RH11/RP04
8733 040774 000316                SWAB     (SP)
8734 040776 042716 177770                BIC      #^C7,(SP)
8735 041002 112637 041017                MOV      (SP)+,WRT.WD+1
8736 041006 063737 033430 041020 1$:      ADD      RPADR,WRT.AD    ;FORM THE ADDRESS OF THE DISK REG.
8737 041014 012737                WRT.R1: MOV      (PC)+,@(PC)+ ;LOAD THE DESIRED REG.
8738 041016 000000                WRT.WD: .WORD 0          ;WORD TO WRITE GOES HERE
8739 041020 000000                WRT.AD: .WORD 0          ;ADDRESS IS FORMED HERE
8740 041022 013746 033430                MOV      RPADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
8741 041026 062716 000010                ADD      #RPCS2,(SP)    ;FORM THE ADDRESS OF RPCS2
8742 041032 032736 010000                BIT      #BIT12,@(SP)+ ;CHECK THE 'NED' BIT
8743 041036 001023                BNE      WRT.R3          ;BR IF DRIVE NON-EXISTENT
8744 041040 004037 040554                JSR      R0,RD.RP        ;CHECK FOR PARITY ERROR ON WRITE
8745 041044 000014                RPER1
8746 041046 041106                WRT.R3
8747 041050 032726 000010                BIT      #BIT03,(SP)+
8748 041054 001416                BEQ      WRT.R4          ;BRANCH IF 'PAR=0'
8749 041056 016037 177776 041070      MOV      -2(R0),1$      ;PICKUP THE INDEX
8750 041064 004037 040554                JSR      R0,RD.RP        ;READ THE REG.
8751 041070 000000                1$:      .WORD 0              ;REG. INDEX
8752 041072 041106                WRT.R3                    ;RETURN TO THIS ADDRESS ON ERROR
8753 041074 104004                ERROR 4                  ;REPORT THE PARITY ON WRITE ERROR
8754 041076 005327                DEC      (PC)+          ;DECREMENT THE ERROR COUNT
8755 041100 000003                WRT.R2: .WORD 3          ;RETRY COUNTER
8756 041102 002344                BGE      WRT.R1          ;TRY AGAIN IF NOT FINISHED
8757 041104 005726                TST      (SP)+          ;CLEAN OFF THE STACK
8758 041106 011000                WRT.R3: MOV      (R0),R0 ;TAKE THE 'PARITY ON WRITE' ERROR EXIT
8759 041110 000401                BR       WRT.R5          ;EXIT
8760 041112 005720                WRT.R4: TST      (R0)+   ;ADJUST FOR ERROR FREE EXIT
8761 041114 000200                WRT.R5: RTS      R0
8762
8763                ;ROUTINE TO SAVE THE RH11/RP04/5/6 REGISTERS AS PER DPB+14
8764                ;
8765                ;CALL
8766                ;
8767                ;      MOV      #DPBNUM,R2      ;DPB POINTER TO R2
8768                ;      JSR      PC,SVRH11    ;SAVE THE DRIVES REG'S
8769 041116 104412                SVRH11: SAVREG          ;SAVE R0 - R5
8770 041120 005702                TST      R2              ;QUEUE ENTRY FOR THE DRIVE ?
8771 041122 001430                BEQ      4$              ;BR IF NONE
8772 041124 013704 033430                MOV      RPADR,R4
8773 041130 111264 000010                MOV      (R2),RPCS2(R4) ;SELECT DRIVE
8774 041134 016203 000014                MOV      14(R2),R3      ;GET THE ERROR TABLE POINTER
8775 041140 001433                BEQ      6$              ;EXIT IF NO ADDRESS
8776 041142 005037 041176                CLR      3$              ;COUNTER & POINTER
8777 041146 023727 041176 000022 1$:      CMP      3$,#RPDB        ;REACHED THE BUFFER REGISTER ?
8778 041154 001006                BNE      2$              ;BR IF NOT
8779 041156 032764 000200 000010                BIT      #BIT07,RPCS2(R4) ;'OR' SET ?
```



```

8780 041164 001002          BNE      2$          ;BR IF SET
8781 041166 005023          CLR      (R3)+      ;STORE RPDB AS ZEROES
8782 041170 000405          BR       4$          ;CONTINUE
8783 041172 004037 040554  2$:      JSR      R0,RD.RP ;READ THE SELECTED REGISTER
8784 041176 000000          3$:      .WORD    0      ;REGISTER INDEX
8785 041200 041224          5$:      5$          ;ERROR RETURN ADDRESS
8786 041202 012623          MOV      (SP)+,(R3)+ ;STORE THE REGISTER CONTENTS
8787 041204 023727 041176 000046 4$:      CMP      3$,#RPEC2 ;REACHED THE END ?
8788 041212 001406          BEQ      6$          ;BR IF YES
8789 041214 062737 000002 041176      ADD      #2,3$      ;INCREMENT THE REGISTER INDEX
8790 041222 000751          BR       1$          ;CONTINUE READING THE REGISTERS
8791 041224 004737 035670 5$:      JSR      PC,C17    ;PROCESS THE UNCORRECTABLE PARITY ERROR
8792 041230 104413          6$:      RESREG          ;RESTORE R0 - R5
8793 041232 000207          RTS      PC          ;RETURN
8794
8795          ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
8796          ;CALL
8797          ;
8798          ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
8799          ;      JSR      PC,SET.IE      ;SET "IE"
8800          ;      RETURN
8801 041234 010446          SET.IE: MOV      R4,-(SP)      ;SAVE R4
8802 041236 013704 033430      MOV      RPADR,R4      ;PICKUP ADDRESS OF RPCS1
8803 041242 010164 000010      MOV      R1,RPCS2(R4) ;SELECT DRIVE
8804 041246 011446          MOV      (R4),-(SP)   ;READ RPCS1
8805 041250 052716 040000      BIS      #BIT14,(SP)  ;SET THE "TRE" BIT OF THE WORD READ
8806 041254 000316          SWAB     (SP)         ;ADJUST FOR DATO
8807 041256 112714 000100      MOV      #BIT06,(R4)  ;SET "IE"
8808 041262 032764 010000 000010      BIT      #BIT12,RPCS2(R4) ;IS "NED"=1?
8809 041270 001002          BNE      1$          ;YES--CLEAR "TRE"
8810 041272 005726          TST     (SP)+        ;CLEAN OFF THE STACK
8811 041274 000402          BR       2$          ;
8812 041276 112664 000001 1$:      MOV      (SP)+,1(R4)  ;CLEAR "TRE"
8813 041302 012604          2$:      MOV      (SP)+,R4   ;RESTORE R4
8814 041304 000207          RTS      PC          ;RETURN TO CALLER
8815
8816          ;QUEUE COUNT
8817 041306          000      QCNT: .BYTE    0      ;DRIVE 0
8818 041307          000      .BYTE    0      ;DRIVE 1
8819 041310          000      .BYTE    0      ;DRIVE 2
8820 041311          000      .BYTE    0      ;DRIVE 3
8821 041312          000      .BYTE    0      ;DRIVE 4
8822 041313          000      .BYTE    0      ;DRIVE 5
8823 041314          000      .BYTE    0      ;DRIVE 6
8824 041315          000      .BYTE    0      ;DRIVE 7
8825
8826          ;QUEUE INPUT POINTERS
8827
8828 041316 041400          QINPT: .WORD    QDRV0     ;DRIVE 0
8829 041320 041420          .WORD    QDRV1     ;DRIVE 1
8830 041322 041440          .WORD    QDRV2     ;DRIVE 2
8831 041324 041460          .WORD    QDRV3     ;DRIVE 3
8832 041326 041500          .WORD    QDRV4     ;DRIVE 4
8833 041330 041520          .WORD    QDRV5     ;DRIVE 5
8834 041332 041540          .WORD    QDRV6     ;DRIVE 6
8835 041334 041560          .WORD    QDRV7     ;DRIVE 7

```



```
8836
8837 ;QUEUE OUTPUT POINTERS
8838
8839 QOUTPT: .WORD QDRV0 ;DRIVE 0
8840 .WORD QDRV1 ;DRIVE 1
8841 .WORD QDRV2 ;DRIVE 2
8842 .WORD QDRV3 ;DRIVE 3
8843 .WORD QDRV4 ;DRIVE 4
8844 .WORD QDRV5 ;DRIVE 5
8845 .WORD QDRV6 ;DRIVE 6
8846 .WORD QDRV7 ;DRIVE 7
8847
8848 QSTART: .WORD QDRV0 ;DRIVE 0 START ADDRESS
8849 QSTOP: .WORD QDRV1 ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
8850 .WORD QDRV2 ;STOP DRIVE 1--START DRIVE 2
8851 .WORD QDRV3 ;STOP DRIVE 2--START DRIVE 3
8852 .WORD QDRV4 ;STOP DRIVE 3--START DRIVE 4
8853 .WORD QDRV5 ;STOP DRIVE 4--START DRIVE 5
8854 .WORD QDRV6 ;STOP DRIVE 5--START DRIVE 6
8855 .WORD QDRV7 ;STOP DRIVE 6--START DRIVE 7
8856 .WORD QTERM ;STOP DRIVE 7
8857
8858 ;DRIVE REQUEST QUEUES
8859
8860 QDRV0: .BLKW 10
8861 QDRV1: .BLKW 10
8862 QDRV2: .BLKW 10
8863 QDRV3: .BLKW 10
8864 QDRV4: .BLKW 10
8865 QDRV5: .BLKW 10
8866 QDRV6: .BLKW 10
8867 QDRV7: .BLKW 10
8868 QTERM=.
8869
8870 ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
8871 ;
8872 ;CALL
8873 ; JSR PC,CLRQUE
8874
8875 CLRQUE: SAVREG ;SAVE R0 - R5
8876 MOV #QCNT,R2 ;ZERO THE QUEUE COUNTS
8877 CLR (R2)+ ;DRIVES 0 & 1
8878 CLR (R2)+ ;DRIVES 2 & 3
8879 CLR (R2)+ ;DRIVES 4 & 5
8880 CLR (R2)+ ;DRIVES 6 & 7
8881 MOV #8,R3 ;MOVE THE STARTING
8882 MOV #QSTART,R1 ;ADDRESS OF THE QUEUE INTO
8883 1$: MOV (R1)+,(R2)+ ;THE QUEUE INPUT POINTER
8884 DEC R3
8885 BNE 1$
8886 MOV #8,R3 ;MOVE THE STARTING ADDRESS
8887 MOV #QSTART,R1 ;OF THE QUEUE INTO THE
8888 2$: MOV (R1)+,(R2)+ ;QUEUE OUTPUT POINTER
8889 DEC R3
8890 BNE 2$
8891 RESREG ;RESTORE R0 - R5
```



```
8892 041654 000207          RTS      PC
8893
8894          ;EMPTY THE QUEUE SPECIFIED BY R1
8895          ;
8896          ;CALL
8897          ;      MOV      DRVNUM,R1      ;DRIVE NUMBER TO R1
8898          ;      JSR      PC,EMPTYQ
8899
8900 041656 105061 041306    EMPTYQ: CLR      QCNT(R1)      ;CLEAR NUMBER OF ITEMS IN QUEUE
8901 041662 006301          ASL      R1
8902 041664 016161 041316 041336    MOV      QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
8903 041672 006201          ASR      R1
8904 041674 000207          RTS      PC
8905
8906          ;ROUTINE TO PUT A REQUEST IN QUEUE
8907          ;
8908          ;CALL
8909          ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER
8910          ;      MOV      #DPB,R2        ;ADDRESS OF PARAMETER BLOCK
8911          ;      JSR      RO,DRVQUE      ;GO PUT REQUEST IN QUEUE
8912          ;      RETURN1      ;RETURN HERE IF QUEUE IS FULL
8913          ;      RETURN2      ;RETURN HERE IF REQUEST IS IN QUEUE
8914
8915 041676 122761 000010 041306    DRVQUE: CMPB     #10,QCNT(R1)    ;IS QUEUE FULL?
8916 041704 001421          BEQ      2$                    ;BR IF YES-TAKE RETURN1
8917 041706 105261 041306          INCB     QCNT(R1)              ;INCREMENT QUEUE COUNT
8918 041712 006301          ASL      R1
8919 041714 010271 041316          MOV      R2,@QINPT(R1)        ;PUT THIS REQUEST IN QUEUE
8920 041720 062761 000002 041316          ADD     #2,QINPT(R1)          ;UPDATE THE QUEUE POINTER
8921 041726 026161 041316 041360          CMP     QINPT(R1),QSTOP(R1)  ;TIME TO RESET THE POINTER
8922 041734 001003          BNE     1$                    ;BRANCH IF NO
8923 041736 016161 041356 041316          MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
8924 041744 006201          1$:   ASR      R1
8925 041746 005720          TST     (R0)+                  ;TAKE RETURN 2
8926 041750 000200          2$:   RTS      R0              ;RETURN TO USER
8927
8928          ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
8929          ;
8930          ;CALL
8931          ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
8932          ;      JSR      PC,GETREQ      ;GO GET THE REQUEST
8933          ;      RETURN      ;R2="DPB" ADDRESS OF THE REQUEST
8934          ;      ;R2=0 IF NO REQUEST IN QUEUE
8935
8936 041752 005002          GETREQ: CLR      R2
8937 041754 105761 041306          TSTB     QCNT(R1)              ;IS THERE ANY REQUEST IN QUEUE?
8938 041760 001404          BEQ      2$                    ;NO---BRANCH
8939 041762 006301          1$:   ASL      R1
8940 041764 017102 041336          MOV     @QOUTPT(R1),R2        ;PICKUP "DPB" POINTER FOR THIS DRIVE
8941 041770 006201          ASR      R1
8942 041772 000207          2$:   RTS      PC              ;RETURN TO USER
8943
8944          ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
8945          ;
8946          ;CALL
8947          ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
```



```
8948      ;      JSR      PC,POPQUE      ;CALL TO REMOVE REQUEST
8949      ;      RETURN      ;R2=ADDRESS OF DPB REMOVED
8950
8951 041774 105361 041306      POPQUE: DECB      QCNT(R1)      ;DECREMENT QUEUE COUNT
8952 042000 006301              ASL      R1
8953 042002 017102 041336      MOV      @QOUTPT(R1),R2      ;GET THE "DPB" POINTER
8954 042006 062761 000002 041336      ADD      #2,QOUTPT(R1)      ;UPDATE THE QUEUE POINTER
8955 042014 026161 041336 041360      CMP      QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
8956 042022 001003              BNE      1$      ;NO--BRANCH TO EXIT
8957 042024 016161 041356 041336      MOV      QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
8958 042032 006201      1$:      ASR      R1
8959 042034 000207              RTS      PC      ;RETURN TO USER
```

;;*****

.SBTTL DATA, CONTROL, & STATUS BLOCKS

;;*****

;BLOCK LOCATION EQUATE STATEMENTS

```
8970      000001      $FMT      =      1      ;FMT,HCI,ECI OR OFFSET CODE
8971      000002      $COMND      =      $FMT+1      ;OPERATION CODE
8972      000003      $PSEL      =      $FMT+2      ;PORT SELECT & BITS A16, A17
8973      000004      $WRDM      =      $FMT+3      ;WORD COUNT (2'S COMP)
8974      000006      $BUF      =      $FMT+5      ;BUFFER ADDR OR REGISTER TABLE POINTER
8975      000010      $SEC      =      $FMT+7      ;SECTOR ADDRESS OR 1ST REG ADDR
8976      000011      $TRK      =      $FMT+10      ;TRACK ADDRESS OF LAST REG ADDR
8977      000012      $CYL      =      $FMT+11      ;CYLINDER ADDR
8978      000014      $REG      =      $FMT+13      ;REGISTER STORAGE (IF ERROR)
8979      000016      $STATUS      =      $FMT+15      ;STATUS WORD (SET BY DRIVER)
```

;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES

```
8983      000020      $WRDL      =      $FMT+17      ;WORD COUNT (NOT 2'S COMP)
8984      000022      $SSEC      =      $WRDL+2      ;SECTOR SIZE FOR CURRENT OPERATION
8985      000024      $CODE      =      $WRDL+4      ;PRESENT COMMAND SELECTION CODE
8986      000026      $PACK      =      $WRDL+6      ;WRITE DATA PACK INDICATOR
8987      000027      $PREVO      =      $WRDL+7      ;PREVIOUS COMMAND SELECTION CODE
8988      000030      $PATT      =      $WRDL+10      ;PATTERN CODE
8989      000032      $PREVA      =      $WRDL+12      ;PREVIOUS ADDRESS - TRACK, SECTOR, CYLINDER
8990      000036      $OPERC      =      $WRDL+16      ;OPERATION COUNT
8991      000042      $POSIT      =      $WRDL+22      ;SEEK COUNT
8992      000046      $TRANS      =      $WRDL+26      ;TOTAL BITS XFERED COUNT (R & W)
8993      000052      $READ      =      $WRDL+32      ;TOTAL BITS READ COUNT
8994      000056      $TOTAL      =      $WRDL+36      ;TOTAL ERRORS (ALL TYPES) COUNT
8995      000060      $SOFT      =      $WRDL+40      ;'SOFT' ERROR COUNT
8996      000062      $HARD      =      $WRDL+42      ;'HARD' ERROR COUNT
8997      000064      $SKI      =      $WRDL+44      ;'SKI' OR 'OCYL' ERROR COUNT
8998      000066      $MISPO      =      $WRDL+46      ;PROG DETECTED MISPOSITIONING ERROR S COUNT
8999      000070      $PASSC      =      $WRDL+50      ;PASS COUNTER
9000      000072      $FAIR      =      $WRDL+52      ;OPERATION QUEUE 'FAIRNESS' COUNT
```

;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS

9001
9002
9003

9004	000074	\$NCODE	=	\$WRDL+54	;NEXT OPERATION CODE
9005	000075	\$NPATC	=	\$NCODE+1	;NEXT PATTERN
9006	000076	\$NSEC	=	\$NCODE+2	;NEXT SECTOR
9007	000077	\$NTRK	=	\$NCODE+3	;NEXT TRACK
9008	000100	\$NCYL	=	\$NCODE+4	;NEXT CYLINDER
9009	000102	\$NWRDL	=	\$NCODE+6	;NEXT BUFFER SIZE
9010	000104	\$NEXT	=	\$NCODE+10	;PARAMETER SELECTION INDICATOR

;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES

9014	000106	MAXCYL	=	\$NCODE+12	;MAXIMUM CYLINDER ADDRESS
9015	000110	MINCYL	=	MAXCYL+2	;MINIMUM CYLINDER ADDRESS
9016	000112	MAXTRK	=	MAXCYL+4	;MAXIMUM TRACK ADDRESS
9017	000114	MINTRK	=	MAXCYL+6	;MINIMUM TRACK ADDRESS
9018	000116	MAXSEC	=	MAXCYL+10	;MAXIMUM SECTOR ADDRESS
9019	000120	MINSEC	=	MAXCYL+12	;MINIMUM SECTOR ADDRESS
9020	000122	\$FIRST	=	MAXCYL+14	;FIRST OPERATION INDICATOR

;BAD SECTOR/TRACK ADDRESS STORAGE AREA INDEX EQUATE

9024	000124	\$BDSEC	=	MAXCYL+16	;BAD SECTOR STORAGE TABLE
------	--------	---------	---	-----------	---------------------------

;DRIVE ID AREA INDEX EQUATE

9028	000224	\$DRVID	=	\$BDSEC+100	;DRIVE ID
------	--------	---------	---	-------------	-----------

;RH11/RP04/5/6 REGISTER EQUATES

9032	000234	\$RPCS1	=	\$DRVID+10	;RP04 REGISTER STORAGE
9033	000236	\$RPWC	=	\$RPCS1+2	
9034	000240	\$RPBA	=	\$RPCS1+4	
9035	000242	\$RPDA	=	\$RPCS1+6	
9036	000244	\$RPCS2	=	\$RPCS1+10	
9037	000246	\$RPDS1	=	\$RPCS1+12	
9038	000250	\$RPER1	=	\$RPCS1+14	
9039	000252	\$RPAS	=	\$RPCS1+16	
9040	000254	\$RPLA	=	\$RPCS1+20	
9041	000256	\$RPDB	=	\$RPCS1+22	
9042	000260	\$RPMR	=	\$RPCS1+24	
9043	000262	\$RPDT	=	\$RPCS1+26	
9044	000264	\$RPSN	=	\$RPCS1+30	
9045	000266	\$RPOF	=	\$RPCS1+32	
9046	000270	\$RPCA	=	\$RPCS1+34	
9047	000272	\$RPCC	=	\$RPCS1+36	
9048	000274	\$RPER2	=	\$RPCS1+40	
9049	000276	\$RPER3	=	\$RPCS1+42	
9050	000300	\$RPEC1	=	\$RPCS1+44	
9051	000302	\$RPEC2	=	\$RPCS1+46	

;BLOCK FOR DRIVE 0

9056	042036	000	000	DRIVE0: .BYTE	0,0	;DRIVE NUMBER
9057	042040	000005		.BLKW	5	
9058	042052	042272		.WORD	+\$RPCS1-\$REG	
9059	042054	000266		.BLKB	\$RPEC2-\$REG	


```
9060
9061
9062 ;BLOCK FOR DRIVE 1
9063
9064 042342 001 000 DRIVE1: .BYTE 1,0 ;DRIVE NUMBER
9065 042344 000005 .BLKW 5
9066 042356 042576 .WORD .+$RPCS1-$REG
9067 042360 000266 .BLKB $RPEC2-$REG
9068
9069
9070 ;BLOCK FOR DRIVE 2
9071
9072 042646 002 000 DRIVE2: .BYTE 2,0 ;DRIVE NUMBER
9073 042650 000005 .BLKW 5
9074 042662 043102 .WORD .+$RPCS1-$REG
9075 042664 000266 .BLKB $RPEC2-$REG
9076
9077
9078 ;BLOCK FOR DRIVE 3
9079
9080 043152 003 000 DRIVE3: .BYTE 3,0 ;DRIVE NUMBER
9081 043154 000005 .BLKW 5
9082 043166 043406 .WORD .+$RPCS1-$REG
9083 043170 000266 .BLKB $RPEC2-$REG
9084
9085
9086 ;BLOCK FOR DRIVE 4
9087
9088 043456 004 000 DRIVE4: .BYTE 4,0 ;DRIVE NUMBER
9089 043460 000005 .BLKW 5
9090 043472 043712 .WORD .+$RPCS1-$REG
9091 043474 000266 .BLKB $RPEC2-$REG
9092
9093
9094 ;BLOCK FOR DRIVE 5
9095
9096 043762 005 000 DRIVES: .BYTE 5,0 ;DRIVE NUMBER
9097 043764 000005 .BLKW 5
9098 043776 044216 .WORD .+$RPCS1-$REG
9099 044000 000266 .BLKB $RPEC2-$REG
9100
9101
9102 ;BLOCK FOR DRIVE 6
9103
9104 044266 006 000 DRIVE6: .BYTE 6,0 ;DRIVE NUMBER
9105 044270 000005 .BLKW 5
9106 044302 044522 .WORD .+$RPCS1-$REG
9107 044304 000266 .BLKB $RPEC2-$REG
9108
9109
9110 ;BLOCK FOR DRIVE 7
9111
9112 044572 007 000 DRIVE7: .BYTE 7,0 ;DRIVE NUMBER
9113 044574 000005 .BLKW 5
9114 044606 045026 .WORD .+$RPCS1-$REG
9115 044610 000266 .BLKB $RPEC2-$REG
```



```

9116
9117
9118 ;GENERAL PURPOSE DPB - USED BY 'READHD', 'RECALT', 'OFFSET',&'RTNCTR'
9119
9120 045076 000000 000000 177774 GENDPB: .WORD 0,0,-4,CYLDER
9121 045104 054746
9122 045106 000000 000000 045116 .WORD 0,0,GENREG,0
9123 045114 000000
9124 045116 000024 GENREG: .BLKW 24 ;REGISTER STORAGE IF ERROR
9125
9126 ;;*****
9127
9128 .SBTTL ERROR MESSAGES
9129
9130 ;;*****
9131
9132 045166 044122 030461 044440 EM1: .ASCIZ /RH11 INTERRUPT OCCURRED (RPAS = 0)/
9133 045174 052116 051105 052522
9134 045202 052120 047440 041503
9135 045210 051125 042522 020104
9136 045216 051050 040520 020123
9137 045224 020075 024460 000
9138
9139 045231 125 042516 050130 EM2: .ASCIZ /UNEXPECTED ATTENTION OCCURRED/
9140 045236 041505 042524 020104
9141 045244 052101 042524 052116
9142 045252 047511 020116 041517
9143 045260 052503 051122 042105
9144 045266 000
9145
9146 045267 115 051501 041123 EM3: .ASCIZ /MASSBUS PARITY ERROR (MCPE=1)/
9147 045274 051525 050040 051101
9148 045302 052111 020131 051105
9149 045310 047522 020122 046450
9150 045316 050103 036505 024461
9151 045324 000
9152
9153 045325 115 051501 041123 EM4: .ASCIZ /MASSBUS PARITY ERROR (PAR=1)/
9154 045332 051525 050040 051101
9155 045340 052111 020131 051105
9156 045346 047522 020122 050050
9157 045354 051101 030475 000051
9158
9159 045362 042101 051104 051505 EM5: .ASCIZ /ADDRESS PLUG CHANGE BIT SET/
9160 045370 020123 046120 043525
9161 045376 041440 040510 043516
9162 045404 020105 044502 020124
9163 045412 042523 000124
9164
9165 045416 044122 030461 042040 EM6: .ASCIZ /RH11 DIDN'T RESPOND TO ADDRESSING/
9166 045424 042111 023516 020124
9167 045432 042522 050123 047117
9168 045440 020104 047524 040440
9169 045446 042104 042522 051523
9170 045454 047111 000107
9171
  
```


CZRJDDO, RP04/5/6 MLT-DR LGC
CZRJDD.F11 06-JUN-79 08:32MACY11 30A(1052) 03-JUL-79 10:41 PAGE 177
ERROR MESSAGES

SEQ 0175

9172	045460	047125	047503	051122	EM10:	.ASCIZ	/UNCORRECTABLE MASSBUS PARITY ERROR/
9173	045466	041505	040524	046102			
9174	045474	020105	040515	051523			
9175	045502	052502	020123	040520			
9176	045510	044522	054524	042440			
9177	045516	051122	051117	000			
9178							
9179	045523	106	052101	046101	EM11:	.ASCIZ	/FATAL MASSBUS PARITY ERROR/
9180	045530	046440	051501	041123			
9181	045536	051525	050040	051101			
9182	045544	052111	020131	051105			
9183	045552	047522	000122				
9184							
9185	045556	042520	051522	051511	EM12:	.ASCIZ	/PERSISTENT DEVICE UNSAFE/
9186	045564	042524	052116	042040			
9187	045572	053105	041511	020105			
9188	045600	047125	040523	042506			
9189	045606	000					
9190							
9191	045607	117	042520	040522	EM13:	.ASCIZ	/OPERATION NOT COMPLETED WITHIN TIME LIMIT/
9192	045614	044524	047117	047040			
9193	045622	052117	041440	046517			
9194	045630	046120	052105	042105			
9195	045636	053440	052111	044510			
9196	045644	020116	044524	042515			
9197	045652	046040	046511	052111			
9198	045660	000					
9199							
9200	045661	104	044522	042526	EM14:	.ASCIZ	/DRIVE WENT OFFLINE/
9201	045666	053440	047105	020124			
9202	045674	043117	046106	047111			
9203	045702	000105					
9204							
9205	045704	047516	051040	051505	EM15:	.ASCIZ	/NO RESPONSE TO PORT REQUEST/
9206	045712	047520	051516	020105			
9207	045720	047524	050040	051117			
9208	045726	020124	042522	052521			
9209	045734	051505	000124				
9210							
9211	045740	042510	042101	051105	EM20:	.ASCIZ	/HEADER CRC ERROR/
9212	045746	041440	041522	042440			
9213	045754	051122	051117	000			
9214							
9215	045761	104	052101	020101	EM21:	.ASCIZ	/DATA CHECK ('DCK') ERROR/
9216	045766	044103	041505	020113			
9217	045774	023450	041504	023513			
9218	046002	020051	051105	047522			
9219	046010	000122					
9220							
9221	046012	051127	052111	020105	EM22:	.ASCIZ	/WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
9222	046020	044103	041505	020113			
9223	046026	051105	047522	020122			
9224	046034	020055	040504	040524			
9225	046042	041440	042510	045503			
9226	046050	024040	042047	045503			
9227	046056	024447	051440	052105			

9228	046064	000				
9229						
9230	046065	127	044522	042524	EM23:	.ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
9231	046072	041440	042510	045503		
9232	046100	042440	051122	051117		
9233	046106	026440	042040	052101		
9234	046114	020101	044103	041505		
9235	046122	020113	023450	041504		
9236	046130	023513	020051	047516		
9237	046136	020124	042523	000124		
9238						
9239	046144	042510	042101	051105	EM24:	.ASCIZ /HEADER READ ERROR - 'FMT' BIT DROPPED/
9240	046152	051040	040505	020104		
9241	046160	051105	047522	020122		
9242	046166	020055	043047	052115		
9243	046174	020047	044502	020124		
9244	046202	051104	050117	042520		
9245	046210	000104				
9246						
9247	046212	042510	042101	051105	EM25:	.ASCIZ /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
9248	046220	051040	040505	020104		
9249	046226	051105	047522	020122		
9250	046234	020055	042510	042101		
9251	046242	051105	041440	046517		
9252	046250	040520	042522	024040		
9253	046256	044047	042503	024447		
9254	046264	042440	051122	051117		
9255	046272	000				
9256						
9257	046273	106	051117	040515	EM26:	.ASCIZ /FORMAT ERROR ('FER')/
9258	046300	020124	051105	047522		
9259	046306	020122	023450	042506		
9260	046314	023522	000051			
9261						
9262	046320	042510	042101	051105	EM27:	.ASCIZ /HEADER COMPARE ('HCE') ERROR/
9263	046326	041440	046517	040520		
9264	046334	042522	024040	044047		
9265	046342	042503	024447	042440		
9266	046350	051122	051117	000		
9267						
9268	046355	115	051511	042503	EM30:	.ASCIZ /MISCELLANEOUS DRIVE ERROR/
9269	046362	046114	047101	047505		
9270	046370	051525	042040	044522		
9271	046376	042526	042440	051122		
9272	046404	051117	000			
9273						
9274	046407	117	042520	040522	EM31:	.ASCIZ /OPERATION INCOMPLETE ('OPI') ERROR/
9275	046414	044524	047117	044440		
9276	046422	041516	046517	046120		
9277	046430	052105	020105	023450		
9278	046436	050117	023511	020051		
9279	046444	051105	047522	000122		
9280						
9281	046452	051104	053111	020105	EM32:	.ASCIZ /DRIVE TIMING ('DTE') ERROR/
9282	046460	044524	044515	043516		
9283	046466	024040	042047	042524		

CZRJDDO, RP04/5/6 MLT-DR LGC
CZRJDD.P11 06-JUN-79 08:32

MACY11 30A(1052) 03-JUL-79 10:41 PAGE 179
ERROR MESSAGES

SEQ 0177

9284	046474	024447	042440	051122	
9285	046502	051117	000		
9286					
9287	046505	120	051101	052111	EM33: .ASCIZ /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
9288	046512	020131	023450	040520	
9289	046520	023522	020051	051105	
9290	046526	047522	020122	043101	
9291	046534	042524	020122	050117	
9292	046542	051105	052101	047511	
9293	046550	020116	052123	051101	
9294	046556	042524	000104		
9295					
9296	046562	051127	052111	020105	EM34: .ASCIZ /WRITE CLOCK FAILURE ('WCF') ERROR/
9297	046570	046103	041517	020113	
9298	046576	040506	046111	051125	
9299	046604	020105	023450	041527	
9300	046612	023506	020051	051105	
9301	046620	047522	000122		
9302					
9303	046624	047111	040526	044514	EM35: .ASCIZ /INVALID ADDRESS ('IAE') ERROR/
9304	046632	020104	042101	051104	
9305	046640	051505	020123	023450	
9306	046646	040511	023505	020051	
9307	046654	051105	047522	000122	
9308					
9309	046662	051127	052111	020105	EM36: .ASCIZ /WRITE LOCK ('WLE') ERROR/
9310	046670	047514	045503	024040	
9311	046676	053447	042514	024447	
9312	046704	042440	051122	051117	
9313	046712	000			
9314					
9315	046713	104	052101	020101	EM37: .ASCIZ /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
9316	046720	044103	041505	020113	
9317	046726	023450	041504	023513	
9318	046734	020051	042523	020124	
9319	046742	052504	044522	043516	
9320	046750	053440	044522	042524	
9321	046756	041440	042510	045503	
9322	046764	041440	046517	040515	
9323	046772	042116	000		
9324					
9325	046775	122	030510	020061	EM40: .ASCIZ /RH11 OR UNIBUS TRANSFER ERROR/
9326	047002	051117	052440	044516	
9327	047010	052502	020123	051124	
9328	047016	047101	043123	051105	
9329	047024	042440	051122	051117	
9330	047032	000			
9331					
9332	047033	102	051525	040440	EM41: .ASCIZ /BUS ADDRESS OR WORD COUNT INCORRECT/
9333	047040	042104	042522	051523	
9334	047046	047440	020122	047527	
9335	047054	042122	041440	052517	
9336	047062	052116	044440	041516	
9337	047070	051117	042522	052103	
9338	047076	000			
9339					

9340	047077	104	052101	020101	EM42: .ASCIZ /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
9341	047104	047503	050115	051101	
9342	047112	020105	051105	047522	
9343	047120	051522	026440	047040	
9344	047126	020117	052117	042510	
9345	047134	020122	051105	047522	
9346	047142	024122	024523	042040	
9347	047150	052105	041505	042524	
9348	047156	000104			
9349					
9350	047160	040503	023516	020124	EM43: .ASCIZ /CAN'T MATCH DATA READ WITH A PATTERN/
9351	047166	040515	041524	020110	
9352	047174	040504	040524	051040	
9353	047202	040505	020104	044527	
9354	047210	044124	040440	050040	
9355	047216	052101	042524	047122	
9356	047224	000			
9357					
9358	047225	105	051122	051117	EM44: .ASCIZ /ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11/
9359	047232	041040	052111	051450	
9360	047240	020051	042523	026124	
9361	047246	041040	052125	047040	
9362	047254	020117	051105	047522	
9363	047262	020122	044523	047107	
9364	047270	046101	042105	041040	
9365	047276	020131	044124	020105	
9366	047304	044122	030461	000	
9367					
9368	047311	105	041503	046040	EM45: .ASCIZ /ECC LOGIC FAILURE - POSITION REGISTER VALUE TOO LARGE/
9369	047316	043517	041511	043040	
9370	047324	044501	052514	042522	
9371	047332	026440	050040	051517	
9372	047340	052111	047511	020116	
9373	047346	042522	044507	052123	
9374	047354	051105	053040	046101	
9375	047362	042525	052040	047517	
9376	047370	046040	051101	042507	
9377	047376	000			
9378					
9379	047377	102	051525	040440	EM46: .ASCIZ /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
9380	047404	042104	042522	051523	
9381	047412	040440	042116	053440	
9382	047420	051117	020104	047503	
9383	047426	047125	020124	047516	
9384	047434	020124	047503	051516	
9385	047442	051511	042524	052116	
9386	047450	000			
9387					
9388	047451	123	042505	020113	EM50: .ASCIZ /SEEK INCOMPLETE ('SKI') OR OFF CYLINDER ('OCYL') ERROR/
9389	047456	047111	047503	050115	
9390	047464	042514	042524	024040	
9391	047472	051447	044513	024447	
9392	047500	047440	020122	043117	
9393	047506	020106	054503	044514	
9394	047514	042116	051105	024040	
9395	047522	047447	054503	023514	

9452	050124	020040	020040	050122							
9453	050132	040504	020040	020040							
9454	050140	050122	051501	020040							
9455	050146	020040	050122	040514							
9456	050154	020040	020040								
9457	050160	050122	041104	020040		.ASCIZ	/RPDB	RPMR	RPDT/<CR><LF>		
9458	050166	020040	050122	051115							
9459	050174	020040	020040	050122							
9460	050202	052104	005015	000							
9461											
9462	050207	122	051520	020116	DH16:	.ASCIZ	/RPSN	RPOF	RPCA	RPCC	STATUS/<CR><LF>
9463	050214	020040	051040	047520							
9464	050222	020106	020040	051040							
9465	050230	041520	020101	020040							
9466	050236	051040	041520	020103							
9467	050244	020040	051440	040524							
9468	050252	052524	006523	000012							
9469											
9470						.EVEN					
9471											
9472	050260	001244	000000		DT1:	.WORD	ATTN,0				
9473											
9474	050264	001242	033262	033264	DT2:	.WORD	DRIVE,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6,ATTN,0				
9475	050272	033266	033270	001244							
9476	050300	000000									
9477											
9478	050302	001242	040600	040602	DT3:	.WORD	DRIVE,RD.ADR,RD.WRD,0				
9479	050310	000000									
9480											
9481	050312	001242	041020	041016	DT4:	.WORD	DRIVE,WRT.AD,WRT.WD,RD.WRD,0				
9482	050320	040602	000000								
9483											
9484	050324	001170	000000		DT6:	.WORD	\$RPADR,0				
9485											
9486	050330	000234	000244	000246	DT14:	.WORD	\$RPCS1,\$RPCS2,\$RPDS1,\$RPER1,\$RPER2,\$RPER3,\$RPEC1,\$RPEC2,0				
9487	050336	000250	000274	000276							
9488	050344	000300	000302	000000							
9489											
9490	050352	000236	000240	000242	DT15:	.WORD	\$RPWC,\$RPBA,\$RPDA,\$RPAS,\$RPLA,\$RPDB,\$RPMR,\$RPDT,0				
9491	050360	000252	000254	000256							
9492	050366	000260	000262	000000							
9493											
9494	050374	000264	000266	000270	DT16:	.WORD	\$RPSN,\$RPOF,\$RPCA,\$RPCC,\$STATUS,0				
9495	050402	000272	000016	000000							
9496											
9497	050410	051120	051505	047105	LIN2C:	.ASCIZ	/PRESENT ORDER = /				
9498	050416	020124	051117	042504							
9499	050424	020122	020075	000							
9500	050431	040	050040	042522	LIN2P:	.ASCIZ	/ PREVIOUS ORDER = /				
9501	050436	044526	052517	020123							
9502	050444	051117	042504	020122							
9503	050452	020075	000								
9504	050455	052	042440	051122	LIN2S:	.ASCIZ	@* ERROR AT BAD TRACK/SECTOR@				
9505	050462	051117	040440	020124							
9506	050470	040502	020104	051124							
9507	050476	041501	027513	042523							

9508	050504	052103	051117	000	
9509	050511	105	051122	051117	LINM3: .ASCIZ /ERROR AT C/
9510	050516	040440	020124	000103	
9511	050524	052040	000		T: .ASCIZ / T/
9512	050527	120	042522	042523	LINN3: .ASCIZ /PRESENT ADDR = C/
9513	050534	052116	040440	042104	
9514	050542	020122	020075	000103	
9515	050550	051440	000		S: .ASCIZ / S/
9516	050553	040	020040	051120	LINP3: .ASCIZ / PREV ADDR = C/
9517	050560	053105	040440	042104	
9518	050566	020122	020075	000103	
9519	050574	052123	051101	020124	LINS3: .ASCIZ /START CYL = /
9520	050602	054503	020114	020075	
9521	050610	000			
9522	050611	040	042440	042116	LINEN3: .ASCIZ / END CYL = /
9523	050616	041440	046131	036440	
9524	050624	000040			
9525	050626	020040	041501	052524	LINA3: .ASCIZ / ACTUAL CYL = /
9526	050634	046101	041440	046131	
9527	050642	036440	000040		
9528	050646	020040	051124	020113	LINT3: .ASCIZ / TRK = /
9529	050654	020075	000		
9530	050657	040	050122	040503	LINCA3: .ASCIZ / RPCA = /
9531	050664	036440	000040		
9532	050670	050122	040504	036440	LINDA3: .ASCIZ /RPDA = /
9533	050676	000040			
9534	050700	050122	040502	036440	LINB3: .ASCIZ /RPBA = /
9535	050706	000040			
9536	050710	020040	050122	041527	LINW3: .ASCIZ / RPWC = /
9537	050716	036440	000040		
9538	050722	052123	051101	020124	LINST3: .ASCIZ /START TRK = /
9539	050730	051124	020113	020075	
9540	050736	000			
9541	050737	123	040524	052122	LINSS3: .ASCIZ /START SEC = /
9542	050744	051440	041505	036440	
9543	050752	000040			
9544	050754	052502	043106	051105	LINM4: .ASCIZ /BUFFER ADDR = /
9545	050762	040440	042104	020122	
9546	050770	020075	000		
9547	050773	040	051440	055111	LINS4: .ASCIZ / SIZE = /
9548	051000	020105	020075	000	
9549	051005	040	040440	052103	LINX4: .ASCIZ / ACTUAL NMBR WRDS XFRD = /
9550	051012	040525	020114	046516	
9551	051020	051102	053440	042122	
9552	051026	020123	043130	042122	
9553	051034	036440	000040		
9554	051040	047507	042117	042040	LIND5: .ASCIZ /GOOD DATA = /
9555	051046	052101	020101	020075	
9556	051054	000			
9557	051055	040	041040	042101	LINB5: .ASCIZ / BAD DATA = /
9558	051062	042040	052101	020101	
9559	051070	020075	000		
9560	051073	040	051440	041505	LINP5: .ASCIZ / SECT POS = /
9561	051100	020124	047520	020123	
9562	051106	020075	000		
9563	051111	110	040505	042504	LINS5: .ASCIZ /HEADER FROM ERROR SECTOR = /

9564	051116	020122	051106	046517	
9565	051124	042440	051122	051117	
9566	051132	051440	041505	047524	
9567	051140	020122	020075	000	
9568	051145	122	042520	030503	LINEP5: .ASCIZ /RPEC1 = /
9569	051152	036440	000040		
9570	051156	051040	042520	031103	LINE05: .ASCIZ / RPEC2 = /
9571	051164	036440	000040		
9572	051170	042523	052103	051117	LINB6: .ASCIZ /SECTOR IS ECC CORRECTABLE /
9573	051176	044440	020123	041505	
9574	051204	020103	047503	051122	
9575	051212	041505	040524	046102	
9576	051220	020105	000		
9577	051223	123	041505	047524	LINC6: .ASCIZ /SECTOR READ CORRECTLY /
9578	051230	020122	042522	042101	
9579	051236	041440	051117	042522	
9580	051244	052103	054514	000040	
9581	051252	047503	051122	041505	LING6: .ASCIZ /CORRECTED ON /
9582	051260	042524	020104	047117	
9583	051266	000040			
9584	051270	051040	052105	044522	LINR6: .ASCIZ / RETRIES/
9585	051276	051505	000		
9586	051301	125	041516	051117	LINU06: .ASCIZ /UNCORRECTABLE AFTER /
9587	051306	042522	052103	041101	
9588	051314	042514	040440	052106	
9589	051322	051105	000040		
9590	051326	020040	047524	040524	LIN7M: .ASCIZ / TOTAL MISPOS ERR = /
9591	051334	020114	044515	050123	
9592	051342	051517	042440	051122	
9593	051350	036440	000040		
9594	051354	051117	042504	051522	LIN7O: .ASCIZ /ORDERS:/
9595	051362	000072			
9596	051364	052040	052117	046101	LIN7P: .ASCIZ / TOTAL SEEKS = /
9597	051372	051440	042505	051513	
9598	051400	036440	000040		
9599	051404	052040	052117	046101	LIN7S: .ASCIZ / TOTAL SKI,OCYL ERR = /
9600	051412	051440	044513	047454	
9601	051420	054503	020114	051105	
9602	051426	020122	020075	000	
9603	051433	040	042440	051122	LIN7T: .ASCIZ / ERRORS:/
9604	051440	051117	035123	000	
9605	051445	040	053440	042122	LIN7X: .ASCIZ / WRDS XFR:/
9606	051452	020123	043130	035122	
9607	051460	000			
9608	051461	040	053440	042122	LIN7R: .ASCIZ / WRDS READ:/
9609	051466	020123	042522	042101	
9610	051474	000072			
9611	051476	044504	043106	051105	LIN8M: .ASCIZ /DIFFERENT ERROR DURING RETRY/
9612	051504	047105	020124	051105	
9613	051512	047522	020122	052504	
9614	051520	044522	043516	051040	
9615	051526	052105	054522	000	
9616	051533	104	052101	020101	LIN9B: .ASCIZ /DATA COMPARISON ERRORS/
9617	051540	047503	050115	051101	
9618	051546	051511	047117	042440	
9619	051554	051122	051117	000123	

9620	051562	020040	020040	020040	LIN9H: .ASCII /	GOOD	BAD/<CR><LF>
9621	051570	020040	043440	047517			
9622	051576	020104	020040	041040			
9623	051604	042101	005015				
9624	051610	047514	020103	020040	.ASCIZ /LOC	DATA	DATA/<CR><LF>
9625	051616	020040	042040	052101			
9626	051624	020101	020040	042040			
9627	051632	052101	006501	000012			
9628	051640	047514	020103	020040	LIN9I: .ASCIZ /LOC	DATA/<CR><LF>	
9629	051646	020040	042040	052101			
9630	051654	006501	000012				
9631	051660	047524	040524	020114	LIN9E: .ASCIZ /TOTAL COMPARE ERRORS = /		
9632	051666	047503	050115	051101			
9633	051674	020105	051105	047522			
9634	051702	051522	036440	000040			
9635	051710	044124	020105	040504	LIN9G: .ASCIZ /THE DATA COMPARED OK/<CR><LF>		
9636	051716	040524	041440	046517			
9637	051724	040520	042522	020104			
9638	051732	045517	005015	000			
9639	051737	105	051122	051117	LIN10A: .ASCIZ /ERROR BURST BEGINS AT WORD /		
9640	051744	041040	051125	052123			
9641	051752	041040	043505	047111			
9642	051760	020123	052101	053440			
9643	051766	051117	020104	000			
9644	051773	040	047111	042040	LIN10B: .ASCIZ / IN DATA FIELD OF ERROR SECTOR/<CR><LF>		
9645	052000	052101	020101	044506			
9646	052006	046105	020104	043117			
9647	052014	042440	051122	051117			
9648	052022	051440	041505	047524			
9649	052030	006522	000012				
9650	052034	051105	047522	020122	LIN10C: .ASCII /ERROR WAS NOT IN THE DATA READ - /<CR><LF>		
9651	052042	040527	020123	047516			
9652	052050	020124	047111	052040			
9653	052056	042510	042040	052101			
9654	052064	020101	042522	042101			
9655	052072	026440	006440	012			
9656	052077	105	041503	041440	.ASCIZ /ECC CORRECTION CAN'T BE PERFORMED/		
9657	052104	051117	042522	052103			
9658	052112	047511	020116	040503			
9659	052120	023516	020124	042502			
9660	052126	050040	051105	047506			
9661	052134	046522	042105	000			
9662	052141	105	041503	041440	LIN10H: .ASCII /ECC CORRECTION RESULTS/<CR><LF>		
9663	052146	051117	042522	052103			
9664	052154	047511	020116	042522			
9665	052162	052523	052114	006523			
9666	052170	012					
9667	052171	101	042104	020122	.ASCIZ /ADDR BAD CORRECTED /<CR><LF>		
9668	052176	020040	041040	042101			
9669	052204	020040	020040	041440			
9670	052212	051117	042522	052103			
9671	052220	042105	006440	000012			
9672	052226	047503	052116	047105	LIN11H: .ASCIZ /CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/<CR><LF>		
9673	052234	051524	047440	020106			
9674	052242	051105	047522	020122			
9675	052250	042523	052103	051117			

CZRJDDO, RP04/5/6 MLT-DR LGC
CZRJDD.P11 06-JUN-79 08:32

MACY11 30A(1052) 03-JUL-79 10:41 PAGE 186
ERROR MESSAGES

SEQ 0184

9676	052256	024040	042522	047520
9677	052264	052122	042105	040440
9678	052272	047502	042526	006451
9679	052300	000012		
9680	052302	042101	051104	020040
9681	052310	020040	040504	040524
9682	052316	005015	000	
9683	052321	040	040	
9684	052323	040		
9685	052324	000040		
9686				
9687				
9688				
9689	052326	005015	040527	047122
9690	052334	047111	035107	050040
9691	052342	047522	051107	046501
9692	052350	040515	046102	020105
9693	052356	051104	053111	051505
9694	052364	046440	054501	041040
9695	052372	020105	051525	042105
9696	052400	005015	000	
9697	052403	040	051120	043517
9698	052410	040522	046515	041101
9699	052416	042514	042055	044522
9700	052424	042526	053440	046111
9701	052432	020114	047516	020124
9702	052440	042502	052440	042523
9703	052446	000104		
9704	052450	051104	053111	000105
9705	052456	047440	043106	044514
9706	052464	042516	000	
9707	052467	040	047117	044514
9708	052474	042516	000	
9709	052477	040	047516	020124
9710	052504	042502	047111	020107
9711	052512	042524	052123	042105
9712	052520	000		
9713	052521	040	046101	042522
9714	052526	042101	020131	042502
9715	052534	047111	020107	042524
9716	052542	052123	042105	000
9717	052547	040	047516	020124
9718	052554	047101	051040	030120
9719	052562	027464	027465	000066
9720	052570	047040	052117	050040
9721	052576	042522	042523	052116
9722	052604	000		
9723	052605	040	047516	020124
9724	052612	053101	044501	040514
9725	052620	046102	000105	
9726	052624	052440	051516	043101
9727	052632	000105		
9728	052634	047125	052111	051440
9729	052642	040524	052524	035123
9730	052650	005015	000012	
9731	052654	050122	032060	000

.ASCIZ /ADDR DATA/<CR><LF>

LIN4SP: .ASCII / /
LINSPO: .ASCII / /
LINSPO: .ASCIZ / /

.SBTTL TELETYPE MESSAGES

USE: .ASCIZ <CR><LF>/WARNING: PROGRAMMABLE DRIVES MAY BE USED/<CR><LF>

NOUSE: .ASCIZ / PROGRAMMABLE-DRIVE WILL NOT BE USED/

UNTMSG: .ASCIZ /DRIVE/
UNTOFF: .ASCIZ / OFFLINE/

UNTON: .ASCIZ / ONLINE/

UNTNOT: .ASCIZ / NOT BEING TESTED/

UNTASN: .ASCIZ / ALREADY BEING TESTED/

NOTRP: .ASCIZ @ NOT AN RP04/5/6@

NOTPRS: .ASCIZ / NOT PRESENT/

NOTAVL: .ASCIZ / NOT AVAILABLE/

NOTSAF: .ASCIZ / UNSAFE/

SYSTAT: .ASCIZ /UNIT STATUS:/<CR><LF><LF>

RP04B: .ASCIZ /RP04/

9732	052661	122	030120	000065	RP05: .ASCIZ /RP05/
9733	052666	050122	033060	000	RP06: .ASCIZ /RP06/
9734	052673	104	044522	042526	STATHD: .ASCII /DRIVE PERFORMANCE SUMMARY/<CR><LF>
9735	052700	050040	051105	047506	
9736	052706	046522	047101	042503	
9737	052714	051440	046525	040515	
9738	052722	054522	005015		
9739	052726	051104	020126	040520	.ASCII /DRV PASS ORDEHS SEKS WRDS XFER WRDS READ /
9740	052734	051523	047440	042122	
9741	052742	051105	020123	020040	
9742	052750	042523	045505	020123	
9743	052756	020040	051127	051504	
9744	052764	054040	042506	020122	
9745	052772	020040	051127	051504	
9746	053000	051040	040505	020104	
9747	053006	047523	052106	044040	.ASCIZ /SOFT HARD SKI MISP OTHER/<CR><LF>
9748	053014	051101	020104	051440	
9749	053022	044513	046440	051511	
9750	053030	020120	052117	042510	
9751	053036	006522	000012		
9752	053042	047504	042516	005015	PDONE: .ASCIZ /DONE/<CR><LF><LF>
9753	053050	000012			
9754	053052	037407	040506	040524	DROPNG: .ASCIZ <07>/?FATAL OR EXCESSIVE ERRORS/
9755	053060	020114	051117	042440	
9756	053066	041530	051505	044523	
9757	053074	042526	042440	051122	
9758	053102	051117	000123		
9759	053106	047105	020104	043117	ENDPAS: .ASCIZ /END OF PASS/
9760	053114	050040	051501	000123	
9761	053122	005015	047105	020104	ENDTST: .ASCIZ <CR><LF>/END OF TEST/
9762	053130	043117	052040	051505	
9763	053136	000124			
9764	053140	005015	051104	053111	DEASSG: .ASCIZ <CR><LF>/DRIVE DEASSIGNED/
9765	053146	020105	042504	051501	
9766	053154	044523	047107	042105	
9767	053162	000			
9768	053163	015	025012	025052	DRNUM: .ASCIZ <CR><LF>/***** DRIVE #/
9769	053170	025052	025052	025052	
9770	053176	020052	051104	053111	
9771	053204	020105	000043		
9772	053210	051440	040524	052122	ASGND: .ASCIZ / STARTED/<CR><LF>
9773	053216	042105	005015	000	
9774	053223	015	037412	023440	NEDCLK: .ASCIZ <CR><LF>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CR><LF>
9775	053230	023514	047440	020122	
9776	053236	050047	020047	046103	
9777	053244	041517	020113	042522	
9778	053252	052521	051111	042105	
9779	053260	047440	020116	054523	
9780	053266	052123	046505	005015	
9781	053274	000			
9782	053275	056	000		PERIOD: .ASCIZ /./
9783	053277	077	000		QUES: .ASCIZ /?/
9784	053301	111	053116	046101	INVLD: .ASCIZ /INVALID COMMAND/<CR><LF>
9785	053306	042111	041440	046517	
9786	053314	040515	042116	005015	
9787	053322	000			

CZRJDD0, RP04/5/6 MLT-DR LGC
CZRJDD.P11 06-JUN-79 08:32

MACY11 30A(1052) 03-JUL-79 10:41 PAGE 188
TELETYPE MESSAGES

SEQ 0186

9788	053323	015	042412	052116	ENTCOM: .ASCIZ <CR><LF>/ENTER COMMANDS: /<CR><LF>
9789	053330	051105	041440	046517	
9790	053336	040515	042116	035123	
9791	053344	006440	000012		
9792	053350	047105	042524	020122	ENTDRV: .ASCIZ /ENTER I.D. FOR DRV #/
9793	053356	027111	027104	043040	
9794	053364	051117	042040	053122	
9795	053372	021440	000		
9796	053375	015	042412	052116	ENTLMT: .ASCIZ <CR><LF>/ENTER ADDRESS LIMITS FOR DRV #/
9797	053402	051105	040440	042104	
9798	053410	042522	051523	046040	
9799	053416	046511	052111	020123	
9800	053424	047506	020122	051104	
9801	053432	020126	000043		
9802	053436	047105	042524	020122	ENTADR: .ASCIZ @ENTER BAD TRK/SEC ADRS FOR DRV #@
9803	053444	040502	020104	051124	
9804	053452	027513	042523	020103	
9805	053460	042101	051522	043040	
9806	053466	051117	042040	053122	
9807	053474	021440	000		
9808	053477	072	000		COLON: .ASCIZ /:/
9809	053501	015	042012	052101	DATEIS: .ASCIZ <CR><LF>/DATE: /
9810	053506	035105	000040		
9811	053512	005015	050117	051105	IDIS: .ASCIZ <CR><LF>/OPERATOR I.D.: /
9812	053520	052101	051117	044440	
9813	053526	042056	035056	000040	
9814	053534	005015	042012	053122	HEDLIN: .ASCIZ <CR><LF><LF>/DRV DRV I.D./<CR><LF>
9815	053542	020040	051104	020126	
9816	053550	027111	027104	005015	
9817	053556	000			
9818	053557	116	047117	006505	NONE: .ASCIZ /NONE/<CR><LF>
9819	053564	000012			
9820	053566	020077	047111	040526	BADENT: .ASCIZ /? INVALID ENTRY/<CR><LF>
9821	053574	044514	020104	047105	
9822	053602	051124	006531	000012	
9823	053610	054523	052123	046505	BUSY: .ASCIZ /SYSTEM BUSY.../<CR><LF>
9824	053616	041040	051525	027131	
9825	053624	027056	005015	000	
9826	053631	015	050012	047522	INTDON: .ASCII <CR><LF>/PROGRAM INITIALIZATION COMPLETE/
9827	053636	051107	046501	044440	
9828	053644	044516	044524	046101	
9829	053652	055111	052101	047511	
9830	053660	020116	047503	050115	
9831	053666	042514	042524		
9832	053672	005015	054524	042520	.ASCIZ <CR><LF>/TYPE A 'CONTROL C' TO ENTER COMMANDS/<CR><LF><LF>
9833	053700	040440	023440	047503	
9834	053706	052116	047522	020114	
9835	053714	023503	052040	020117	
9836	053722	047105	042524	020122	
9837	053730	047503	046515	047101	
9838	053736	051504	005015	000012	
9839					
9840					.EVEN
9841					
9842					;PARAMETER ENTRY TABLE
9843					


```
9844 053744 054072 000000 001404 PARLST: .WORD PAR1,0,MAXDL
9845 053752 054100 177777 001410 .WORD PAR2,-1,INTRVL
9846 053760 054230 177777 001402 .WORD PAR19,-1,PASCNT
9847 053766 054107 177777 001414 .WORD PAR3,-1,CMPLMT
9848 053774 054177 000001 001420 .WORD PAR11,1,WCSEL
9849 054002 054205 000007 001422 .WORD PAR14,7,RATIO
9850 054010 054222 000001 001430 .WORD PAR16,1,ENDET
9851 054016 054170 000001 001416 .WORD PAR10,1,FORMAT
9852 054024 054213 000001 001424 .WORD PAR15,1,AUTOCK
9853 054032 054237 000001 001426 .WORD PAR20,1,NOTPRT
9854 054040 000000 .WORD 0 ;TABLE TERMINATOR
9855
9856 054042 047105 042524 020122 ASKPAR: .ASCIZ /ENTER PARAMETERS: /
9857 054050 040520 040522 042515
9858 054056 042524 051522 020072
9859 054064 000040
9860 054066 027440 000040 SLASH: .ASCIZ @ / @
9861
9862 054072 040515 042130 000114 PAR1: .ASCIZ /MAXDL/
9863 054100 047111 051124 046126 PAR2: .ASCIZ /INTRVL/
9864 054106 000
9865 054107 103 050115 046514 PAR3: .ASCIZ /CMPLMT/
9866 054114 000124
9867 054116 040515 041530 046131 PAR4: .ASCIZ /MAXCYL/
9868 054124 000
9869 054125 115 047111 054503 PAR5: .ASCIZ /MINCYL/
9870 054132 000114
9871 054134 040515 052130 045522 PAR6: .ASCIZ /MAXTRK/
9872 054142 000
9873 054143 115 047111 051124 PAR7: .ASCIZ /MINTRK/
9874 054150 000113
9875 054152 040515 051530 041505 PAR8: .ASCIZ /MAXSEC/
9876 054160 000
9877 054161 115 047111 042523 PAR9: .ASCIZ /MINSEC/
9878 054166 000103
9879 054170 047506 046522 052101 PAR10: .ASCIZ /FORMAT/
9880 054176 000
9881 054177 127 051503 046105 PAR11: .ASCIZ /WCSEL/
9882 054204 000
9883 054205 122 052101 047511 PAR14: .ASCIZ /RATIO/
9884 054212 000
9885 054213 101 052125 041517 PAR15: .ASCIZ /AUTOCK/
9886 054220 000113
9887 054222 047105 042504 000124 PAR16: .ASCIZ /ENDET/
9888 054230 040520 041523 052116 PAR19: .ASCIZ /PASCNT/
9889 054236 000
9890 054237 116 052117 051120 PAR20: .ASCIZ /NOTPRT/
9891 054244 000124
9892
9893 .EVEN
9894
9895 ;PARAMETER TABLE POINTERS FOR ADDRESS LIMITS
9896
9897 054246 054266 TABLE: .WORD TABLE0 ;PARAMETER TABLE FOR DRIVE 0
9898 054250 054334 .WORD TABLE1 ;PARAMETER TABLE FOR DRIVE 1
9899 054252 054402 .WORD TABLE2 ;PARAMETER TABLE FOR DRIVE 2
```


9900	054254	054450			.WORD	TABLE3		:PARAMETER TABLE FOR DRIVE 3
9901	054256	054516			.WORD	TABLE4		:PARAMETER TABLE FOR DRIVE 4
9902	054260	054564			.WORD	TABLE5		:PARAMETER TABLE FOR DRIVE 5
9903	054262	054632			.WORD	TABLE6		:PARAMETER TABLE FOR DRIVE 6
9904	054264	054700			.WORD	TABLE7		:PARAMETER TABLE FOR DRIVE 7
9905								
9906								
9907								:PARAMETER TABLE FOR ADDRESS LIMITS
9908	054266	054125	000000	042146	TABLE0:	.WORD	PAR5,0,MINCYL+DRIVE0	
9909	054274	054116	000000	042144		.WORD	PAR4,0,MAXCYL+DRIVE0	
9910	054302	054143	000022	042152		.WORD	PAR7,18.,MINTRK+DRIVE0	
9911	054310	054134	000022	042150		.WORD	PAR6,18.,MAXTRK+DRIVE0	
9912	054316	054161	000025	042156		.WORD	PAR9,21.,MINSEC+DRIVE0	
9913	054324	054152	000025	042154		.WORD	PAR8,21.,MAXSEC+DRIVE0,0	
9914	054332	000000						
9915								
9916	054334	054125	000000	042452	TABLE1:	.WORD	PAR5,0,MINCYL+DRIVE1	
9917	054342	054116	000000	042450		.WORD	PAR4,0,MAXCYL+DRIVE1	
9918	054350	054143	000022	042456		.WORD	PAR7,18.,MINTRK+DRIVE1	
9919	054356	054134	000022	042454		.WORD	PAR6,18.,MAXTRK+DRIVE1	
9920	054364	054161	000025	042462		.WORD	PAR9,21.,MINSEC+DRIVE1	
9921	054372	054152	000025	042460		.WORD	PAR8,21.,MAXSEC+DRIVE1,0	
9922	054400	000000						
9923								
9924	054402	054125	000000	042756	TABLE2:	.WORD	PAR5,0,MINCYL+DRIVE2	
9925	054410	054116	000000	042754		.WORD	PAR4,0,MAXCYL+DRIVE2	
9926	054416	054143	000022	042762		.WORD	PAR7,18.,MINTRK+DRIVE2	
9927	054424	054134	000022	042760		.WORD	PAR6,18.,MAXTRK+DRIVE2	
9928	054432	054161	000025	042766		.WORD	PAR9,21.,MINSEC+DRIVE2	
9929	054440	054152	000025	042764		.WORD	PAR8,21.,MAXSEC+DRIVE2,0	
9930	054446	000000						
9931								
9932	054450	054125	000000	043262	TABLE3:	.WORD	PAR5,0,MINCYL+DRIVE3	
9933	054456	054116	000000	043260		.WORD	PAR4,0,MAXCYL+DRIVE3	
9934	054464	054143	000022	043266		.WORD	PAR7,18.,MINTRK+DRIVE3	
9935	054472	054134	000022	043264		.WORD	PAR6,18.,MAXTRK+DRIVE3	
9936	054500	054161	000025	043272		.WORD	PAR9,21.,MINSEC+DRIVE3	
9937	054506	054152	000025	043270		.WORD	PAR8,21.,MAXSEC+DRIVE3,0	
9938	054514	000000						
9939								
9940	054516	054125	000000	043566	TABLE4:	.WORD	PAR5,0,MINCYL+DRIVE4	
9941	054524	054116	000000	043564		.WORD	PAR4,0,MAXCYL+DRIVE4	
9942	054532	054143	000022	043572		.WORD	PAR7,18.,MINTRK+DRIVE4	
9943	054540	054134	000022	043570		.WORD	PAR6,18.,MAXTRK+DRIVE4	
9944	054546	054161	000025	043576		.WORD	PAR9,21.,MINSEC+DRIVE4	
9945	054554	054152	000025	043574		.WORD	PAR8,21.,MAXSEC+DRIVE4,0	
9946	054562	000000						
9947								
9948	054564	054125	000000	044072	TABLE5:	.WORD	PAR5,0,MINCYL+DRIVE5	
9949	054572	054116	000000	044070		.WORD	PAR4,0,MAXCYL+DRIVE5	
9950	054600	054143	000022	044076		.WORD	PAR7,18.,MINTRK+DRIVE5	
9951	054606	054134	000022	044074		.WORD	PAR6,18.,MAXTRK+DRIVE5	
9952	054614	054161	000025	044102		.WORD	PAR9,21.,MINSEC+DRIVE5	
9953	054622	054152	000025	044100		.WORD	PAR8,21.,MAXSEC+DRIVE5,0	
9954	054630	000000						
9955								


```

9956 054632 054125 000000 044376 TABLE6: .WORD PAR5,0,MINCYL+DRIVE6
9957 054640 054116 000000 044374 .WORD PAR4,0,MAXCYL+DRIVE6
9958 054646 054143 000022 044402 .WORD PAR7,18.,MINTRK+DRIVE6
9959 054654 054134 000022 044400 .WORD PAR6,18.,MAXTRK+DRIVE6
9960 054662 054161 000025 044406 .WORD PAR9,21.,MINSEC+DRIVE6
9961 054670 054152 000025 044404 .WORD PAR8,21.,MAXSEC+DRIVE6,0
9962 054676 000000
9963
9964 054700 054125 000000 044702 TABLE7: .WORD PAR5,0,MINCYL+DRIVE7
9965 054706 054116 000000 044700 .WORD PAR4,0,MAXCYL+DRIVE7
9966 054714 054143 000022 044706 .WORD PAR7,18.,MINTRK+DRIVE7
9967 054722 054134 000022 044704 .WORD PAR6,18.,MAXTRK+DRIVE7
9968 054730 054161 000025 044712 .WORD PAR9,21.,MINSEC+DRIVE7
9969 054736 054152 000025 044710 .WORD PAR8,21.,MAXSEC+DRIVE7,0
9970 054744 000000
9971
9972
9973 054746 000000 000000 000000 CYLDER: .WORD 0,0,0,0 ;HEADER BUFFER FOR 'READHD' ROUTINE
9974 054754 000000
9975
9976 054756 ENDPGM = . ;LAST LOCATION OF PROG + 2
9977
9978 ;:*****
9979
9980 ;ROUTINE TO GET THE DATE AND THE OPERATOR FROM THE OPERATOR
9981 ;CALL:
9982 ; JSR PC,OPRDAT
9983 ; RETURN
9984 ;
9985 ;NOTE: THIS ROUTINE IS ENTERED ONLY AT INITIAL START
9986
9987 054756 104401 055070 OPRDAT: TYPE ,ENTDAT ;'ENTER DATE'
9988 054762 104411 RDLIN ;READ THE ENTRY
9989 054764 012605 MOV (SP)+,R5 ;PUT THE ENTRY ADDRESS INTO R5
9990 054766 112537 001220 MOV (R5)+,DATE ;STORE THE DATE
9991 054772 112537 001221 MOV (R5)+,DATE+1 ;STORE THE DATE
9992 054776 112537 001222 MOV (R5)+,DATE+2 ;STORE THE DATE
9993 055002 112537 001223 MOV (R5)+,DATE+3 ;STORE THE DATE
9994 055006 112537 001224 MOV (R5)+,DATE+4 ;STORE THE DATE
9995 055012 112537 001225 MOV (R5)+,DATE+5 ;STORE THE DATE
9996 055016 112537 001226 MOV (R5)+,DATE+6 ;STORE THE DATE
9997 055022 112537 001227 MOV (R5)+,DATE+7 ;STORE THE DATE
9998 055026 104401 055107 TYPE ,ENTID ;'ENTER OPERATOR I.D.'
9999 055032 104411 RDLIN ;READ THE ENTRY
10000 055034 012605 MOV (SP)+,R5 ;ENTRY ADDRESS
10001 055036 112537 001232 MOV (R5)+,OPERID ;STORE THE I.D.
10002 055042 112537 001233 MOV (R5)+,OPERID+1 ;STORE THE I.D.
10003 055046 112537 001234 MOV (R5)+,OPERID+2 ;STORE THE I.D.
10004 055052 112537 001235 MOV (R5)+,OPERID+3 ;STORE THE I.D.
10005 055056 112537 001236 MOV (R5)+,OPERID+4 ;STORE THE I.D.
10006 055062 112537 001237 MOV (R5)+,OPERID+5 ;STORE THE I.D.
10007 055066 000207 RTS PC ;RETURN
10008
10009 055070 005015 047105 042524 ENTDAT: .ASCIZ <CR><LF>/ENTER DATE: /
10010 055076 020122 040504 042524
10011 055104 020072 000
    
```


10012 055107 105 052116 051105
10013 055114 047440 042520 040522
10014 055122 047524 020122 027111
10015 055130 027104 020072 000
10016 055136
10017
10018
10019
10020
10021
10022
10023
10024
10025
10026 055136 010046
10027 055140 010146
10028 055142 013746 000004
10029 055146 013746 000006
10030 055152 010600
10031
10032 055154 104400
10033 055156 012637 000006
10034 055162 012737 055202 000004
10035 055170 012701 020000
10036 055174 005711
10037 055176 005721
10038 055200 000775
10039 055202 162701 000002
10040 055206 010006
10041 055210 012637 000006
10042 055214 012637 000004
10043 055220 010137 055232
10044 055224 012601
10045 055226 012600
10046 055230 000207
10047 055232 000000
10048
10049
10050
10051
10052
10053
10054
10055
10056
10057
10058
10059
10060 055234 177700
10061 055236 000776
10062 055240 000000
10063
10064 055242 005737 001260
10065 055246 001456
10066 055250 005037 001260
10067 055254 012700 001170

```
ENTID: .ASCIZ /ENTER OPERATOR I.D.: /  
.EVEN  
.SBTTL ROUTINE TO SIZE MEMORY  
*****  
*CALL:  
* JSR PC,$SIZE  
* RETURN  
*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION  
$SIZE: MOV RO,-(SP) ;;SAVE R0 ON THE STACK  
MOV R1,-(SP) ;;SAVE R1 ON THE STACK  
MOV @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC  
MOV @#ERRVEC+2,-(SP)  
MOV SP,R0 ;;SAVE THE STACK POINTER  
;;SET THE ERRVEC PS TO THE PRESENT PS  
TRAP ;;PUSH OLD PSW AND PC ON STACK  
MOV (SP)+,@#ERRVEC+2 ;;SAVE THE PSW IN @#ERRVEC+2  
MOV #2,@#ERRVEC ;;SET FOR TIMEOUT  
MOV #20000,R1 ;;FIRST ADDRESS  
1$: TST (R1) ;;TEST THIS ADDRESS  
TST (R1)+ ;;STEP TO NEXT ADDRESS  
BR 1$ ;;TRY ANOTHER  
2$: SUB #2,R1 ;;DROP BACK  
MOV R0,SP ;;RESTORE THE STACK  
MOV (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR  
MOV (SP)+,@#ERRVEC  
MOV R1,$LSTAD ;;LAST ADDRESS  
MOV (SP)+,R1 ;;RESTORE R1  
MOV (SP)+,R0 ;;RESTORE R0  
RTS PC  
$LSTAD: .WORD 0 ;;CONTAINS THE LAST ADDRESS  
.SBTTL BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11  
;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS  
;OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.  
;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF  
;REQUIRED.  
;NOTE: THIS ROUTINE DESTROYS R0-R4  
;CALL  
;:  
; JSR PC,BUSADR  
; RETURN  
;:  
HIAD: .WORD 177700  
HIVEC: .WORD 776  
BOUND: .WORD 0  
BUSADR: TST CHGADR ;INPUT FROM TTY REQUESTED?  
BEQ 7$ ;NO--BRANCH  
CLR CHGADR ;YES--CLEAR THE REQUEST FLAG  
1$: MOV #RPNADR,R0 ;FIRST ADDRESS
```



```
10068 055260 104401 055466          TYPE      ,MRPCS1      ;"RPCS1="
10069 055264 012046          MOV      (R0)+,-(SP)  ;PRESENT RPCS1 ADDRESS
10070 055266 104402          TYP0C    ;TYPE IT
10071 055270 104401 052323          TYPE      ,LINSPI    ;2 SPACES
10072 055274 104411          RDLIN    ;GET THE ENTRY
10073 055276 012601          MOV      (SP)+,R1    ;ADDRESS OF ASCII TEXT
10074 055300 013737 055234 055240  MOV      HIAD,BOUND   ;SET THE ADDRESS MAX
10075 055306 004537 055510          JSR      R5,CK.NUM   ;CHECK THE NUMBER
10076 055312 055332          3$      ;CARRIAGE RETURN ONLY ENTERED
10077 055314 055404          7$      ;PERIOD ONLY ENTERED
10078 055316 055254          1$      ;ILLEGAL INPUT
10079 055320 055326          2$      ;TERMINATED WITH A CARRIAGE RETURN
10080 055322 055254          1$      ;TERMINATED WITH A ".
10081 055324 055400          4$      ;TERMINATED WITH A ".
10082 055326 010260 177776          2$:     MOV      R2,-2(R0)  ;SAVE NEW RPCS1
10083 055332 104401 055477          3$:     TYPE      ,MRHVEC  ;"RHVEC="
10084 055336 012046          MOV      (R0)+,-(SP)  ;PRESENT RH11 VECTOR ADDRESS ON THE STACK
10085 055340 104402          TYP0C    ;TYPE IT
10086 055342 104401 052323          TYPE      ,LINSPI    ;2 SPACES
10087 055346 104411          RDLIN    ;READ THE ENTRY
10088 055350 012601          MOV      (SP)+,R1    ;ASCII TEXT ADDRESS
10089 055352 013737 055236 055240  MOV      HIVEC,BOUND  ;SET THE VECTOR-MAX
10090 055360 004537 055510          JSR      R5,CK.NUM   ;CHECK THE NUMBER
10091 055364 055404          7$      ;CARRIAGE RETURN ONLY ENTERED
10092 055366 055404          7$      ;PERIOD ONLY ENTERED
10093 055370 055332          3$      ;ILLEGAL INPUT
10094 055372 055400          4$      ;TERMINATED WITH A CARRIAGE RETURN
10095 055374 055332          3$      ;TERMINATED WITH A ".
10096 055376 055400          4$      ;TERMINATED WITH A ".
10097 055400 010260 177776          4$:     MOV      R2,-2(R0)  ;SAVE INPUT
10098 055404 013701 000004          7$:     MOV      ERRVEC,R1  ;SAVE THE ERROR VECTOR
10099 055410 012737 055444 000004  MOV      #8$,ERRVEC  ;SETUP FOR TRAP
10100 055416 005777 123546          TST      @$RFADR     ;CHECK FOR RH11
10101 055422 010137 000004          MOV      R1,ERRVEC  ;RESTORE ERROR VECTOR
10102 055426 012700 001170          MOV      #RPAADR,R0 ;FIRST ADDRESS OF NEW PARAMETERS
10103 055432 012701 033430          MOV      #RPADR,R1  ;FIRST ADDRESS OF WHERE TO PUT THEM
10104 055436 012021          MOV      (R0)+,(R1)+ ;BUS ADDRESS
10105 055440 012021          MOV      (R0)+,(R1)+ ;VECTOR ADDRESS
10106 055442 000207          RTS      PC         ;RETURN
10107 055444 010137 000004          8$:     MOV      R1,ERRVEC  ;RESTORE ERROR VECTOR
10108 055450 022626          CMP      (SP)+,(SP)+ ;CLEAN OFF THE STACK
10109 055452 104006          ERROR   6         ;REPORT THE ERROR
10110 055454 005737 000042          TST      @#42       ;IS THERE A MONITOR?
10111 055460 001675          BEQ     1$         ;NO--GO ASK FOR ADDRESS
10112 055462 000137 005432          JMP     $GET42      ;GO TO END OF PROGRAM
10113
10114 055466 050122 051503 020061 MRPCS1: .ASCIZ @RPCS1 = @
10115 055474 020075          000
10116 055477          122 053110 041505 MRHVEC: .ASCIZ @RHVEC = @
10117 055504 036440 000040
10118
10119          .SBTTL CK.NUM - CHECK NUMBER (OCTAL)
10120          ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
10121          ;AND FORMS AN OCTAL NUMBER IN R2
10122          ;CALL:
10123          ;      MOV      #ADR,R1          ;ADDRESS OF ASCIZ STRING
```



```
10124 :      MOV      #NUM,R2      ;MAX SIZE OF INPUT NUMBER
10125 :      JSR      R5,CK.NUM     ;GO FORM THE NUMBER
10126 :      RETURN   ADR1          ;"CR" ONLY ENTERED -- R2 = 0
10127 :      RETURN   ADR2          ;"PERIOD" ONLY ENTERED -- R2 = 0
10128 :      RETURN   ADR3          ;ILLEGAL CHARACTER IN THE INPUT STRING
10129 :      RETURN   ADR4          ;"CR" ENTERED -- R2 = NUMBER
10130 :      RETURN   ADR5          ;"COMMA" -- R2 = NUMBER
10131 :      RETURN   ADR6          ;"PERIOD" -- R2 = NUMBER
10132
10133 055510 010446      CK.NUM: MOV      R4,-(SP)      ;SAVE R4
10134 055512 010346      MOV      R3,-(SP)      ;SAVE R3
10135 055514 010246      MOV      R2,-(SP)      ;SAVE R2
10136 055516 005004      CLR      R4            ;RETURN POINTER
10137 055520 005003      CLR      R3            ;START NUMBER AT ZERO
10138 055522 005002      CLR      R2            ;STORE RESULT
10139 055524 004537 027656 JSR      R5,CK.CHR     ;CHECK ONE CHARACTER
10140 055530 055630      6$      ;ILLEGAL CHARACTER
10141 055532 055634      8$      ;CARRIAGE RETURN
10142 055534 055630      6$      ;" "
10143 055536 055632      7$      ;" "
10144 055540 055544      1$      ;DIGIT 0-7
10145 055542 055630      6$      ;DIGIT 8-9
10146 055544 062705 000004 1$: ADD      #4,R5      ;INCREMENT RETURN PAST "CR" AND "PERIOD" ONLY RETURNS
10147 055550 006303      2$: ASL      R3        ;FOR THE OCTAL NUMBER IN R3
10148 055552 103426      BCS      6$          ;DON'T LET IT GET TO BIG
10149 055554 006303      ASL      R3
10150 055556 103424      BCS      6$
10151 055560 006303      ASL      R3
10152 055562 103422      BCS      6$
10153 055564 060203      ADD      R2,R3
10154 055566 004537 027656 JSR      R5,CK.CHR     ;CHECK ONE CHARACTER
10155 055572 055634      8$      ;ILLEGAL CHARACTER
10156 055574 055616      5$      ;CARRIAGE RETURN
10157 055576 055614      4$      ;" "
10158 055600 055606      3$      ;" "
10159 055602 055550      2$      ;DIGIT 0-7
10160 055604 055634      8$      ;DIGIT 8-9
10161 055606 105711      3$: TSTB     (R1)      ;DOES A "CR" FOLLOW THE "PERIOD"
10162 055610 001011      BNE      8$          ;BR IF NOT
10163 055612 005724      TST      (R4)+      ;INCREMENT THE RETURN
10164 055614 005724      4$: TST      (R4)+      ;INCREMENT THE RETURN INDEX
10165 055616 005724      5$: TST      (R4)+      ;INCREMENT THE RETURN INDEX
10166 055620 023703 055240 CMP      BOUND,R3     ;INPUT VALUE TOO LARGE?
10167 055624 101003      BHI      8$          ;BR IF IT IS
10168 055626 000401      BR       7$          ;BR IF NOT
10169 055630 005725      6$: TST      (R5)+      ;INCREMENT THE RETURN ADDRESS
10170 055632 005725      7$: TST      (R5)+      ;INCREMENT THE RETURN ADDRESS
10171 055634 060405      8$: ADD      R4,R5      ;SETUP FOR PROPER RETURN
10172 055636 010302      MOV      R3,R2      ;LOAD ENTERED VALUE
10173 055640 005726      TST      (SP)+      ;CLEAN OFF THE STACK
10174 055642 012603      MOV      (SP)+,R3    ;RESTORE R3
10175 055644 012604      MOV      (SP)+,R4    ;RESTORE R4
10176 055646 011505      MOV      (R5),R5     ;GET RETURN ADDRESS
10177 055650 000205      RTS      R5          ;RETURN
10178
10179
```


CZRJDDO, RP04/5/6 MLT-DR LGC
CZRJDD.P11 06-JUN-79 08:32

MACY11 30A(1052) 03-JUL-79 10:41 L 15 PAGE 195
(CK.NUM - CHECK NUMBER (OCTAL))

SEQ 0193

```
10180 055652 005015 055103 045122 TITLE: .ASCII <CR><LF>/CZRJD-D/<CR><LF>
10181 055660 026504 006504 012
10182 055665 115 052114 042055 .ASCIZ @MLT-DR LGC@<CR><LF><LF>
10183 055672 020122 043514 006503
10184 055700 005012 000
10185 055703 015 052012 020117 LOADRV: .ASCII <CR><LF>/TO TEST DRIVE 0, REPLACE THE 'XXDP' PACK ON DRIVE 0/<CR><LF>
10186 055710 042524 052123 042040
10187 055716 044522 042526 030040
10188 055724 020054 042522 046120
10189 055732 041501 020105 044124
10190 055740 020105 054047 042130
10191 055746 023520 050040 041501
10192 055754 020113 047117 042040
10193 055762 044522 042526 030040
10194 055770 005015
10195 055772 044527 044124 040440 .ASCII /WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,/<CR><LF>
10196 056000 047516 044124 051105
10197 056006 050040 041501 026113
10198 056014 041440 042514 051101
10199 056022 046440 046505 051117
10200 056030 020131 047514 040503
10201 056036 044524 047117 032040
10202 056044 026060 005015
10203 056050 047101 020104 042522 .ASCIZ /AND RESTART THE PROGRAM/<CR><LF>
10204 056056 052123 051101 020124
10205 056064 044124 020105 051120
10206 056072 043517 040522 006515
10207 056100 000012
10208 056102 005015 054523 052123 NOLOAD: .ASCIZ <CR><LF>/SYSTEM HAS 16K MEMORY, 'XXDP' LOADER WILL BE OVERWRITTEN/<CR><L
10209 056110 046505 044040 051501
10210 056116 030440 045466 046440
10211 056124 046505 051117 026131
10212 056132 023440 054130 050104
10213 056140 020047 047514 042101
10214 056146 051105 053440 046111
10215 056154 020114 042502 047440
10216 056162 042526 053522 044522
10217 056170 052124 047105 005015
10218 056176 000
10219
10220 000001 .END
```


CZRJDDO, RP04/5/6 MLT-DR LGC
CZRJDD.P11 06-JUN-79 08:32

MACY11 30A(1052) 03-JUL-79 10:41 PAGE 199
CROSS REFERENCE TABLE -- USER SYMBOLS

CMCNT	001312	2340#	4194*	4195*	4204	4206	4209*	4266													
CMCYL	001314	2341#	4196*	4197*	4212	4278*															
CMDAT	013524	4211	4228	4232#																	
CMHED	013430	4212#																			
CMPAR	013242	3483	4185#																		
CMPARD	013260	3740	4189#																		
CMPLMT	001414	2392#	4141	4199	9847																
CMPRES	017616	3284	3339	3371	3396	3419	3436	3465	4994#	4997											
CMprt	013770	4230	4244	4254	4264	4285#															
CMPRX	013762	4225	4227	4267	4280#																
CMSEC	001316	2342#	4198*	4215	4270*	4271	4273*														
CMSTR	013352	4201#	4269	4272	4276	4279															
CMTRK	001317	2343#	4274*	4275	4277*																
COLON	053477	5804	5809	9808#																	
COMTBL	001760	2508#	4973	6105																	
CR =	000015	1859#	2297#	6854	7015	7025	9443	9457	9462	9620	9624	9628	9635	9644							
		9650	9662	9667	9672	9680	9689	9728	9734	9747	9752	9761	9764	9768							
		9772	9774	9784	9788	9796	9809	9811	9814	9818	9820	9823	9826	9832							
		10009	10180	10182	10185	10195	10203	10208													
		1860#	6986	7025																	
CRLF =	000200	2122#	2140#																		
CSF =	000002	2124#	2142#																		
CSU =	000010	3891*	3892	3895*	3929*	5320	5374	5377	5380	5383	9120	9973#									
CYLDER	054746	2356#	4852*	4855*	4861	4865	4867	6145*	6148*	6151	6159	6160	6250								
CYLMT	001350	5895	6067#																		
DATAPK	025226	2680	2683#																		
DATAO	003026	2665	2666	2667	2668	2669	2670	2671	2672	2673	2674	2675	2676	2677							
DATA1	003066	2678	2679	2681	2700#																
		2311#	6030	6033	9990*	9991*	9992*	9993*	9994*	9995*	9996*	9997*									
DATE	001220	6032	9809#																		
DATEIS	053501	2058#																			
DCK =	100000	3640	3648#																		
DCKER	007656	3673#	3964																		
DCKER1	010010	2181#	2190#																		
DCL =	000100	2187#																			
DCU =	000001	1866#	2283	3056																	
DDISP =	177570	5885	5984#																		
DEASGN	024650	3280	9764#																		
DEASSG	053140	2029#																			
DE1 =	000040	2025#																			
DFF20 =	000002	2975	9411#																		
DH1	047626	5168	9437#																		
DH14	050004	5177	9450#																		
DH15	050110	5180	9462#																		
DH16	050207	2982	3003	9413#																	
DH2	047633	2989	9422#																		
DH3	047710	2996	9427#																		
DH4	047736	3010	9434#																		
DH6	047775	2026#																			
DIGB =	000004	2283#	3056*	3064*	6884*																
DISPLA	001142	3520	3554	3574	3585	3655	3672	3707	3744	3754	3776	3816	3840	3860							
DISPLY=	104414	3900	3926	3941	3963	3969	3987	3997	4006	4021	4037	4057	4068	4087							
		4105	4110	4111	4112	4115	4118	4121	4124	4127	4130	4133	4136	4157							
		4175	4309	4314	4315	4316	4329	4332	4335	4344	4347	4404	4409	4441							
		4444	4447	4450	4453	4456	4459	4462	4464	4465	4488	4494	4499	4501							
		4608	5112	5115	5132	5158	5160	5165	5166	5167	5168	5169	5172	5177							

LF = 000012	1858#	2298#	6854	7019	7025	9443	9457	9462	9620	9624	9628	9635	9644
	9650	9662	9667	9672	9680	9689	9728	9734	9747	9752	9761	9764	9768
	9772	9774	9784	9788	9796	9809	9811	9814	9818	9820	9823	9826	9832
	10009	10180	10182	10185	10195	10203	10208						
LIMIT 001310	2339#	4141*	4199*	4200*	4319	4321*							
LINA3 050626	5319	9525#											
LINB3 050700	5241	9534#											
LINB5 051055	5356	9557#											
LINB6 051170	5405	5418	9572#										
LINCA3 050657	5278	9530#											
LINC6 051223	5412	9577#											
LINDA3 050670	5274	9532#											
LINDEC 022362	4346	5171	5255	5260	5265	5287	5290	5292	5295	5299	5303	5311	5314
	5322	5326	5340	5345	5366	5446	5461	5494	5497	5534#			
LIND5 051040	5352	9554#											
LINEN3 050611	5312	9522#											
LINE05 051156	5396	9570#											
LINEP5 051145	5392	9568#											
LINE1 020266	3519	3553	3573	3584	3654	3671	3753	3770	3815	3839	3859	3899	3925
	3940	3962	3968	3986	3996	4005	4020	4036	4056	4067	4086	4104	4156
	4174	4308	5107#										
LINE2 020332	3521	3555	3575	3586	3656	3673	3756	3778	3817	3841	3861	3901	3927
	3942	3970	3988	3998	4007	4022	4038	4058	4069	4088	4106	4158	4176
	4310	5128#	5509										
LINE2A 020502	5140	5145	5163#										
LINE2B 020520	5164	5167#											
LINE3 020740	3522	3556	3576	3587	3674	3757	3779	3818	3842	3862	3902	3943	4008
	4023	4039	4089	4159	5214#								
LINE3A 020746	4070	4107	4311	5220#									
LINE3B 020754	4059	5226#											
LINE3C 020766	3928	5233#											
LINE3D 020776	4177	5193	5239#										
LINE3E 021044	3971	5253#											
LINE3F 021132	3989	5272#											
LINE4 021414	3523	3557	3577	3588	3675	3758	3780	3819	3863	3903	3944	3972	4009
	4024	4040	4071	4108	4160	4178	4312	5194	5333#				
LINE5 021504	3759	3781	3822	3866	3906	4012	4027	5352#					
LINE5A 021604	3867	3907	3930	4013	4028	5373#							
LINE5B 021676	3731	5392#											
LINE6 021740	3680	5405#											
LINE6A 021752	3737	5412#											
LINE6B 021760	3730	5418#											
LINE6C 021766	3661	3764	3790	3799	3829	3874	3914	3950	3978	4048	4078	4094	5424#
LINE6D 021774	3663	3735	3766	3832	3877	3917	3953	3981	4050	4081	4096	5430#	
LINE7 022050	3525	3559	3579	3590	3664	3727	3745	3804	3807	3830	3833	3844	3875
	3878	3915	3918	3951	3954	3979	3991	4000	4015	4030	4051	4079	4097
	4162	4180	4349	4605	5454#								
LINE7A 022176	3933	4062	5482#										
LINE8 022316	3724	3801	4604	5508#									
LING6 051252	5424	9581#											
LINKDV 027152	4385	4477	4895	5363	6443	6453#							
LINM3 050511	5214	9509#											
LINM4 050754	5335	9544#											
LINN3 050527	5220	9512#											
LINOCT 022330	4114	4117	4120	4123	4126	4129	4132	4135	4328	4331	4334	4443	4446
	4449	4455	4458	4461	4491	4496	5203	5243	5246	5276	5280	5337	5355

CZRJDDO, RP04/5/6 MLT-DR LGC
CZRJDD.P11 06-JUN-79 08:32

J 1
MACY11 30A(1052) 03-JUL-79 10:41 PAGE 220
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0216

.\$ASTA	1#		
.\$CATC	1#	1817#	2228
.\$CMTA	1#	1817#	2255
.\$DB2D	1#	1817#	7364
.\$DB2O	1#	1817#	7427
.\$DIV	1#		
.\$EOP	1#		
.\$ERRO	1#	1817#	6862
.\$ERRT	1#	1817#	6907
.\$MULT	1#		
.\$POWE	1#		
.\$RAND	1#	1817#	7281
.\$RDDE	1#		
.\$RDOC	1#		
.\$READ	1#	1817#	7172
.\$R2AZ	1#		
.\$SAVE	1#	1817#	7318
.\$SB2D	1#		
.\$SB2O	1#		
.\$SCOP	1#		
.\$SIZE	1#	1817#	10018
.\$SUPR	1#		
.\$TRAP	1#	1817#	7467
.\$TYPB	1#		
.\$TYPD	1#	1817#	7104
.\$TYPE	1#	1817#	6955
.\$TYPO	1#	1817#	7026
.\$4OCA	1#		
.1170	1#		

. ABS. 056177 000

ERRORS DETECTED: 0

SAIL:CZRJDD.BIN,CZRJDD.LST/CRF/SOL/NL:TOC:MD:MC:CND/LI:ME=CZRJDD.SML,CZRJDD.011,CZRJDD.P11
RUN-TIME: 73 104 8 SECONDS
RUN-TIME RATIO: 452/186=2.4
CORE USED: 50K (99 PAGES)