

801

EJF:DCM9ADSEJ

00010000

770712

POP10 411

37HOR1DZRKPGSEQ

00010000

770712

.REM %

IDENTIFICATION  
-----

PRODUCT CODE: MAINDEC-11-DZRHK-G-D  
PRODUCT NAME: RK11/RK05 PERFORMANCE EXERCISER  
DATE: APRIL, 1977  
MAINTAINER: DIAGNOSTIC GROUP  
AUTHOR: JIM KAPADIA  
REVISIONS: TOM SAWYER, GEORGE GALLANT, CHUCK HESS

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 MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE IN A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1974,1977 BY DIGITAL EQUIPMENT CORPORATION

TABLE OF CONTENTS

- 1.0 ABSTRACT
- 2.0 REQUIREMENTS
  - 2.1 EQUIPMENT
  - 2.2 PRELIMINARY PROGRAMS
  - 2.3 EXECUTION TIME
- 3.0 STARTING ADDRESSES
- 4.0 PROGRAM CONTROL MODES
  - 4.1 PAPER TAPE LOADING
  - 4.2 RKDP DUMP MODE
  - 4.3 RKDP CHAIN MODE
  - 4.4 ACT11
- 5.0 DRIVE SELECTION
- 6.0 SWITCH OPTIONS
- 7.0 PROGRAM STRUCTURE AND DESCRIPTION
  - 7.1 NON-EXERCISER TESTS
  - 7.2 EXERCISER PROGRAM
- 8.0 LOOPING CAPABILITIES
- 9.0 TRANSFER DATA LOGGING
- 10.0 ERROR LOGGING
- 11.0 ERROR REPORTING AND RECOVERY
- 12.0 SUBROUTINES AND HANDLERS

## 1.0 ABSTRACT

THE RK11/RK05 PERFORMANCE EXERCISER IS A HIGH LEVEL EXERCISER PROGRAM AIMED AT SIMULATING A RK11/RK05 SYSTEM ENVIRONMENT AND CHECKING FOR ERRORS THAT ARISE IN SUCH AN ENVIRONMENT (INTERACTION, POLLING, ETC). IT ALSO PROVIDES A MEANS OF EVALUATING A SYSTEM THROUGH ITS ERROR LOGGING AND DATA-TRANSFER LOGGING FACILITIES.

AT THE BEGINNING OF THE PROGRAM THERE IS A SERIES OF TESTS SPECIFICALLY AIMED AT DETECTING AND ANALYZING FAILURES ASSOCIATED WITH BOUNDARY CONDITION TRANSFERS.

THE LATTER PART (AND THE MORE SIGNIFICANT ONE) CONSISTS OF THE EXERCISER.

## 2.0 REQUIREMENTS

### 2.1 EQUIPMENT

- A. PDP11 WITH CONSOLE TELTYPE
- B. 8K OF MEMORY - 12K FOR CHAIN MODE
- C. RK11 OR RKV11 CONTROLLER
- D. 1-8 RK05 OR RK05F DRIVES (DRIVE TYPES MAY BE MIXED)

### 2.2 PRELIMINARY PROGRAMS

SINCE THIS IS A HIGH-LEVEL EXERCISER PROGRAM THE CONTROLLER AND THE DRIVE SHOULD BE FREE OF BASIC FAULTS. IT IS POSSIBLE TO HANG THE PROGRAM IF THE HEAD POSITIONING LOGIC IS FAULTY ON DUAL DENSITY DRIVES. THUS THE FOLLOWING PROGRAMS SHOULD BE RUN BEFORE ATTEMPTING TO USE THIS PROGRAM.

- A. RK11 BASIC LOGIC TESTS (I AND II)
- B. RK11/RK05 DYNAMIC TESTS
- C. RK05 UTILITY PACKAGE (IF NEEDED)

### 2.3 EXECUTION TIME

THIS VARIES FROM 30 TO 90 MINUTES FOR A PASS. IT SHOULD BE NOTED THAT THIS IS AN EXERCISER LEVEL PROGRAM AND SHOULD BE PREFERABLY RUN FOR A LONG PERIOD OF TIME.

NOTE: THE FIRST PASS IS A SHORT (5-20 MINUTE) PASS TO SERVE AS A QUICK VERIFY OF THE RK SUBSYSTEM.

## 3.0 STARTING ADDRESS

200 - ALL SWITCHES DOWN. SEE SEC. 6.0 FOR SWITCHES.

210 - RESTART ADDRESS. THE RESTART ADDRESS PROVIDES THE USER WITH AN ABILITY TO GO STRAIGHT TO EXERCISER PART OF THE PROGRAM (SKIPPING TESTS 1-7). THERE IS A SWITCH OPTION (SW 4) WHICH ALLOWS THE USER TO INHIBIT THE REWRITE OF RANDOM PATTERNS ON ALL DRIVES, ON RESTART. SEE SEC- 6.9.

4.0 PROGRAM CONTROL MODES AND OPERATOR ACTION

PAPER TAPE LOADING  
RKDP DUMP MODE  
RKDP CHAIN MODE  
ACT11

4.1 PAPER TAPE LOADING

4.1.1 LOAD PROGRAM INTO MEMORY USING STANDARD PROCEDURE FOR ABSOLUTE TAPES.

4.1.2 MAKE SURE THAT THE DRIVES TO BE CHECKED ARE LOADED WITH DISKS AND ARE IN 'RUN'. 'WRT ENABLE' THEM. CHECK THAT 'WRT PROT' LIGHT ON THESE DRIVES IS OFF. PUT DRIVES THAT ARE NOT TO BE TESTED ON 'LOAD'.

4.1.3 LOAD ADDRESS 200

4.1.4 SET SWITCHES IF DESIRED (SEE SEC 6.0) AND PRESS START

4.1.5 THE PROGRAM IDENTIFIES ITSELF

MAINDEC-11-DZRKH-G

RK11/RK05 PERFORMANCE EXERCISER

THEN IT PROCEEDS TO TEST THE DRIVES.

4.2 RKDP DUMP MODE

4.2.1 THE PROGRAM IS LOADED BY THE RKDP MONITOR.

4.2.2 SET SA=200. SELECT ANY SWITCHES YOU WANT AND PRESS START.

4.2.3 THE PROGRAM IDENTIFIES ITSELF AND PRINTS OUT:

'TO TEST DRIVE 'N' HALT PROGRAM, REMOVE RKDP PACK AND REPLACE IT WITH A WORK PACK, CLEAR LOCATION 40, AND RESTART PROGRAM'

IN RESPONSE TO THIS MESSAGE, PERFORM THE ACTIONS REQUESTED IF THE DRIVE ON WHICH THE RKDP PACK IS MOUNTED IS TO BE TESTED.

4.3 RKDP CHAIN MODE

THE PROGRAM IS CHAIN LOADED FROM RKDP PACK ON DRIVE 'N'. AFTER IDENTIFYING ITSELF, THE FOLLOWING MESSAGE APPEARS:

'DRIVE 'N' NOT TESTED'

DRIVE 'N' WILL NOT BE TESTED SINCE THE RKDP PACK IS ON THAT DRIVE.

#### 4.4 ACT11 MODE

THE PROGRAM IS LOADED BY THE ACT11 MONITOR. AFTER IDENTIFYING ITSELF, ASCERTAINS THE NUMBER OF DRIVES PRESENT AND PROCEEDS TO TEST EACH OF THEM AS BEFORE.

#### 5.0 DRIVE SELECTION

PUT ALL THE DRIVES THAT ARE TO BE EXERCISED AND TESTED ON 'RUN', WRITE ENABLE THEM. MAKE SURE THAT THE 'WRT PROT' IS OFF. THE PROGRAM RECOGNIZES THAT THESE DRIVES ARE ON LINE AND PROCEEDS TO TEST THEM. RK05F DRIVES WILL HAVE THE LETTER F TYPED AFTER THE DRIVE NUMBER.

IF A FATAL ERROR OCCURS ON A DRIVE WHILE THE PROGRAM IS RUNNING THE DRIVE IS AUTOMATICALLY DESELECTED ('DSELCT') AND DROPPED FROM THE DRIVE SELECTION LIST.

#### 6.0 SWITCH OPTIONS

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' WHENEVER THE PROGRAM ENTERS THE SCOPE ROUTINE OR BEGINS A NEW TEST. 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.

SW<15>=1	HALT ON ERROR
SW<13>=1	INHIBIT ERROR PRINTOUTS
SW<12>=1	TYPE OUT THE ERROR HISTORY
SW<11>=1	DUMP OUT ALL RK11 REGISTERS
SW<10>=1	RING BELL ON ERROR
SW<09>=1	LOOP ON SPECIFIC ERROR
SW<08>=1	DUMP OUT TRANSFER DATA AND ERROR STATISTICS
SW<06>=1	SELECT BUS ADDRESS LIMITS FOR DATA TRANSFERS
SW<05>=1	HALT BEFORE DOING THE NEXT SET OF COMMANDS
SW<04>=1	DO NOT REWRITE THE DISKS ON 210 RETSART
SW<03>=1	TYPE OUT ELAPSED TIME AT ERROR
SW<02>=1	DROP DRIVE AFTER MAXIMUM ERRORS

SW<01>=1 TYPE SERIAL NUMBER OF ERRORING DRIVE  
SW<00>=1 TYPE ONLY ELAPSED TIME IF SW<08> AND SW<03> = 1

6.1 SW<15>

THE PROGRAM HALTS ON ENCOUNTERING AN ERROR, AFTER TYPING OUT THE ERROR MESSAGE AND PERTINENT INFORMATION. PRESSING "CONTINUE" RESTORES NORMAL OPERATION OF THE PROGRAM.

6.2 SW<13>

THIS SWITCH INHIBITS ALL ERROR MESSAGES. NORMALLY USED WHEN LOOPING ON ERROR (SW 9).

6.3 SW<12>

IF THIS SWITCH IS SET WHEN AN ERROR OCCURS, INFORMATION ABOUT THE HISTORY OF THAT ERROR IS TYPED OUT.

THE FUNCTION THAT WAS BEING PERFORMED ON THE RK11 IS TYPED OUT. THE FUNCTION COULD BE EITHER A READ, WRITE, WRITE CHECK, READ CHECK. BESIDES THESE NORMAL FUNCTIONS, IT COULD BE A CONTROL RESET, DRIVE RESET OR POSITIONING OF THE HEADS (SEEKING). FOR THE FOUR FUNCTIONS THE INITIAL DISK ADDRESS, BUS ADDRESS AND WORD COUNT (2'S COMPLEMENT) ARE ALSO GIVEN. FOR DRIVE RESET AND POSITIONING THE DRIVE NUMBER OR WHICH THE OPERATION WAS BEING PERFORMED IS GIVEN.

SIMILAR INFORMATION IS TYPED OUT ABOUT THE FUNCTION THAT WAS DONE JUST BEFORE THE ONE GIVING THE ERROR.

6.4 SW<11>

IF THIS SWITCH IS SET WHEN AN ERROR OCCURS, THE CONTENTS OF ALL RK11 REGISTERS ARE TYPED OUT.

6.5 SW<09>

THIS SWITCH PROVIDES THE TIGHTEST POSSIBLE SCOPE LOOP. LOOPING IS DONE WHEN AN ERROR OCCURS. NOTE THAT THERE ARE TWO CLASSES OF ERRORS AND HENCE TWO CLASSES OF ERROR LOOPS, REFER TO SEC 8.0 FOR THE DIFFERNECE IN THE ERROR LOOPS PROVIDED BY SW 9.

6.6 SW<08>

WHEN THIS SWITCH IS SET, THE ERROR AND TRANSFER DATA STATISTICS WHICH HAVE BEEN COLLECTED UNTIL THAT TIME, ARE TYPED OUT.

THE TRANSFER DATA STATISTICS GIVE THE NUMBER OF WORDS WRITTEN AND READ ON EACH DRIVE THAT IS PRESENT. IT SHOULD BE NOTED THAT READ CHECK AND WRITE CHECK ARE CONSIDERED TO BE ESSENTIALLY READ OPERATIONS.

THE ERROR STATISTICS GIVE THE NUMBER OF ERRORS THAT HAVE OCCURRED

(IF ANY) IN THE FOLLOWING CATEGORIES, ON THE DRIVES THAT ARE PRESENT:

CHECK SUM ERROR  
WRITE CHECK ERROR  
DATA COMPARISON ERROR  
HARD ERROR  
SEEK ERROR  
SEEK INCOMPLETE

ABORTS - WHEN AN ERROR OCCURS THE FUNCTION IS RETRIED TWICE. IF STILL THE ERROR PERSISTS THE FUNCTION IS ABORTED AND THE ABORT COUNT IS INCREMENTED FOR THAT DRIVE.

6.7 SW<06>

THIS SWITCH ENABLES THE USER TO SELECT THE LIMITS OF THE MEMORY BUS ADDRESSES BETWEEN WHICH THE DATA TRANSFERS WILL BE DONE. NORMALLY THE TRANSFERS ARE DONE BETWEEN THE LOWER LIMIT (BASEBA) AND THE HIGHER LIMIT (MAXBA). THESE TWO LIMITS ARE NORMALLY SELECTED BY THE PROGRAM AND USE THE MAXIMUM AVAILABLE MEMORY. IF THE USER WANTS TO DO DATA TRANSFERS BETWEEN SELECTED MEMORY ADDRESSES (EX: BETWEEN 12K AND 16K) THEN THIS SWITCH SHOULD BE SET AT THE STARTING OF THE PROGRAM. THE FOLLOWING MESSAGE APPEARS:

TYPE OCTAL BUS ADDRESS FOR DATA XFER, BETWEEN XXXXXX AND YYYYYY

LO LIMIT?  
HI LIMIT?

IN RESPONSE THE USER SHOULD TYPE IN ANY TWO BUS ADDRESSES (OCTAL) BETWEEN XXXXXX AND YYYYYY. IF THE USER TYPES IN ANYTHING OUT OF THE X AND Y RANGE THE QUESTION IS ASKED AGAIN.

THIS SWITCH COULD BE QUITE USEFUL IN DETERMINING WHETHER THE PROBLEM IS WITHIN THE RK11 OR OUTSIDE (IN MEMORY). NORMALLY, IF THE PROBLEM IS WITHIN THE RK11, ERRORS WILL KEEP ON OCCURRING REGARDLESS OF WHERE IN THE MEMORY DATA TRANSFERS ARE TAKING PLACE. ON THE OTHER HAND IF THE PROBLEM IS MEMORY RELATED, THE ERRORS WILL TEND TO DISAPPEAR FOR DATA TRANSFERS TO CERTAIN MEMORY BLOCKS AND WOULD REAPPEAR FOR OTHER ONES.

6.8 SW<05>

THIS SWITCH PROVIDES THE USER A CAPABILITY TO HALT THE PROGRAM AT A KNOWN POINT. THE HALT IS DONE AFTER THE CURRENT SET OF EIGHT COMMANDS IN THE QUEUE HAVE BEEN EXECUTED. THE "HALT" IS LOCATED AT THE BEGINNING OF THE 'GENBRQ' ROUTINE, JUST BEFORE A SET OF 8 NEW COMMANDS IS GENERATED. AFTER THE PROGRAM HALTS, THE EXECUTION CAN BE RESUMED BY PRESSING CONTINUE, OR THE PROGRAM CAN BE STARTED BACK AT 200 OR RESTARTED AT 210.

6.9 SW<04>



THIS SWITCH PROVIDES THE USER WITH AN ABILITY TO SKIP THE TIME CONSUMING REWRITE OF ALL THE DISKS WHEN THE PROGRAM IS RESTARTED AT 210. THIS SWITCH CAN BE USED ONLY WHEN RESTARTING THE PROGRAM AT 210 WITH SW 4 SET. ON RESTARTING THE PROGRAM AT 210, THE INITIAL BOUNDARY CONDITION TESTS (1ST1-1ST7) ARE SKIPPED. IF SWITCH 4 IS SET, THE REWRITE OF ALL THE DISKS (WHICH WOULD HAVE BEEN NORMALLY DONE) IS ALSO SKIPPED. THE USER IS CAUTIONED TO USE THIS SWITCH CAREFULLY. THE DISKS SHOULD HAVE BEEN WRITTEN WITH RANDOM PATTERNS AT LEAST ONCE BEFORE RESTARTING THE PROGRAM AT 210. IT SHOULD BE NOTED THAT TESTS 1-7 WRITE ON CYLINDERS 0,1. ON RESTART, THE STATISTICS COLLECTED SO FAR ARE SAVED.

## 6.10 SW&lt;03&gt;

THIS SWITCH ALLOWS THE TIMEOUT OF THE ELAPSED TIME AT WHICH ERROR OCCURRED. THE TIMING STARTS AT THE BEGINNING OF THE EXERCISER PROGRAM. THIS SWITCH SHOULD NOT BE SET IF KWILL LINE CLOCK IS NOT AVAILABLE ON THE SYSTEM.

## 6.11 SW&lt;02&gt;

THIS SWITCH CAUSES DRIVES WHICH EXCEED A MAXIMUM NUMBER OF ERRORS TO BE DEASSIGNED BY THE PROGRAM. THE PROGRAM CONTINUES TESTING OTHER DRIVES WHICH HAVE NOT ACCUMULATED THE REQUIRED NUMBER OF ERRORS.

## 6.12 SW&lt;01&gt;

IF THIS SWITCH IS SET, THE PROGRAM ALLOWS A SERIAL NUMBER TO BE SPECIFIED FOR EACH DRIVE TESTED. THE SERIAL NUMBER IS TYPED WITH EACH ERROR MESSAGE FOR THAT PARTICULAR DRIVE.

## 6.13 SW&lt;00&gt;

IF SW<08> AND SW<03> ARE SET, SETTING THIS SWITCH TYPES OUT THE ELAPSED TIME FROM THE START OF THE PROGRAM.

## 7.0 EXERCISER PROGRAM

THE EXERCISER PROGRAM ATTEMPTS TO SIMULATE A DISK OPERATING SYSTEM ENVIRONMENT BY DOING RANDOM EVENTS (FUNCTIONS) USING RANDOMLY SELECTED PARAMETERS (DISK ADDRESS, BUS ADDRESS, WORD COUNT ETC). AN ATTEMPT IS MADE TO DETECT INTER-ACTION PROBLEMS, OVERLAPPING SEEK PROBLEMS, ETC. FOR EXAMPLE, OVER 500 MILLION BITS ARE TRANSFERRED PER HOUR ON A TYPICAL RK11/RK05 SYSTEM (BASED ON 2 DRIVES, PDP11/50, 28K SYSTEM).

EIGHT JOBS OR COMMANDS ARE GENERATED AT A TIME (GENBRQ) AND PUT IN A QUEUE TO BE PROCESSED. THE ALGORITHM WORKS AS FOLLOWS. COMMANDS IN THE QUEUE ARE PREPOSITIONED (HEADS) BY PREFORMING OVERLAPPING SEEKS. WHILE SOME OF THE DRIVES ARE BEING POSITIONED, THE LAST AVAILABLE (AND EXECUTABLE) COMMAND IS PERFORMED. THUS WHILE SOME DRIVES ARE BUSY POSITIONING THEIR HEADS, SOME DRIVE IS PERFORMING A FUNCTION (DATA TRANSFER, ETC). AS SOON AS THE CONTROLLER IS FREE, A CHECK IS MADE TO SEE IF THERE IS ANY DRIVE WHICH HAS ALREADY POSITIONED ITS HEAD. IF ONE IS FOUND THE COMMAND IS EXECUTED ON THAT DRIVE AND THE CONTROLLER AGAIN BECOMES BUSY. IF NO POSITIONED COMMAND IS FOUND, A CHECK IS MADE TO SEE IF THERE IS A COMMAND THAT IS TO BE POSITIONED. IF YES, IT IS POSITIONED AND THE LAST AVAILABLE COMMAND IS EXECUTED. IF IT IS FOUND THAT NO DRIVE NEEDS TO BE POSITIONED (THIS COULD HAPPEN IF THERE IS ONLY ONE COMMAND LEFT IN THE QUEUE OR THE REMAINING COMMANDS IN THE QUEUE ARE TO BE PERFORMED ON THE SAME DRIVE), THEN THE COMMANDS IS/ ARE EXECUTED.

THE ABOVE ALGORITHM HELPS SIMULATE A REAL ENVIRONMENT, AT THE SAME TIME MAXIMISING THE RATE OF DATA TRANSFERS. THE EXERCISER PROGRAM GIVES AN ELABORATE ERROR DETECTION CAPABILITY. THE STATE OF THE PROGRAM IS CONTINUOUSLY TRACKED BY SOFTWARE KEYS, FLAGS, ETC. THESE FLAGS AND KEYS HAVE BEEN EXPLAINED IN DETAIL AT THE BEGINING OF THE LISTINGS, WHERE THEY ARE DEFINED. ON DUAL DENSITY DRIVES, ONLY ONE LOGICAL DRIVE IS SELECTED DURING EACH QUEUE BUILD. THIS INSURES THAT OVERLAPPED SEEKS WILL NOT INTEFER WITH THE HEAD POSITIONING LOGIC.

THE PARAMETERS USED FOR DOING THE COMMANDS ARE SELECTED RANDOMLY USING A RANDOM GENERATOR. THE FUNCTION TO BE PERFORMED IS SELECTED RANDOMLY FROM ONE OF THE FOUR: WRITE, READ, WRITE CHECK, OR READ CHECK. THE DRIVE NUMBER IS SELECTED FROM THE AVAILABLE DRIVES. THE DISK ADDRESS IS SELECTED OVER THE ENTIRE RANGE AND THE WORD COUNT AND BUS ADDRESS ARE SELECTED RANDOMLY IN SUCH A WAY THAT A NON-EXISTENT MEMORY ERROR OR OVERRUN CONDITION DOES NOT OCCUR.

RANDOM DATA BLOCKS ARE WRITTEN ON THE DISK. THE FIRST WORD OF EACH SECTOR BLOCK IS A NUMBER (2'S COMPLEMENT) INDICATING THE TOTAL NUMBER OF WORDS WRITTEN IN THAT SECTOR. THE REST OF THE WORDS IN THE BLOCK ARE GENERATED USING THE DISK ADDRESS (OF THAT SECTOR) AS THE RANDOM SEED NUMBER.

#### 8.0 LOOPING CAPABILITIES:

SWITCH 9 GIVES LOOPING CAPABILITIES, ON ERROR. THERE ARE TWO CLASSES OF ERRORS:

- A. ERRORS OCCURING IN THE NON-EXERCISER PART OF THE PROGRAM (ERROR NUMBERS UNDER 100 IN THE ERROR ITEMS TABLE)
- B. ERRORS OCCURING IN THE EXERCISER PART OF THE PROGRAM (ERROR NUMBERS STARTING FROM 100 AND UP IN THE ERROR ITEMS TABLE)
- C. NON-EXERCISER SCOPE LOOPS: IN THIS CASE, THE PROGRAM LOOPS ON A SPECIFIC ERROR GIVING A NARROW SCOPE LOOP. THIS SCOPE LOOP IS SIMILAR TO THE ONE PROVIDED IN THE RK11 BASIC LOGIC TEST AND DYNAMIC TEST, WHICH THE USER MIGHT BE FAMILIAR WITH.
- D. EXERCISER SCOPE LOOPS: WHEN AN ERROR OCCURS (AFTER TYPING OUT THE ERROR MESSAGE) CONTROL IS TRANSFERRED TO THE BEGINNING OF THE COMMAND-QUEUE. THE COMMANDS FROM THE FIRST COMMAND ONWARDS, ARE EXECUTED AGAIN TILL THE POINT OF ERROR. THIS LOOPING PROVIDES THE USER WITH A CAPABILITY TO RECREATE A SET OF EVENTS THAT LED TO THE ERROR.

#### 9.0 TRANSFER DATA LOGGING

IN THIS PROGRAM, WHENEVER A DATA TRANSFER TAKES PLACE IT IS LOGGED WHETHER IT IS READ, READ CHECK, WRITE OR WRITE CHECK. SEPERATE COUNTS ARE KEPT FOR DATA TRANSFERS TAKING PLACE ON EACH DRIVE IN THE SYSTEM. AT ANY GIVEN TIME THE USER CAN GET THESE TRANSFER STATISTICS BY SETTING SWITCH 8 TO 1 (SEE SEC.6.6). THIS IS HELPFUL FOR EVALUATING A SYSTEM.

#### 10.0 ERROR LOGGING

THROUGHOUT THE EXERCISER PROGRAM, WHEN AN ERROR OCCURS IT IS LOGGED. THE FOLLOWING CLASSES OF ERRORS ARE LOGGED FOR EACH DRIVE IN THE SYSTEM:

- CHECK SUM ERROR
- WRITE CHECK ERROR
- DATA COMPARISON ERROR
- HARD ERRORS
- SEEK ERROR
- SEEK INCOMPLETE ERROR
- ABORTS

THE ERROR STATISTICS CAN BE OBTAINED BY PUTTING SWITCH 8 TO 1. THE ERROR STATISTICS CAN BE USED IN CONJUNCTION WITH DATA TRANSFER STATISTICS TO GIVE AN IDEA OF THE SYSTEM PERFORMANCE (NUMBER OF WORDS TRANSFERRED PER ERROR, CSE FREQUENCY, RECOVERABLE VERSUS NON-RECOVERABLE ERRORS ETC.).

#### 11.0 ERROR REPORTING AND RECOVERY

WHENEVER AN ERROR OCCURS IT IS REPORTED ALONG WITH RELEVANT INFORMATION. THE RK11 REGISTERS REPORTED IN THE ERROR MESSAGES REPRESENT THE CONTENTS AT THE TIME OF ERROR. EACH ERROR MESSAGE CONTAINS A 'PC' NUMBER, THIS IS THE PC LOCATION IN THE PROGRAM WHERE THE ERROR CALL IS LOCATED. THE USER IS ADVISED TO REFERENCE THIS LOCATION IN THE LISTINGS, IN CASE MORE INFORMATION ABOUT THE ERROR IS DESIRED.

SOME (SYSTEM) ERRORS REFER TO SOFTWARE FLAGS AND KEYS WHICH ARE USED TO MONITOR THE ONGOING ACTIVITIES ON THE SYSTEM. THESE FLAGS ARE EXPLAINED AT THE BEGINING OF THE LISTINGS AND SOULD BE REFERRED TO, IF THE NEED ARISES.

IF A FATAL ERROR CONDITION IS DETECTED (LIKE DRIVE UNSAFE, WRITE PROTECT SET, DRIVE READY CLEAR, ETC.) THE DRIVE IS REMOVED FROM THE DRIVE SELECTION TABLE AND DROPPED FROM FURTHER TESTING. A MESSAGE IS GIVEN INDICATING DROPPING OF THAT DRIVE. FOR FURTHER INFORMATION, REFER TO THE 'CHKDRV' AND 'DSELC' ROUTINES IN THE LISTINGS.

RECOVERABLE ERRORS ARE RETRIED THREE TIMES. IF THE ERROR CONDITION FAILS TO CORRECT OR A IF A DIFFERENT ERROR OCCURS THE FUNCTION IS ABORTED. MESSAGES ARE PRINTED ONLY ONCE FOR EACH ERROR. AFTER EIGHT ABORTS ARE RECORDED ON A DRIVE THE DRIVE IS DROPPED. DUAL DENSITY DRIVES ARE ALWAYS DROPPED IN PAIRS.

## 12.0 SUBROUTINES AND HANDLERS

THERE ARE TWO WAYS IN WHICH MOST OF THE SUBROUTINES USED IN THIS PROGRAM ARE CALLED:

## 1. THROUGH THE NORMAL JSR CALL

JSR REG,SUBROUTINE

2. THROUGH THE 'TRAP' INSTRUCTION. THE TRAP INSTRUCTION WITH ITS LOWER BYTE ENCODED SERVES AS A CALL FOR SOME ROUTINES. WHEN THE 'TRAP' IS EXECUTED A TRAP OCCURS TO THE TRAP VECTOR AND THE TRAP DECODER IS ENTERED. THE TRAP DECODER (\$STRAP) WILL PICK UP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION AND USE IT TO INDEX THROUGH THE TRAP TABLE (\$STRAPAD) FOR THE STARTING ADDRESS OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL GO TO THE DESIRED ROUTINE.

## 3. \$SCOPE - THE SCOPE HANDLER

THE SCOPE HANDLER IS ENTERED THROUGH THE EXECUTION OF THE 'IOT' INSTRUCTION. IT KEEPS TRACK OF VARIOUS POINTERS, FLAGS AND DECIDES IF LOOPING IS TO BE DONE ON ERROR (SW 9). IT SHOULD BE NOTED THAT THIS HANDLER IS USED MOSTLY IN THE NON-EXERCISER PART OF THE PROGRAM.

## 4. \$ERROR - ERROR HANDLER ROUTINE

THE ERROR HANDLER IS ENTERED THROUGH THE EXECUTION OF THE 'EMT' INSTRUCTION. THE LOWER BYTE OF THE EMT INSTRUCTION IS ENCODED TO GIVE AN IDENTIFIER TO THE ERROR CALL. THUS 'ERROR 1' IS 104001, ETC. THE ERROR ROUTINE DECIDES IF ANY ACTION IS TO BE TAKEN DEPENDING ON THE SWITCH SETTING (LIKE, HALT ON ERROR, INHIBIT ERROR TYPEOUT, ETC.).

MOST OF THE SUBROUTINES RESIDE IN THE LATTER PART OF THE PROGRAM. THE USER CAN REFER TO THEM THROUGH THE CROSS REFERENCE TABLE AT THE END OF THE LISTINGS OR TABLE OF CONTENTS AT THE BEGINING.

587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620

```

.TITLE MD-11-DZKXHG, RK11-RK05 PERFORMANCE EXERCISER
*COPYRIGHT (C) 1973,1977
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754
*
*PROGRAM BY JIM KAPADIA
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
*
*REVISED BY GEORGE GALLANT, TOM SAWYER - MARCH 1976
*REVISED BY CHUCK HESS - AUGUST 1976
SBTTL OPERATIONAL SWITCH SETTINGS
*
*          SWITCH          USE
*          -----          -----
*          15             HALT ON ERROR
*          14             LOOP ON TEST
*          12             TYPE OUT ERROR HISTORY
*          10             BELL ON ERROR
*          9              LOOP ON ERROR
*          8              TYPE OUT ERROR AND TRANSFER DATA STATISTICS
*          6              SELECT BUS ADDRESS LIMITS FOR DISK DATA TRANSFERS
*          5              HALT BEFORE DOING NEXT SET OF COMMANDS(GENBRQ)
*          4              DO NOT REWRITE THE DISKS ON RESTART AT 2IC
*          3              TYPE OUT ELAPSED TIME AT ERROR
*          2              DROP DRIVE AFTER MAXM ERRORS ON THIS DRIVE
*          1              TYPE SERIAL NUMBER OF ERRORING DRIVE
*          0              IF SW8=1, ONLY TYPE ELAPSED TIME
*          11             DUMP OUT ALL RK11 REGISTERS ON ERROR
*
* YOU ARE ADVISED TO READ THE DOCUMENT FOR THIS PROGRAM.
  
```

621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676

001100  
  
  
  
  
000011  
000012  
000015  
000200  
177776  
  
177774  
177772  
177570  
177570  
  
000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007  
000006  
000007  
  
000000  
000040  
000100  
000140  
000200  
000240  
000300  
000340  
  
100000  
040000  
020000  
010000  
004000  
002000  
001000  
000400  
000200  
000100  
000040  
000020

.SBTTL BASIC DEFINITIONS

.\*INITIAL ADDRESS OF THE STACK POINTER \*\*\* 1100 \*\*\*

STACK= 1100  
 .EQUIV EMT,ERROR ;: BASIC DEFINITION OF ERROR CALL  
 .EQUIV IOT,SCOPE ;: BASIC DEFINITION OF SCOPE CALL

.\*MISCELLANEOUS DEFINITIONS

HT= 11 ;: CODE FOR HORIZONTAL TAB  
 LF= 12 ;: CODE FOR LINE FEED  
 CR= 15 ;: CODE FOR CARRIAGE RETURN  
 CRLF= 200 ;: CODE FOR CARRIAGE RETURN-LINE FEED  
 PS= 177776 ;: PROCESSOR STATUS WORD  
 .EQUIV PS,PSW  
 STKLM= 177774 ;: STACK LIMIT REGISTER  
 PIQ= 177772 ;: PROGRAM INTERRUPT REQUEST REGISTER  
 DSWR= 177570 ;: HARDWARE SWITCH REGISTER  
 DDISP= 177570 ;: HARDWARE DISPLAY REGISTER

.\*GENERAL PURPOSE REGISTER DEFINITIONS

R0= %0 ;: GENERAL REGISTER  
 R1= %1 ;: GENERAL REGISTER  
 R2= %2 ;: GENERAL REGISTER  
 R3= %3 ;: GENERAL REGISTER  
 R4= %4 ;: GENERAL REGISTER  
 R5= %5 ;: GENERAL REGISTER  
 R6= %6 ;: GENERAL REGISTER  
 R7= %7 ;: GENERAL REGISTER  
 SP= %6 ;: STACK POINTER  
 PC= %7 ;: PROGRAM COUNTER

.\*PRIORITY LEVEL DEFINITIONS

PRO= 0 ;: PRIORITY LEVEL 0  
 PR1= 40 ;: PRIORITY LEVEL 1  
 PR2= 100 ;: PRIORITY LEVEL 2  
 PR3= 140 ;: PRIORITY LEVEL 3  
 PR4= 200 ;: PRIORITY LEVEL 4  
 PR5= 240 ;: PRIORITY LEVEL 5  
 PR6= 300 ;: PRIORITY LEVEL 6  
 PR7= 340 ;: PRIORITY LEVEL 7

.\*"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000  
 SW14= 40000  
 SW13= 20000  
 SW12= 10000  
 SW11= 4000  
 SW10= 2000  
 SW09= 1000  
 SW08= 400  
 SW07= 200  
 SW06= 100  
 SW05= 40  
 SW04= 20

677 000010  
678 000004  
679 000002  
680 000001  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693 100000  
694 040000  
695 020000  
696 010000  
697 004000  
698 002000  
699 001000  
700 000400  
701 000200  
702 000100  
703 000040  
704 000020  
705 000010  
706 000004  
707 000002  
708 000001  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721 000004  
722 000010  
723 000014  
724 000014  
725 000014  
726 000020  
727 000024  
728 000030  
729 000034  
730 000060  
731 000064  
732 000240

SW03= 10  
SW02= 4  
SW01= 2  
SW00= 1  
.EQUIV SW09,SW9  
.EQUIV SW08,SW8  
.EQUIV SW07,SW7  
.EQUIV SW06,SW6  
.EQUIV SW05,SW5  
.EQUIV SW04,SW4  
.EQUIV SW03,SW3  
.EQUIV SW02,SW2  
.EQUIV SW01,SW1  
.EQUIV SW00,SW0

.\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000  
BIT14= 40000  
BIT13= 20000  
BIT12= 10000  
BIT11= 4000  
BIT10= 2000  
BIT09= 1000  
BIT08= 400  
BIT07= 200  
BIT06= 100  
BIT05= 40  
BIT04= 20  
BIT03= 10  
BIT02= 4  
BIT01= 2  
BIT00= 1  
.EQUIV BIT09,BIT9  
.EQUIV BIT08,BIT8  
.EQUIV BIT07,BIT7  
.EQUIV BIT06,BIT6  
.EQUIV BIT05,BIT5  
.EQUIV BIT04,BIT4  
.EQUIV BIT03,BIT3  
.EQUIV BIT02,BIT2  
.EQUIV BIT01,BIT1  
.EQUIV BIT00,BIT0

.\*BASIC "CPU" TRAP VECTOR ADDRESSES

ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS  
RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS  
TBITVEC= 14 ;: "T" BIT  
TRTVEC= 14 ;: TRACE TRAP  
BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)  
IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) \*\*SCOPE\*\*  
PWRVEC= 24 ;: POWER FAIL  
EMTVEC= 30 ;: EMULATOR TRAP (EMT) \*\*ERROR\*\*  
TRAPVEC= 34 ;: "TRAP" TRAP  
TKVEC= 60 ;: TTY KEYBOARD VECTOR  
TPVEC= 64 ;: TTY PRINTER VECTOR  
PIRQVEC= 240 ;: PROGRAM INTERRUPT REQUEST VECTOR



```

733          000100          KWLVEC=100          ;KWL11 CLOCK VECTOR
734
735          .EQUIV BIT15,ERR
736          .EQUIV BIT14,HE
737          .EQUIV BIT13,SCP
738
739          .EQUIV BIT12,DPL
740          .EQUIV BIT10,DRU
741          .EQUIV BIT09,SIN
742          .EQUIV BIT07,DRY
743          .EQUIV BIT06,RWS
744          .EQUIV BIT05,WPS
745
746
747
748          .EQUIV BIT12,SKE
749          .EQUIV BIT01,CSE
750          .EQUIV BIT00,WCE
751
752          .SBTTL TRAP CATCHER
753
754          .=0
755          000000          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
756          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
757          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
758          .=174
759          000174          000000          DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
760          000174          000000          SWREG: .WORD 0          ;;SOFTWARE SWITCH REGISTER
761
762          .SBTTL STARTING ADDRESS(ES)
763          000200          000137          003376          JMP @*START ;;JUMP TO STARTING ADDRESS OF PROGRAM
764
765          .=210
766          000210          105237          001253          INCB FRSTAT          ;RESTART ADDRESS, IF RESTART IS
767          000214          000137          003376          JMP @*START          ;DONE AT 210, THE BOUNDARY CONDITION
768          ;TESTS (TST1-7) ARE SKIPPED. IF SW 4
769          ;IS SET THEN THE DISKS ARE NOT REWRITTEN
770          ;(WRDSK) WITH RANDOM PATTERNS. NORMALLY
771          ;ALL THE DISKS PRESENT ARE COMPLETELY
772          ;WRITTEN WITH RANDOM PATTERNS, AT THE
773          ;BEGINING OF THE
774          ;EXERCISER PART OF THE PROGRAM.
775
776          .SBTTL ACT11 HOOKS
777          ;;*****
778          ;HOOKS REQUIRED BY ACT11
779          000220          000046          $SVPC=.          ;SAVE PC
780          000046          022750          SENDAD          ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SECP
781          000052          000000          .WORD 0          ;;2)SET LOC.52 TO ZERO
782          000052          000220          .=$SVPC          ;; RESTORE PC
783
784
785          ;K11 REGISTER DEFINITIONS
786          .SBTTL MEMORY MANAGEMENT DEFINITIONS
787
788

```

```

789          ;*KT11 VECTOR ADDRESS
790
791          000250          MMVEC= 250
792
793          ;*KT11 STATUS REGISTER ADDRESSES
794
795          177572          SR0= 177572
796          177574          SR1= 177574
797          177576          SR2= 177576
798          172516          SR3= 172516
799
800          ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
801
802          172300          KIPDR0= 172300
803          172302          KIPDR1= 172302
804          172304          KIPDR2= 172304
805          172306          KIPDR3= 172306
806          172310          KIPDR4= 172310
807          172312          KIPDR5= 172312
808          172314          KIPDR6= 172314
809          172316          KIPDR7= 172316
810
811          ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
812
813          172320          KDPDR0= 172320
814          172322          KDPDR1= 172322
815          172324          KDPDR2= 172324
816          172326          KDPDR3= 172326
817          172330          KDPDR4= 172330
818          172332          KDPDR5= 172332
819          172334          KDPDR6= 172334
820          172336          KDPDR7= 172336
821
822          ;*KERNEL "I" PAGE ADDRESS REGISTERS
823
824          172340          KIPAR0= 172340
825          172342          KIPAR1= 172342
826          172344          KIPAR2= 172344
827          172346          KIPAR3= 172346
828          172350          KIPAR4= 172350
829          172352          KIPAR5= 172352
830          172354          KIPAR6= 172354
831          172356          KIPAR7= 172356
832
833          ;*KERNEL "D" PAGE ADDRESS REGISTERS
834
835          172360          KDPAR0= 172360
836          172362          KDPAR1= 172362
837          172364          KDPAR2= 172364
838          172366          KDPAR3= 172366
839          172370          KDPAR4= 172370
840          172372          KDPAR5= 172372
841          172374          KDPAR6= 172374
842          172376          KDPAR7= 172376
843
844

```

845  
846  
847  
848  
849  
850  
851 001100  
852 001100  
853 001100 000000  
854 001102 000  
855 001103 000  
856 001104 000000  
857 001106 000000  
858 001110 000000  
859 001112 000000  
860 001114 000  
861 001115 001  
862 001116 000000  
863 001120 000000  
864 001122 000000  
865 001124 000000  
866 001126 000000  
867 001130 000000  
868 001132 000000  
869 001134 000  
870 001135 000  
871 001136 000000  
872 001140 177570  
873 001142 177570  
874 001144 177560  
875 001146 177562  
876 001150 177564  
877 001152 177566  
878 001154 000  
879 001155 002  
880 001156 012  
881 001157 000  
882 001160 000000  
883  
884 001162 000000  
885 001164 000000  
886 001166 000000  
887 001170 000000  
888 001172 000000  
889 001174 000000  
890 001176 000000  
891 001200 000000  
892 001202 000000  
893 001204 000000  
894 001206 177607 000377  
895 001212 077  
896 001213 015  
897 001214 000012  
898  
899 001216 177400  
900 001220 177402

.SBTTL COMMON TAGS

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

. =1100  
SCMTAG: .WORD 0  
SPASS: .WORD 0  
\$STNM: .BYTE 0  
SERFLG: .BYTE 0  
\$ICNT: .WORD 0  
\$LPADR: .WORD 0  
\$LPERR: .WORD 0  
\$ERTTL: .WORD 0  
\$ITEMB: .BYTE 0  
\$ERMAX: .BYTE 1  
\$ERRPC: .WORD 0  
\$GDADR: .WORD 0  
\$BDADR: .WORD 0  
\$GDDAT: .WORD 0  
\$BDDAT: .WORD 0  
\$SAUTOB: .BYTE 0  
\$INTAG: .BYTE 0  
\$SWR: .WORD 0  
\$DISPLAY: .WORD 0  
\$TKS: 177560  
\$TKB: 177562  
\$TPS: 177564  
\$TPB: 177566  
\$NULL: .BYTE 0  
\$FILLS: .BYTE 2  
\$FILLC: .BYTE 12  
\$STPFLG: .BYTE 0  
\$REGAD: .WORD 0  
\$REG0: .WORD 0  
\$REG1: .WORD 0  
\$REG2: .WORD 0  
\$REG3: .WORD 0  
\$REG4: .WORD 0  
\$REG5: .WORD 0  
\$REG6: .WORD 0  
\$REG7: .WORD 0  
\$REG10: .WORD 0  
\$ESCAPE: 0  
\$BELL: .ASCIZ <207><377><377>  
\$QUES: .ASCII /?/  
\$SCRLF: .ASCII <15>  
\$LF: .ASCIZ <12>  
RKDS: .WORD 177400  
RKER: .WORD 177402

:: START OF COMMON TAGS  
:: CONTAINS PASS COUNT  
:: CONTAINS THE TEST NUMBER  
:: CONTAINS ERROR FLAG  
:: CONTAINS SUBTEST ITERATION COUNT  
:: CONTAINS SCOPE LOOP ADDRESS  
:: CONTAINS SCOPE RETURN FOR ERRORS  
:: CONTAINS TOTAL ERRORS DETECTED  
:: CONTAINS ITEM CONTROL BYTE  
:: CONTAINS MAX. ERRORS PER TEST  
:: CONTAINS PC OF LAST ERROR INSTRUCTION  
:: CONTAINS ADDRESS OF 'GOOD' DATA  
:: CONTAINS ADDRESS OF 'BAD' DATA  
:: CONTAINS 'GOOD' DATA  
:: CONTAINS 'BAD' DATA  
:: RESERVED--NOT TO BE USED  
:: AUTOMATIC MODE INDICATOR  
:: INTERRUPT MODE INDICATOR  
:: ADDRESS OF SWITCH REGISTER  
:: ADDRESS OF DISPLAY REGISTER  
:: TTY KBD STATUS  
:: TTY KBD BUFFER  
:: TTY PRINTER STATUS REG. ADDRESS  
:: TTY PRINTER BUFFER REG. ADDRESS  
:: CONTAINS NULL CHARACTER FOR FILLS  
:: CONTAINS # OF FILLER CHARACTERS REQUIRED  
:: INSERT FILL CHARS. AFTER A "LINE FEED"  
:: "TERMINAL AVAILABLE" FLAG (BIT<07>)=0=YES  
:: CONTAINS THE ADDRESS FROM  
:: WHICH (\$REG0) WAS OBTAINED  
:: CONTAINS (( \$REGAD)+0)  
:: CONTAINS (( \$REGAD)+2)  
:: CONTAINS (( \$REGAD)+4)  
:: CONTAINS (( \$REGAD)+6)  
:: CONTAINS (( \$REGAD)+10)  
:: CONTAINS (( \$REGAD)+12)  
:: CONTAINS (( \$REGAD)+14)  
:: CONTAINS (( \$REGAD)+16)  
:: CONTAINS (( \$REGAD)+20)  
:: ESCAPE ON ERROR ADDRESS  
:: CODE FOR BELL  
:: QUESTION MARK  
:: CARRIAGE RETURN  
:: LINE FEED

901	001222	177404	RKCS:	.WORD	177404	
902	001224	177406	RKWC:	.WORD	177406	
903	001226	177410	RKBA:	.WORD	177410	
904	001230	177412	RKDA:	.WORD	177412	
905	001232	177416	RKDB:	.WORD	177416	
906	001234	177546	KWLS:	.WORD	177546	;STATUS REGISTER FOR KW11L
907						
908	001236	000372	PCNTR:	.WORD	250.	
909						
910	001240	000220	RKVEC:	.WORD	220	;NORMAL RK11 INTERRUPT VECTOR ADDRESS
911	001242	000222	RKSTAT:	.WORD	222	;PSW TO BE USED ON INTERRUPT
912						
913	001244	000240	PPRLVL:	.WORD	240	;PROGRAM PRIORITY LEVEL=5. PRIORITY LEVEL
914						;AT WHICH THE PROGRAM OPERATES CAN BE CHANGED
915						;BY ALTERING THIS LOCATION.
916	001246	000340	KWPLVL:	.WORD	340	;PRIORITY LEVEL OF THE KW11L CLOCK SERVICE
917						;ROUTINE.
918						
919	001250	177777	SRDRV:	.WORD	177777	;'SRDRV' CONTAINS THE DRIVE NO WHOOSE SERIAL
920						;NO IS TO BE TYPED OUT WHEN AN ERROR OCCURS,
921						;IF SW 1 IS SET. WHEN (SRDRV)=-1 SERIAL NO
922						;IS NOT TYPED OUT, BECAUSE THE ERROR WAS NOT
923						;POSITIVELY ATTRIBUTABLE TO A SPECIFIC DRIVE.
924						
925						
926	001252	000	FTITLE:	.BYTE	0	
927	001253	000	FRSTRT:	.BYTE	0	;FLAG FOR RESTART AT 210

928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983

001254 000010  
001264 000000  
001266 000010  
001306 000010

: THIS TABLE CONTAINS (IN ASCENDING ORDER) THE DRIVE NUMBERS THAT ARE  
: PRESENT. THUS IF 3 DRIVES 0,1,2 ARE PRESENT: PDR WILL CONTAIN PDR1 WILL  
: CONTAIN 1 AND PDR2 WILL CONTAIN 2. THE UPPER BIT OF EACH 'PDR' BYTE IS SET IF THE  
: CORRESPONDING DRIVE IS AN 'F' DRIVE.

PDR. .BLKB 10  
DRVPRS: .WORD 0 ;CONTAINS TOTAL NUMBER OF DRIVES PRESENT

: THE FOLLOWING LOCATIONS CONTAIN SERIAL NUMBERS CORRESPONDING TO EACH  
: DRIVE. THE SERIAL NUMBERS ARE KEYED IN BY THE USER, WHEN THE PROGRAM  
: IS STARTED WITH SWITCH 1 SET TO 1. THIS FEATURE IS NORMALLY USED IN  
: PRODUCTION ENVIRONMENT.

SRNO: .BLKW 10 ;SERIAL NO'S FOR DRIVES 0-7

: THE FOLLOWING 8 KEYS ARE FOR THE 8 COMMANDS IN THE QUEUE, TO BE  
: EXECUTED ON DIFFERENT DRIVES. EACH KEY IS ASSOCIATED WITH AN EXECUTABLE  
: COMMAND ON THE RK11. VARIOUS BITS OF THE KEY DESCRIBE A COMMAND  
: AS INDICATED BELOW

- : <0-2> DRIVE NUMBER ON WHICH THE COMMAND IS TO BE EXECUTED
- : <4> INDICATES THAT THE HEADS ARE BEING/OR HAVE BEEN POSITIONED ON THE DRIVE
- : <5> INDICATES A 'WPT CHK' SHOULD BE DONE FOLLOWING THE 'WRITE'
- : <6> INDICATES A WRITE CHECK FUNCTION HAS BEEN INITIATED
- : <7> INDICATES THAT A FUNCTION IS IN PROGRESS (IT IS NOT SET WHEN POSITIONING IS BEING DONE ON A DRIVE)
- : <8-10> INDICATES THE POSITION OF THIS KEY IN THE 8-KEY TABLE (POSITIONS BEING 0,1,2,3,4,5,6,7)
- : <11> INDICATES THAT FUNCTION CORRESPONDING TO THIS KEY HAS BEEN ABORTED
- : <12> INDICATES HIGH PRIORITY FOR THE COMMAND (NORMALLY SET AFTER AN ERROR OCCURED ON THE COMMAND)
- : <14> INDICATES THAT THE COMMAND CORRESPONDING TO THIS KEY HAS BEEN ABORTED BECAUSE THE DRIVE WAS DESELECTED (DSELECT)
- : <15> INDICATES THAT THE COMMAND HAS BEEN COMPLETED (ALSO SET WHEN COMMAND IS ABORTED AFTER RETRIES)

KEY: .BLKW 10 ;KEY FOR THE COMMANDS IN QUEUE

: THE PARAMETERS TO BE USED FOR EACH COMMAND IN THE QUEUE  
: ARE STORED IN A TABLE STARTING AT 'CMND'. BITS <8-10>  
: OF THE COMMAND KEYS (KEY, KEY2, ---KEY8) ARE USED TO POINT  
: TO THE RIGHT SET OF PARAMETERS.

- : WORD 1 CONTAINS RKDA TO BE USED
- : WORD 2 CONTAINS RKCS (FUNCTION BITS ONLY)
- : WORD 3 CONTAINS RKWC (WORD COUNT 2'S COMP)
- : WORD 4 CONTAINS RKBA

```

984 001326 000040      CMND:  .BLKW  40      ;STORAGE TABLE
985
986 ;THESE ARE BUSY FLAGS FOR THE DRIVES. IF A DRIVE IS BUSY PERFORMING
987 ;ANY FUNCTION (INCLUDING POSITIONING) THEN BIT 7 OF THE FLAG FOR THAT
988 ;DRIVE IS SET. BITS 0-3 CONTAIN THE OFFSET TO KEY # WHICH MADE THE DRIVE
989 ;BUSY. EX: DRIVE #3 WAS MADE TO DO A WRITE BY COMMAND
990 ;KEYS, HENCE 'BUSY3' WILL CONTAIN 210. NOTE THAT 10 IS THE
991 ;OFFSET FOR KEYS (TAKING KEY AS BASE).  KEY #= OFFSET<0-3>/2 + 1
992
993 001426 000010      BUSY:  .BLKB  10      ;BUSY FLAGS FOR DRIVES 0-7
994
995 ;THESE FLAGS WHEN SET INDICATE THAT A DRIVE IS BEING
996 ;POSITIONED OR HAS ALREADY BEEN POSITIONED.
997
998
999 001436 000010      POS:   .BLKB  10      ;DRIVE 0 POSITIONED
1000
1001
1002
1003 ;RETRY COUNTS FOR A PARTICULAR FUNCTION ON A DRIVE THE FUNCTION IS ABORTED
1004 ;ON A DRIVE WHEN THE RETRY COUNT REACHES 3.
1005
1006 001446 000010      RETRY: .BLKB  10      ;DRIVES 0-7 RERTY COUNTS
1007
1008 001456 000000      WCFLG: .WORD  0      ;IF BIT 15 IS SET WRITE CHK IS TO BE DONE
1009 ;FOLLOWING THE WRITE. BITS 0-3 CONTAIN THE
1010 ;OFFSET TO KEY# (FROM BASE=KEY)
1011
1012 001460 000000      QSCNT: .WORD  0      ;THIS IS A COUNT FOR KEEPING TRACK OF THE TIME
1013 ;TAKEN BY ALL THE B COMMANDS IN THE QUEUE.
1014 ;IF THIS COUNTS DOWN TO 0 AN ERROR IS REPORTED
1015
1016
1017 001462 000000      PRSFNC: .WORD  0      ;CONTAINS INFO ABOUT THE PRESENT COMMAND
1018 ;BEING PERFORMED ON THE RK11
1019 001464 000000      PSTFNC: .WORD  0      ;CONTAINS INFO ABOUT THE COMMAND PERFORMED
1020 ;BEFORE THE 'PRSCMD'
1021
1022
1023 001466 000000      CICNT:  .WORD  0      ;THIS IS A COUNT-TIMER USED FOR KEEPING TRACK
1024 001470 000000      CICNT1: .WORD  0      ;OF THE TIME TAKEN BY ANY FUNCTION TO BE
1025 ;COMPLETED. IF THE COUNT GOES TO 0 AN ERROR IS REPORTED.
1026
1027 001472 000000      TIMER:  .WORD  0
1028 001474 000000      ERCODE: .WORD  0
1029 001476 000000      DRVPTR: .WORD  0
1030 001500 000000      DRVCNT: .WORD  0
1031
1032
1033 001502 000000      QDRV:   .WORD  0      ;TEMPORARY REGISTERS USED BY 'GENBRQ'
1034 001504 000000      QCYL:   .WORD  0      ;ROUTINE TO STORE VARIOUS PARAMETERS
1035 001506 000000      QSUR:   .WORD  0      ;OF A COMMAND AS THEY ARE GENERATED.
1036 001510 000000      QSEC:   .WORD  0
1037 001512 000000      QFNC:   .WORD  0
1038 001514 000000      QBUSAD: .WORD  0
1039 001516 000000      QWRCNT: .WORD  0

```

```

1040
1041
1042           ; THIS TABLE CONTAINS VARIOUS MAPPING FACTORS TO BE USED
1043           ; FOR GENERATING RANDOM PARAMETERS FROM RANDOM NUMBERS
1044
1045 001520 000000  DRMAP: .WORD 0           ; MAPPING FACTOR FOR GENERATING RANDOM DRIVE NUMBER
1046 001522 000000  CYLMAP: .WORD 0          ; MAPPING FACTOR FOR CYLINDER
1047 001524 000000  SECMAP: .WORD 0          ; MAPPING FACTOR FOR SECTOR
1048 001526 000000  FNMAP: .WORD 0           ; MAPPING FACTOR FOR FUNCTION
1049 001530 000000  BAMAP: .WORD 0           ; MAPPING FACTOR FOR BUS ADDRESS
1050 001532 000000  WCMAP: .WORD 0           ; MAPPING FACTOR FOR WORD COUNT
1051
1052
1053           ; THESE TWO FLAGS CORRESPOND TO THE 2 INTERRUPT HANDLERS (RK11) USED
1054           ; IN THIS PROGRAM. WHEN THE INTERRUPT HANDLER IS ENTERED THE FLAG IS
1055           ; CLEARED OR SET.
1056
1057 001534 000      INTFLG: .BYTE 0           ; FOR 'INTHND ', CLEARED ON ENTERING HANDLER
1058 001535 000      INTIFL: .BYTE 0          ; FOR 'INTISK ', SET ON ENTERING HANDLER
1059
1060 001536 000000  SAVKEY: .WORD 0
1061 001540 000000  ECOUNT: .WORD 0
1062
1063           ; THIS TABLE CONTAINS COUNTS FOR THE NUMBER OF OF ERRORS OCCURING ON A
1064           ; DRIVE (NOTE: ONLY THOSE ERRORS WHICH ARE POSITIVELY ATTRIBUTABLE TO A
1065           ; SPECIFIC DRIVE). THE COUNT KEPT ONLY IF SWITCH 2 IS SET. WHEN THE COUNT
1066           ; REACHES THE MAXIMUM ALLOWABLE (USUALLY 3) THE DRIVE IS DROPPED FROM
1067           ; TESTING AND IS TAKEN OUT OF THE DRIVE SELECTION TABLE.
1068
1069 001542 000010  ERDRV: .BLKB 10           ; COUNT FOR DRIVES 0-7
1070
1071 001552 000000  KWHR: .WORD 0           ; COUNTS HOURS (2'S COMPLEMENT)
1072 001554 000000  KWMIN: .WORD 0          ; COUNTS MINUTES (2'S COMPLEMENT)
1073 001556 000000  KWSEC: .WORD 0          ; COUNTS SECONDS (2'S COMPLEMENT)
1074 001560 000000  KWCOUNT: .WORD 0        ; COUNTS CPS FROM KWILL (2'S COMPLMNT)
1075
1076           ; THIS TABLE CONTAINS COUNTS FOR HARD ERRORS ON A PARTICULAR DRIVE.
1077           ; EX HECN2 WILL CONTAIN THE TOTAL NUMBER OF HARD ERRORS THAT OCCURED ON
1078           ; DRIVE 2
1079
1080 001562 000010  HECN: .BLKW 10           ; DRIVE 0-7 HARD ERROR COUNTS
1081
1082           ; THIS TABLE CONTAINS COUNTS FOR SEEK ERRORS
1083           ; ON A PARTICULAR DRIVE.
1084
1085 001602 000010  SKECN: .BLKW 10           ; DRIVE 0-7 SEEK ERROR COUNTS
1086
1087
1088           ; THIS TABLE CONTAINS COUNTS FOR SIN ERRORS ON A
1089           ; PARTICULAR DRIVE
1090
1091 001622 000010  SINCN: .BLKB 10           ; DRIVE 0-7 SIN COUNTS
1092
1093           ; THIS TABLE CONTAINS COUNTS FOR WRITE CHECK ERRORS
1094           ; THAT OCCURED ON A PARTICULAR DRIVE
1095

```

1096 001632 000010

WCECN: .BLKW 10 ;WCE COUNT FOR DRIVES 0-7

1097

1098

1099

1100

;THIS TABLE CONTAINS COUNTS FOR CHECK SUM EPROR THAT  
;OCCURED ON A PARTICULAR DRIVE

1101

1102 001652 000010

CSECN: .BLKW 10 ;CSE COUNT FOR DRIVES 0-7

1103

1104

1105

1106

1107

1108

1109 001672 000010

ABORT: .BLKW 10 ;ABORT COUNT FOR DRIVES 0-7

1110

1111

1112

1113 001712 000010

;COUNTS FOR NUMBER OF DATA ERRORS THAT OCCURED ON INDIVIDUAL DRIVES.

DATER: .BLKW 10 ;DRIVES 0-7



```

1114 001732 000000          NWRTL: .WORD 0          ;LO WORD: OF THE 2 WORD COUNT-GIVING TOTAL
1115 001734 000000          NWRTH: .WORD 0          ;HI WORD: # OF WORDS WRITTEN ON DRIVE C
1116 001736 000016          .BLKW  14.         ;FOR REST OF DRIVES 1-7
1117
1118
1119 001772 000000          NRDL:  .WORD 0          ;LO WORD: 2 WORD COUNT GIVING TOTAL
1120 001774 000000          NRDH:  .WORD 0          ;HI WORD: # OF WORDS READ ON DRIVE C
1121 001776 000016          .BLKW  14.         ;FOR DRIVES 1-7
1122
1123 002032 001326          PCMND: .WORD  CMND      ;POINTERS TO PARAMETERS FOR COMMANDS IN QUEUE
1124 002034 001336          .WORD  CMND+10        ;POINTER TO SECOND COMMAND
1125 002036 001346          .WORD  CMND+20        ;POINTER TO THIRD COMMAND
1126 002040 001356          .WORD  CMND+30        ;POINTER TO FOURTH COMMAND
1127 002042 001366          .WORD  CMND+40        ;POINTER TO FIFTH COMMAND
1128 002044 001376          .WORD  CMND+50        ;POINTER TO SIXTH COMMAND
1129 002046 001406          .WORD  CMND+60        ;POINTER TO SEVENTH COMMAND
1130 002050 001416          .WORD  CMND+70        ;POINTER TO EIGHTH COMMAND
1131
1132
1133 002052 000000          BASEBA: .WORD 0          ;CONTAINS THE LOWEST BUS ADDRESS STARTING WHICH DATA TRANSFERS
1134                                     ;CAN BE DONE
1135 002054 000000          MAXBA:  .WORD 0          ;CONTAINS THE HIGHEST BUS ADDRESS TO WHICH DATA TRANSFERS
1136                                     ;CAN BE DONE.
1137 002056 000000          REPCNT: .WORD 0          ;CONTAINS THE REPETITION COUNT- THE NUMBER
1138                                     ;OF TIMES A REQUESTS WILL BE GENERATED. WHEN THIS
1139                                     ;COUNT GOES TO 0, IT MEANS AN END OF PASS. HOWEVER
1140                                     ;NOTE THAT THERE IS NO TRUE END OF PASS, IN THIS KIND
1141                                     ;OF EXERCISER PROGRAM. THE EXERCISER RESUMES FROM
1142                                     ;THE POINT IT LEFT OFF, AFTER TYPING OUT THE END IF
1143                                     ;PASS MESSAGE.
1144 002060 000000          XXDPMO: .WORD 0          ;LOW BYTE CONTAINS ADDRESS OF RK05 DRIVE
1145                                     ;WHICH PROGRAM WAS LOADED FROM; HIGH BYTE
1146                                     ;CONTAINS THE RK05 'XXOP' CODE.
1147
1148                                     ;ASCII MESSAGES
1149
1150 002062 005015 045523 000105 MSG1:  .ASCIZ <15><12>/SKE/
1151 002070 005015 041527 000105 MSG2:  .ASCIZ<15><12>/WCE/
1152 002076 005015 051503 000105 MSG3:  .ASCIZ<15><12>/CSE/
1153 002104 005015 040510 042122 MSG4:  .ASCIZ <15><12>/HARD EROR/
1154 002112 042440 047522 000122
1155 002120 047440 020116 047504 MSG5:  .ASCIZ/ ON DOING /
1156 002126 047111 020107 000
1157 002133 127 044522 042524 MSG6:  .ASCIZ /WRITE/
1158 002140 000
1159 002141 122 040505 000104 MSG7:  .ASCIZ /READ/
1160 002146 051127 020124 044103 MSG8:  .ASCIZ /WRT CHK/
1161 002154 000113
1162 002156 042122 041440 045510 MSG9:  .ASCIZ /RD CHK/
1163 002164 000
1164 002165 015 040412 047502 MSG10: .ASCIZ <15><12>/ABORTED/<15><12>
1165 002172 052122 042105 005015
1166 002200 000
1167 002201 123 042505 000113 MSG11: .ASCIZ /SEEK/
1168 002206 005015 041520 000075 MSG12: .ASCIZ <15><12>/PC=/
1169 002214 044120 051531 041040 MSG13: .ASCIZ /PHYS BA=/

```



1226 002662 040  
1227 002663 040  
1228 002664 000040  
1229

BLNKS3: .ASCII //  
BLNKS2: .ASCII //  
BLNKS1: .ASCIZ //  
.EVEN

1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285

002666

002666 027632  
002670 031730  
002672 032422  
002674 000000  
  
002676 027650  
002700 031776  
002702 032436  
002704 000000  
  
002706 027723  
002710 031730  
002712 032422  
002714 000000  
  
002716 027746  
002720 031730  
002722 032422  
002724 000000  
  
002726 027771  
002730 031730  
002732 032422

.SBTTL ERROR POINTER TABLE

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

```

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

\$ERRTB:

\*THERE ARE TWO CLASSES OF ERRORS:  
 :\*1. ERRORS IN EXERCISER PART OF THE PROGRAM - ERROR NUMBERS BELOW 100  
 :\*2. ERRORS IN THE NON-EXERCISER PART OF THE PROGRAM - ERROR NUMBERS EQUAL  
 :\*TO AND GREATER THAN 100.  
 :\*THE DOCUMENT CONTAINS MORE INFORMATION ON THESE.  
 :\*THE FOLLOWING ERRORS OCCUR IN THE EXERCISER PART OF THE PROGRAM.

:ITEM 1

```

EM1      ;ERROR ON WRITE
DH1      ;PC      RKCS      RKER      RKDS      RKDA
DT1      ;$ERRPC $REG0      $REG1      $REG2      $REG3
0
    
```

:ITEM 2

```

EM2      ;ATTEMPT TO INITIATE FUNCTION ON 'BUSY' DRIVE
DH2      ;PC      DRIVE
DT2      ;$ERRPC $REG0
0
    
```

:ITEM 3

```

EM3      ;CONTROL READY NOT SET
DH1      ;PC      RKCS      RKER      RKDS      RKDA
DT1      ;$ERRPC $REG0      $REG1      $REG2      $REG3
0
    
```

:ITEM 4

```

EM4      ;R/W/S READY NOT SET
DH1      ;PC      RKCS      RKER      RKDS      RKDA
DT1      ;$ERRPC $REG0      $REG1      $REG2      $REG3
0
    
```

:ITEM 5

```

EM5      ;CONTROL READY NOT SET AFTER FIRST INTERRUPT ON ISSUING SEEN
DH1      ;PC      RKCS      RKER      RKDS      RKDA
DT1      ;$ERRPC $REG0      $REG1      $REG2      $REG3
    
```

1286	002734	000000	0	
1287				
1288				
1289			:ITEM	6
1290				
1291	002736	030056	EM6	:WRONG BITS IN RKCS EXPECT SEEK
1292	002740	031730	DH1	:PC RKCS RKER RKDS RKDA
1293	002742	032422	DT1	:SERRPC \$REG0 \$REG1 \$REG2 \$REG3
1294	002744	000000	0	
1295				
1296			:ITEM	7
1297				
1298	002746	030115	EM7	: 'BUSY' FLAG CLEAR ON INTERRUPTING DRIVE
1299	002750	031776	DH2	:PC DRIVE
1300	002752	032436	DT2	:SERRPC \$REG0
1301	002754	000000	0	
1302				
1303			:ITEM	10
1304				
1305	002756	030162	EM10	: 'POSITIONING' FLAG FOR INTERRUPTING DRIVE CLEAR
1306	002760	031776	DH2	:PC DRIVE
1307	002762	032436	DT2	:SERRPC \$REG0
1308	002764	000000	0	
1309				
1310			:ITEM	11
1311				
1312	002766	030237	EM11	: 'ERR'OR SET AFTER FIRST INTERRUPT ON ISSUING SEEK
1313	002770	031730	DH1	:PC RKCS RKER RKDS RKDA
1314	002772	032422	DT1	:SERRPC \$REG0 \$REG1 \$REG2 \$REG3
1315	002774	000000	0	
1316				
1317			:ITEM	12
1318				
1319	002776	030317	EM12	:SCP SET AFTER FIRST INTERRUPT ON ISSUING SEEK
1320	003000	031730	DH1	:PC RKCS RKER RKDS RKDA
1321	003002	032422	DT1	:SERRPC \$REG0 \$REG1 \$REG2 \$REG3
1322	003004	000000	0	
1323				
1324			:ITEM	13
1325				
1326	003006	030371	EM13	:CONTROL READY NOT SET AFTER SEEK DONE INTERRUPT
1327	003010	031730	DH1	:PC RKCS RKER RKDS RKDA
1328	003012	032422	DT1	:SERRPC \$REG0 \$REG1 \$REG2 \$REG3
1329	003014	000000	0	
1330				
1331			:ITEM	14
1332				
1333	003016	030444	EM14	: INTERRUPTING DRIVE (SEEK DONE) WAS NOT 'BUSY'
1334	003020	031730	DH1	:PC RKCS RKER RKDS RKDA
1335	003022	032422	DT1	:SERRPC \$REG0 \$REG1 \$REG2 \$REG3
1336	003024	000000	0	
1337				
1338			:ITEM	15
1339				
1340	003026	030517	EM15	:R/W/S READY NOT SET FOR INTERRUPTING DRIVE (SEEK DONE)
1341	003030	031730	DH1	:PC RKCS RKER RKDS RKDA



1398			: ITEM	25						
1399				0						
1400	003126	000000		DH25	: PC	RKCS	RKER	RKDS	RKDA	DRIVE #
1401	003130	032155		DT25	: SERRPC	\$REG0	\$REG1	\$REG2	\$REG3	\$REG4
1402	003132	032466		0						
1403	003134	000000		0						
1404			: ITEM	26						
1405				EM26	: STUCK IN LOOP. 8 Q-COMMANDS SHOULD BE DONE BY NOW					
1406				DH1	: PC	RKCS	RKER	RKDS	RKDA	
1407	003136	031023		DT1	: SERRPC	\$REG0	\$REG1	\$REG2	\$REG3	
1408	003140	031730		0						
1409	003142	032422		0						
1410	003144	000000		0						
1411			: ITEM	27						
1412				EM27	: ATTEMPT TO DO WRITE CHECK BEFORE WRITE					
1413				DH27	: PC	KEY	FJUNCTION	CODE		
1414	003146	031076		DT103	: PC	\$REG0	\$REG1			
1415	003150	032232		0						
1416	003152	032504		0						
1417	003154	000000		0						
1418			: ITEM	30						
1419				EM30	: ATTEMPT TO REEXECUTE A COMMAND IN PROGRESS OR ALREADY FINISHED					
1420				DH30	: PC	KEY				
1421	003156	031137		DT2	: SERRPC	\$REG0				
1422	003160	032313		0						
1423	003162	032436		0						
1424	003164	000000		0						
1425			: ITEM	31						
1426				EM31	: 'FUNCTION IN PROGRESS' FLAG FOR INTERRUPTING DRIVE IS NOT SET					
1427				DH2	: PC	DRIVE				
1428	003166	031232		DT2	: SERRPC	\$REG0				
1429	003170	031776		0						
1430	003172	032436		0						
1431	003174	000000		0						
1432			: ITEM	32						
1433				EM32	: UNEXPECTED DRIVE INTERRUPTED					
1434				DH103	: PC	EXPT	RECVD			
1435	003176	031323		DT103	: SERRPC	\$REG0	\$REG1			
1436	003200	032265		0						
1437	003202	032504		0						
1438	003204	000000		0						
1439			: ITEM	33						
1440				EM33	: WRONG FUNCTION CODE IN RKCS AFTER INTERRUPT					
1441				DH103	: PC	EXPT	RECVD			
1442	003206	031355		DT103	: SERRPC	\$REG0	\$REG1			
1443	003210	032265		0						
1444	003212	032504		0						
1445	003214	000000		0						
1446			: ITEM	34						
1447				EM34	: DRIVE READY CLEAR					
1448				DH1	: PC	RKCS	RKER	RKDS	RKDA	
1449	003216	031432		DT1	: SERRPC	\$REG0	\$REG1	\$REG2	\$REG3	
1450	003220	031730		0						
1451	003222	032422		0						
1452	003224	000000		0						

1454			: ITEM	35					
1455					EM35	: DRIVE POWER LOW			
1456	003226	031451			DH1	: PC RKCS	RKER	RKDS	RKDA
1457	003230	031730			DT1	: \$ERRPC \$REG0	\$REG1	\$REG2	\$REG3
1458	003232	032422							
1459	003234	000000			0				
1460									
1461			: ITEM	36					
1462									
1463	003236	031467			EM36	: DRIVE UNSAFE			
1464	003240	031730			DH1	: PC RKCS	RKER	RKDS	RKDA
1465	003242	032422			DT1	: \$ERRPC \$REG0	\$REG1	\$REG2	\$REG3
1466	003244	000000			0				
1467									
1468			: ITEM	37					
1469									
1470	003246	031503			EM37	: WPS SET			
1471	003250	031730			DH1	: PC RKCS	RKER	RKDS	RKDA
1472	003252	032422			DT1	: \$ERRPC \$REG0	\$REG1	\$REG2	\$REG3
1473	003254	000000			0				
1474									
1475			: *						
1476			: *						
1477			: *						
1478									
1479			: ITEM	100					
1480									
1481	003256	030720			EM23	: DATA (COMPARISON) ERROR			
1482	003260	032110			DH23	: PC RKBA	EXPCT	RECVD	RKDA
1483	003262	032422			DT1	: \$ERRPC \$REG0	\$REG1	\$REG2	\$REG3
1484	003264	003000			0				
1485									
1486			: ITEM	101					
1487									
1488	003266	031513			EM101	: INTERRUPT DID NOT OCCUR AFTER WRITE			
1489	003270	031730			DH1	: PC RKCS	RKER	RKDS	RKDA
1490	003272	032422			DT1	: \$ERRPC \$REG0	\$REG1	\$REG2	\$REG3
1491	003274	000000			0				
1492									
1493			: ITEM	102					
1494									
1495	003276	031553			EM102	: 'ERR'OR SET			
1496	003300	031730			DH1	: PC RKCS	RKER	RKDS	RKDA
1497	003302	032422			DT1	: \$ERRPC \$REG0	\$REG1	\$REG2	\$REG3
1498	003304	000000			0				
1499									
1500			: ITEM	103					
1501									
1502	003306	031567			EM103	: RKDA INCREMENTED WRONGLY			
1503	003310	032265			DH103	: PC EXPCT	RECVD		
1504	003312	032504			DT103	: \$ERRPC \$REG0	\$REG1		
1505	003314	000000			0				
1506									
1507			: ITEM	104					
1508									
1509	003316	031615			EM104	: RKBA INCREMENTED WRONGLY			



1510	003320	032265	DH103	;PC	EXPCT	RECVD			
1511	003322	032504	DT103	;	SERRPC	\$REG0	\$REG1		
1512	003324	000000	0						
1513									
1514			:ITEM	105					
1515									
1516	003326	031643	EM105	;	RKWC	DID NOT	OVERFLOW	TO	0
1517	003330	032327	DH105	;PC	RKDA	RKWC			
1518	003332	032504	DT103	;	SERRPC	\$REG0	\$REG1		
1519	003334	000000	0						
1520									
1521			:ITEM	106					
1522									
1523	003336	031673	EM106	;	MEX	BITS	INCORRECT		
1524	003340	031730	DH1	;PC	RKCS	RKER	RKDS	RKDA	
1525	003342	032422	DT1	;	SERRPC	\$REG0	\$REG1	\$REG2	\$REG3
1526	003344	000000	0						
1527									
1528			:ITEM	107					
1529									
1530	003346	030720	EM23	;	DATA	(COMPARISON)	ERROR	ON	READ
1531	003350	032265	DH103	;PC	EXPCT	RECVD			
1532	003352	032504	DT103	;	SERRPC	\$REG0	\$REG1		
1533	003354	000000	0						
1534									
1535			:ITEM	110					
1536									
1537	003356	031712	EM110	;	WRITE	CHECK	ERROR		
1538	003360	032354	DH110	;PC	RKCS	RKER	RKBA	RKDA	
1539	003362	032422	DT1	;	SERRPC	\$REG0	\$REG1	\$REG2	\$REG3
1540	003364	000000	0						

```

1541 ;IF POWER FAILED, ON RETURN OF POWER ENTER HERE.
1542
1543 PFSTRT: ISR          PC WAITIME          ;WAIT SOME TIME
1544 003366 004737 022630 001253 INCB          FRSTRT          ;INDICATE THAT THE STATISTICS HAVE
1545 ;TO BE SAVED, ON RETRN FROM PWR FAIL.
1546
1547
1548
1549
1550 003376 000005          START: RESET          ;CLEAR THE BUS
1551 ;;GIVE DRIVES TIME TO LOAD IF PROGRAM WAS STARTED BY APT
1552 003400 023737 000042 000046          CMP          J#42,J#46
1553 003406 001016          BNE          STARTA
1554 003410 005077          CLR          J#KDA          ;SELECT UNIT 0
1555 003414 012700 000250          MOV          #250,R0          ;WAIT FOR..
1556 003420 032777 000200 175570 20$: BIT          #200,J#KDS          ;DRIVE READY..
1557 003426 001006          BNE          STARTA          ;IN CASE..
1558 003430 005001          CLR          R1          ;OF APT..
1559 003432 005301          DEC          R1          ;START, BUT..
1560 003434 001376          BNE          #-2          ;DON'T WAIT..
1561 003436 005300          DEC          R0          ;FOREVER.
1562 003440 001367          BNE          20$
1563 003442 000000          HALT          ;RKDS BIT 7 (DRIVE READY) NEVER SET.
1564 003444
1565
1566          STARTA:
1567          .SBTTL INITIALIZE THE COMMON TAGS
1568          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1569          MOV          # $CMTAG,R6          ;;FIRST LOCATION TO BE CLEARED
1570          CLR          (R6)+          ;;CLEAR MEMORY LOCATION
1571          CMP          #SWR,R6 ;;DONE?
1572          BNE          #-6          ;;LOOP BACK IF NO
1573          MOV          #STACK,SP          ;;SETUP THE STACK POINTER
1574          ;;INITIALIZE A FEW VECTORS
1575          MOV          #SCOPE,J#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1576          MOV          #340,J#IOTVEC+2 ;;LEVEL 7
1577          MOV          #STRAP,J#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1578          MOV          #340,J#TRAPVEC+2 ;;LEVEL 7
1579          MOV          #SPWRON,J#PWRVEC ;;POWER FAILURE VECTOR
1580          MOV          #340,J#PWRVEC+2 ;;LEVEL 7
1581          MOV          #.,SLPADR          ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1582          MOV          #.,SLPERR          ;;SETUP THE ERROR LOOP ADDRESS
1583          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1584          ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1585          MOV          J#ERRVEC, -(SP) ;;SAVE ERROR VECTOR
1586          MOV          #64$,J#ERRVEC ;;SET UP ERROR VECTOR
1587          MOV          #DSWR,SWR          ;;SETUP FOR A HARDWARE SWICH REGISTER
1588          MOV          #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1589          CMP          #-1,J#SWR          ;;TRY TO REFERENCE HARDWARE SWR
1590          BNE          66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1591          ;;AND THE HARDWARE SWR IS NOT = -1
1592          BR          65$          ;;BRANCH IF ) TIMEOUT
1593          MOV          #65$, (SP) ;;SET UP FOR TRAP RETURN
1594          RTI
1595          MOV          #SWREG,SWR ;;POINT TO SOFTWARE SWR
1596          MOV          #DISPREG,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1597          MOV          (SP)+,J#ERRVEC ;;RESTORE ERROR VECTOR
    
```

```

1597 003632 023737 000042 000046      CMP      2042,2046      ;ARE WE IN ACT11 AUTOMATIC MODE?
1598 003640 001416                      BEQ      69$           ;YES, SKIP TITLE PRINTOUT
1599                                     ;:SBTTL TYPE PROGRAM NAME
1600                                     ;:TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1601 003642 005227 177777      INC      #-1           ;:FIRST TIME?
1602 003646 001052                      BNE     67$           ;:BRANCH IF NO
1603 003650 104401 003706      TYPE    68$           ;:TYPE ASCIZ STRING
1604                                     .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1605 003654 005737 000042      TST     2042          ;:ARE WE RUNNING UNDER XXDP/ACT?
1606 003660 001006                      BNE     69$           ;:BRANCH IF YES
1607 003662 023727 001140 000176      CMP     SWR,#SWREG    ;:SOFTWARE SWITCH REG SELECTED?
1608 003670 001005                      BNE     70$           ;:BRANCH IF NO
1609 003672 104406                      GTSWR                    ;:GET SOFT-SWR SETTINGS
1610 003674 000403                      BR      70$
1611 003676 112737 000001 001134 69$:  MOV     #1,$AUTOB    ;;SET AUTO-MODE INDICATOR
1612 003704 70$:
1613 003704 000433                      BR      67$           ;:GET OVER THE ASCIZ
1614                                     ;:68$: .ASCIZ <CRLF>*RK11/RK05 PERFORMANCE EXERCISER*15<12>*MAINDEC-11-DCRKH-G* <CRLF>
1615                                     ;:67$:
1616 003774 012737 026206 000030      MOV     #ERROR,20EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
1617 004002 013737 001244 000032      MOV     PPRLVL,20EMTVEC+2 ;LEVEL 5
1618 004010 012737 022552 000100      MOV     #KWSRV,20KWLVEC ;KWILL CLOCK SERVICE
1619 004016 013737 001246 000102      MOV     KWPLVL,20KWLVEC+2 ;LEVEL 7
1620
1621                                     ;:THE FOLLOWING CODE FINDS OUT THE PROGRAM CONTROL MODE:
1622                                     ;:PAPER TAPE (MANUAL), ACT11, RKDP CHAIN OR DUMP
1623
1624
1625 004024 005037 002060                      CLR     XXDPMO        ;CLEAR 'XXDP' LOAD DEVICE STORAGE
1626 004030 122737 000002 000041      CMPB   #2,41         ;LOADED FROM AN RK05 ?
1627 004036 001160                      BNE    ST2           ;BR IF NOT
1628 004040 013737 000040 002060      MOV     40,XXDPMO    ;GET DEVICE INDICATOR AND DRIVE ADDRESS OF
1629                                     ;LOADING RK05
1630 004046 122737 000010 002060      CMPB   #10,XXDPMO   ;VALID DRIVE ADDRESS ?
1631 004054 101002                      BHI    2$           ;BR IF YES
1632 004056 105037 002060                      CLRB   XXDPMO        ;CHANGE TO DRIVE ZERO
1633 004062 005737 000042 2$:      TST     42           ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
1634 004066 001424                      BEQ    3$           ;BR IF NEITHER
1635 004070 104401 004076      TYPE    72$         ;:TYPE ASCIZ STRING
1636 004074 000413                      BR     71$         ;:GET OVER THE ASCIZ
1637                                     ;:72$: .ASCIZ <15><12>/NOT TESTING DRIVE /
1638                                     ;:71$:
1639 004124 005046                      CLR     -(SP)        ;CLEAR WORD ON STACK
1640 004126 113716 002060      MOV     XXDPMO,(SP) ;GET DRIVE ADDRESS
1641 004132 104403                      TYPOS                    ;TYPE THE ADDRESS
1642 004134 001                      .BYTE  1            ;ONLY 1 CHARACTER
1643 004136 000                      .BYTE  0            ;SUPPRESS LEADING ZEROS
1644 004136 000520                      BR     ST2          ;GET NUMBER OF DRIVES
1645 004140 005227 177777 3$:      INC     #-1         ;FIRST TIME THROUGH HERE ?
1646 004144 001115                      BNE    ST2          ;BR IF NOT
1647 004146 104401 004154      TYPE    74$         ;:TYPE ASCIZ STRING
1648 004152 000411                      BR     73$         ;:GET OVER THE ASCIZ
1649                                     ;:74$: .ASCIZ <15><12>/TO TEST DRIVE /
1650                                     ;:73$:
1651 004176 005046                      CLR     -(SP)        ;CLEAR WORD ON THE STACK
1652 004200 113716 002060      MOV     XXDPMO,(SP) ;GET DRIVE ADDRESS

```

1653	004204	104403		TYPOS			;TYPE THE DRIVE ADDRESS
1654	004206	001		.BYTE	1		;ONLY 1 CHARACTER
1655	004207	000		.BYTE	0		;SUPPRESS LEADING ZEROS
1656	004210	104401	004216	TYPE	76S		;TYPE ASCIZ STRING
1657	004214	000431		BR	75S		;GET OVER THE ASCIZ
1658				75S:	.ASCIZ	/HALT PROGRAM, REMOVE RKDP PACK AND REPLACE IT/<<15><12>	
1659	004300						
1660	004300	104401	004306	TYPE	78S		;TYPE ASCIZ STRING
1661	004304	000435		BR	77S		;GET OVER THE ASCIZ
1662				78S:	.ASCIZ	/WITH A WORK PACK, CLEAR LOCATION 40, AND RESTART PROGRAM/	
1663	004400			77S:			
1664							
1665							
1666							
1667	004400	104416		ST2:	CON.RESET		
1668	004402	005037	001264	CLR	DRVPRS		;FIND WHICH DRIVE #'S ARE PRESENT
1669	004406	005000		CLR	R0		
1670	004410	005002		CLR	R2		
1671	004412	005037	001254	CLR	PDR		;CLEAR DRIVES PRESENT TABLE
1672	004416	005037	001256	CLR	PDR+2		
1673	004422	005037	001260	CLR	PDR+4		
1674	004426	005037	001262	CLR	PDR+6		
1675	004432	012701	001254	MOV	#PDR,R1		
1676	004436	012703	001306	MOV	#KEY,R3		
1677	004442	010277	174562	15:	MOV	R2,ARKDA	;SELECT A DRIVE
1678	004446	032777	000200	174542	BIT	#200,ARKDS	;IS IT IN SYSTEM?
1679	004454	001415		BEQ	3S		;NO
1680	004456	010237	001502	MOV	R2,QDRV		;LOAD DRIVE ADDRESS INTO QDRV
1681	004462	104420		DRV.RESET			;RESET THE DRIVE
1682	004464	005737	002060	TST	XXDPMO		;PROGRAM LOADED FROM AN RKOS ?
1683	004470	001403		BEQ	2S		;BR IF NOT
1684	004472	120037	002060	CMPE	R0,XXDPMO		;LOADED FROM THIS RKOS ?
1685	004476	001404		BEQ	3S		;BR IF YES
1686	004500	110021		25:	MOVB	R0,(R1)+	;STORE THE DRIVE NUMBER
1687	004502	010223		MOV	R2,(R3)+		;STORE ADDRESS IN KEY TABLE
1688	004504	005237	001264	INC	DRVPRS		;BUMP THE NUMBER OF DRIVES COUNTER
1689	004510	062702	020000	35:	ADD	#20000,R2	;NEXT DRIVE ADDRESS
1690	004514	005200		INC	R0		;NEXT DRIVE NUMBER
1691	004516	022700	000010	CMP	#8,R0		;DONE ALL DRIVES???
1692	004522	001347		BNE	1S		;LOOP TILL DONE
1693	004524	013703	001264	MOV	DRVPRS,R3		;FIND WHICH DRIVES ARE TYPE F
1694	004530	001510		BEQ	ST4		;BR IF NOT DRIVES PRESENT
1695	004532	012701	001254	MOV	#PDR,R1		
1696	004536	005000		CLR	R0		
1697	004540	005002		CLR	R2		
1698	004542	104401	002362	45:	TYPE	,MSG20	
1699	004546	111102		MOVB	(R1),R2		;GET DRIVE NUMBER
1700	004550	010200		MOV	R2,R0		
1701	004552	002472		BLT	9S		
1702							
1703	004554	104401	002372	TYPE	,MSG24		
1704							
1705	004560	010246		MOV	R2,-(SP)		;TYPE THE DRIVE NUMBER
1706	004562	004403		TYPOS			
1707	004564	001		.BYTE	1		
1708	004565	000		.BYTE	0		

1709	004566	000241				CLC			;MOVE DRIVE NUMBER TO BITS 15,14,13
1710	004570	006002				ROR	R2		;BIT0 TO CARRY
1711	004572	006002				ROR	R2		;BIT0 TO BIT15
1712	004574	006002				ROR	R2		;BIT0 TO BIT14
1713	004576	006002				ROR	R2		;BIT0 TO BIT13
1714	004600	042702	017777			BIC	#17777,R2		;CLEAR ANY EXTRANEIOUS BITS
1715									
1716	004604	010237	001502			MOV	R2,QDRV		
1717	004610	104420				DRV.RESET			;RESET THE DRIVE VIA QDRV
1718									
1719	004612	032702	020000			BIT	#20000,R2		;EVEN DRIVE NUMBER???
1720	004616	001003				BNE	5\$		;NO - CLEAR BIT
1721									
1722	004620	052702	020000			BIS	#20000,R2		;MAKE IT AN ODD DRIVE
1723	004624	000402				BR	6\$		
1724									
1725	004626	042702	020000		5\$:	BIC	#20000,R2		;MAKE IT AN EVEN DRIVE
1726									
1727	004632	010277	174372		6\$:	MOV	R2,ARKDA		;SELECT THE NEW DRIVE
1728	004636	032777	000200	174352		BIT	#200,ARKDS		;MAKE SURE DRIVE IS IN SYSTEM
1729	004644	001420				BEQ	8\$		;IF NOT, SKIP THIS TEST
1730									
1731	004646	012777	000011	174346		MOV	#11,ARKCS		;START A SEEK TO CYL 0
1732	004654	104417				CON.RDY			;WAIT FOR CONTROLLER
1733	004656	032777	000100	174332		BIT	#100,ARKDS		;IS IT IN MOTION???
1734	004664	001010				BNE	8\$		;NO - J TYPE DRIVE
1735									
1736	004666	152711	000200			BISB	#200,(R1)		;YES - SET THE F TYPE BIT
1737	004672	104401	002375			TYPE	,MSG25		
1738									
1739	004676	032777	000100	174312	7\$:	BIT	#100,ARKDS		;WAIT FOR HEADS TO STOP
1740	004704	001774				BEQ	7\$		
1741									
1742	004706	105737	001253		8\$:	TSTB	FRSTAT		
1743	004712	001012				BNE	9\$		
1744									
1745	004714	032777	000002	174216		BIT	#SW1,ASWR		;SERIAL NO. SW SET?
1746	004722	001406				BEQ	9\$		;NO
1747									
1748	004724	104401	002326			TYPE	,MSG17		;TYPE "SR NO"
1749	004730	104413				RODEC			;READ FROM TTY INPUT
1750	004732	006300				ASL	R0		;SAVE SERIAL NO FOR THE DRIVE
1751	004734	012660	001266			MOV	(SP)+,SRNO(R0)		
1752									
1753	004740	005201			9\$:	INC	R1		
1754	004742	005303				DEC	R3		
1755	004744	003300				BGT	4\$		
1756	004746	104401	001213			TYPE	,\$CRLF		

```

1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774 004752 004737 017466
1775 004756 012702 002052
1776 004762 012703 002054
1777 004766 005737 017524
1778 004772 100022
1779 004774 013700 017772
1780 005000 020027 001540
1781 005004 002012
1782
1783 005006 162700 000040
1784 005012 012701 177772
1785
1786 005016 006300
1787 005020 005201
1788 005022 001375
1789 005024 162700 000002
1790 005030 000415
1791
1792 005032 012713 147776
1793 005036 000413
1794
1795 005040 013700 017770
1796
1797
1798 005044 005737 000040
1799 005050 001003
1800 005052 162700 000500
1801 005056 000402
1802 005060 162700 006000
1803
1804 005064 010013
1805
1806 005066 012712 032514
1807 005072 032777 000100 174040
1808 005100 001510
1809 005102 104401 005110
1810 005106 000432
1811
1812 005174
    
```

```

; 'MAXBA' IS THE HIGHEST BUS ADDRESS (MEMORY) TO WHICH DATA TRANSFERS CAN
; BE DONE BY THE PROGRAM.
; 'MAXBA' IS FIGURED USING THE FOLLOWING ALGORITHM:
; 1. IF KT11 IS NOT PRESENT,
; A. AND THE PROGRAM IS RUN UNDER XXDP, THEN THE TOP 1.5 K IS RESERVED
; AND THE 'MAXBA' IS COMPUTED ($LSTAD-6000).
; B. AND THE PROGRAM IS NOT RUNNING UNDER XXDP, THEN THE TOP 320 WORDS
; ARE RESERVED FOR 'MOM',LOADER,ETC. AND THE 'MAXBA' IS COMPUTED ($LSTAD-500).
; 2. IF KT11 IS PRESENT,
; A. AND MORE THAN 28K MEMORY IS PRESENT, THEN THE MAXIMUM BUS ADDRESS
; IS 147776 (OCTAL).
; B. AND LESS THAN 28K IS PRESENT, THEN THE TOP 2K IS RESERVED FOR RKDP
; MONITOR AND 'MAXBA' IS COMPUTED.
; FIGURE OUT THE AVAILABLE MEMORY AND 'MAXBA'

ST4: JSR PC,$SIZE ;GO SIZE THE MEMORY
      MOV #BASEBA,R2 ;INITIALIZE POINTERS
      MOV #MAXBA,R3
      TST $KT11 ;KT11 AVAILABLE?
      BPL 4$ ;NO
      MOV $LSTBK,R0 ;GET THE LAST BANK OF MEMORY
      CMP R0,#1540 ;28K OR MORE?
      BGE 3$ ;YES

      SUB #40,R0 ;BACK UP 2 K'S (RKDP MONITOR, ETC.)
      MOV #6,R1 ;AND FORM THE MAXIMUM BUS ADDRESS
                        ;FOR DATA TRANSFER

1$: ASL R0
    INC R1
    BNE 1$
    SUB #2,R0
    BR 6$

2$: BR 6$

3$: MOV #147776,(R3) ;FOR 28K OR MORE, THIS IS THE 'MAXBA'
    BR 7$

4$: MOV $LSTAD,R0 ;KT11 NOT PRESENT, GET THE LAST
                        ;AVAILABLE ADDRESS

5$: TST #40 ;'XXDP' LOADED PROGRAM ?
    BNE 8$ ;YES
    SUB #500,R0 ;NO, SAVE THE LAST 320 WORDS
    BR 6$

6$: SUB #6000,R0 ;SAVE THE LAST 1.5K OF MEMORY (RKDP
                        ;MONITOR, ETC.)
    MOV R0,(R3) ;SAVE THE MAXIMUM BUS ADDRESS(MAXBA) TO
                        ;WHICH DATA TRANSFER CAN BE DONE SAFELY
                        ;'BASEBA'

7$: MOV #PGEND,(R2)
    BIT #SW06,$SWR
    BEQ ST3
    TYPE 65$ ;TYPE ASCIZ STRING
    BR 64$ ;GET OVER THE ASCIZ

;65$: .ASCIZ <15><12>/TYPE OCTAL BUS ADDRESSES FOR DATA XFER. BETWEEN
64$:
    
```

```

1813 005174 011246 MOV (R2),-(SP) ;'BASEBA'
1814 005176 104402 TYPOC
1815 005200 104401 005206 TYPE ,67$ ;;TYPE ASCIZ STRING
1816 005204 000402 BR 66$ ;;GET OVER THE ASCIZ
1817 ::67$: .ASCIZ / & /
1818 66$:
1819 005212 011346 MOV (R3),-(SP) ;'MAXBA'
1820 005214 104402 TYPOC
1821 005216 9$:
1822 005216 104401 005224 TYPE ,69$ ;;TYPE ASCIZ STRING
1823 005222 000407 BR 68$ ;;GET OVER THE ASCIZ
1824 ::69$: .ASCIZ <15><12>/LO LIMIT /
1825 68$:
1826 005242 104412 RDOCT
1827 005244 012600 MOV (SP)+,RO
1828 005246 020012 CMP RO,(R2) ;CORRECT LO LIMIT?
1829 005250 103762 BLO 9$
1830 005252 020013 CMP RO,(R3) ;CORRECT LO LIMIT?
1831 005254 103360 BHIS 9$
1832 005256 010012 MOV RO,(R2) ;'BASEBA'
1833 10$:
1834 005260 104401 005266 TYPE ,71$ ;;TYPE ASCIZ STRING
1835 005264 000407 BR 70$ ;;GET OVER THE ASCIZ
1836 ::71$: .ASCIZ <15><12>/HI LIMIT /
1837 70$:
1838 005304 104412 RDOCT
1839 005306 012600 MOV (SP)+,RO
1840 005310 020013 CMP RO,(R3) ;CORRECT HI LIMIT?
1841 005312 101362 BHI 10$
1842 005314 020012 CMP RO,(R2) ;CORRECT LO LIMIT?
1843 005316 101760 BLOS 10$
1844 005320 010013 MOV RO,(R3) ;'MAXBA'
1845
1846 005322 023727 002054 037476 ST3: CMP MAXBA,#37476 ;BK MEMORY - CLOBBER XXDPT
1847 005330 002003 BGE 1$
1848 005332 012737 037476 002054 MOV #37476,MAXBA ;BUT SAVE LOADER
1849 005340 105737 001253 1$: TST FRSTRT ;PROGRAM RESTARTED AT 210"
1850 005344 001402 BEQ BCTST ;NO
1851 005346 000137 007772 JMP EXRC5R ;YES, SKIP TEST 1 TO 7

```

007

```

1852
1853
1854
1855
1856 005352 012737 001306 001476
1857 005360 013737 001264 001500
1858 005366 017737 174104 001502
1859 005374 062737 000002 001476
1860 005402 005337 001500
1861 005406 100002
1862 005410 000137 007772
  
```

```

;THIS IS THE BEGINING OF THE CONSTRAINED TESTS AIMED AT CHECKING THE
;DIFFERENT BOUNDARY CONDITIONS OF RK11/RK05

;FIND OUT THE DRIVE NUMBER TO BE TESTED.
ECTST: MOV #KEY,DRVPTR ;INITIALIZE PTR TO DRV#
MOV DRVPRS,DRVCNT ;NUMBER OF DRIVES PRESENT
NXTDRV: MOV @DRVPTR,@DRV ;SAVE DRIVE # (BITS 15-13)
ADD #2,DRVPTR ;INCRMENT PTR TO NXT DRV#
DEC DRVCNT ;DONE ALL DRIVES?
BPL TST1 ;NO, GO TEST THIS DRIVE
JMP EXRCR ;ALL DONE, GO TO EXERCISER PART
  
```



1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872 005414 000004  
1873  
1874 005416 013701 001502  
1875 005422 010102  
1876 005424 062702 000002  
1877  
1878 005430 012737 005436 001110  
1879  
1880 005436 104416  
1881 005440 104420  
1882 005442 104416  
1883 005444 012737 111111 032514  
1884 005452 012703 000401  
1885 005456 004737 006304  
1886  
1887 005462 104101  
1888 005464 004737 020112  
1889 005470 104102  
1890 005472 004737 020126  
1891 005476 104103  
1892  
1893 005500 004737 020230  
1894 005504 104105  
1895  
1896 005506 032701 000010  
1897 005512 001005  
1898 005514 062701 000012  
1899 005520 062702 000016  
1900 005524 000747

```
*****  
;TEST 1 PERFORM WRITE OF 401 WORDS (1 SECTOR + 1 WORDS)  
;THIS TEST PERFORMS A WRITE OF 401 WORDS (1 SECTOR + 1WORD) AND  
;CHECKS IF RKDA,RKBA,RKWC INCREMENTED CORRECTLY.WRITING IS DONE  
;ON CYLINDER 0, SURFACE 0, SECTORS 0,1 AND 10,11. IT SHOULD BE  
;NOTED THAT THIS IS A BOUNDARY CONDITION TRANSFER. THE VALIDITY  
;OF THE TRANSFER IS CHECKED IN THE NEXT TEST.  
;DATA PATTERN WRITTEN IS 111111.  
*****  
†ST1: SCOPE  
MOV QDRV,R1 ;GET RKDA  
MOV R1,R2 ;SAVE RKDA  
ADD #2,R2 ;EXPCD RKDA AFTER WRITE IS DONE  
MOV #1$, $LPERR ;RETURN ADDRESS FOR LUPING  
1$: CON.RESET  
DRV.RESET  
CON.RESET  
2$: MOV #111111, DBUF ;CLEAR MASK BITS IN POLLING LOGIC  
;PATTERN TO BE WRITTEN  
MOV #401, R3 ;WORD COUNT FOR WRITE  
JSR PC, DOWRITE ;GO DO WRITE  
ERROR 101 ;INTERRUPT DID NOT OCCUR AFTER WRITE  
JSR PC, CHKCS ;CHECK ERROR BIT IN RKCS  
ERROR 102 ;ERROR BIT IN RKCS SET ON DOING WRITE  
JSR PC, CHKDA ;CHECK IF RKDA INCREMENTED RIGHT  
ERROR 103 ;RKDA DID NOT INCREMENT RIGHT AFTER  
;A WRITE OF 401 WORDS.  
JSR PC, CHKWC ;CHECK IF RKWC OVERFLOWED TO 0  
ERROR 105 ;RKWC DID NOT OVERFLOW TO 0 AFTER  
;A WRITE OF 401 WORDS.  
BIT #10, R1 ;SECTORS 10,11 WRITTEN?  
BNE TST2 ;YES  
ADD #12, R1 ;RKDA TO BE USED NEXT (SEC 10)  
ADD #16, R2 ;EXPCD RKDA AFTER WRITE IS DONE  
BR 2$ ;GO WRITE SECS 10,11
```

1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908 005526 000004  
1909  
1910 005530 013701 001502  
1911 005534 012737 005542 001110  
1912 005542 104416  
1913 005544 104420  
1914 005546 104416  
1915  
1916  
1917 005550 004737 006434  
1918  
1919 005554 012703 001000  
1920  
1921 005560 004737 006424  
1922  
1923 005564 104101  
1924  
1925 005566 004737 020112  
1926 005572 104102  
1927  
1928 005574 012704 032514  
1929 005600 010402  
1930 005602 004737 020150  
1931 005606 104104  
1932  
1933 005610 012705 177764  
1934 005614 022712 111111  
1935 005620 001410  
1936 005622 012737 111111 001164  
1937 005630 004737 005720  
1938 005634 104100  
1939  
1940  
1941  
1942  
1943  
1944 005636 005205  
1945 005640 001421  
1946 005642 005722  
1947 005644 020227 033516  
1948 005650 001361  
1949  
1950 005652 005712  
1951 005654 001407  
1952 005656 005037 001164  
1953 005662 004737 005720  
1954 005666 104100  
1955  
1956

```
*****
;TEST 2 READ & CHECK THAT 401 WORD WRITE WAS DONE CORRECTLY
;THIS TEST PERFORMS A READ OF THE 401 WORDS WRITTEN IN THE
;PREVIOUS TEST AND CHECKS THAT THEY WERE CORRECTLY READ. MOREOVER
;IT CHECKS THAT ONLY ONE NON-ZERO WORD (401TH) WAS WRITTEN IN THE
;SECOND SECTOR AND THE REST OF THE WORDS ARE ALL ZEROS.
*****
↑ST2: SCOPE

MOV QDRV R1 ;GET DRIVE #
MOV #1$, $LPERR ;ADDRESS FOR LUPING ON EROR
1$: CON.RESET
DRV.RESET
CON.RESET

JSR PC,CLEANBUF ;CLEAN UP THE DATA BUFFER
;INTO WHICH READ WILL
;BE DONE
;WORD COUNT
MOV #1000,R3

JSR PC,DOREAD ;GO DO A READ OF 2 SECTORS
;FROM DISK ADDRESS GIVEN IN R1
;INTERRUPT DID NOT OCCUR AFTER
;READ OF 401 WORDS WAS DONE.
ERROR 101 ;CHECK IF ERROR BIT IN RKCS SET?
;ERROR BIT IN RKCS SET ON DOING A
;READ OF 401 WORDS.
JSR PC,CHKCS ;STARTING BUS ADDRESS, INTO WHICH READ
;WAS DONE
ERROR 102 ;CHECK IF RKBA INCREMENTED RIGHT
;RKBA DID NOT INCREMENT RIGHT AFTER READ
;OF 401 WORDS.
MOV #-14,R5 ;ALLOW 12 ERRORS, AT THE MOST
CMP #111111,(R2) ;CORRECT DATA READ?
BEQ 3$ ;YES
MOV #111111,$REG1 ;GET EXPCTD DATA WORD
JSR PC,ERINF1 ;GET ERROR INFORMATION
ERROR 100 ;DATA ERROR OCCURRED WHEN A
;READ OF 401 WORDS WAS DONE
;THE DISK ADDRESS FROM WHERE
;THE DATA WAS READ INCORRECTLY
;IS GIVEN IN THE ERROR MESSAGE

INC R5 ;REPORT 12 ERORS AT MOST
BEQ 6$
TST (R2)+ ;INCREMENT POINTER
CMP R2,#DBUF+1002 ;CHECKED ALL 401 WORDS?
BNE 2$

4$: TST (R2) ;CHECK THAT REST OF 377 WORDS
;ARE ALL 0'S
BEQ 5$ ;GET EXPCTD DATA WORD (0)
CLR $REG1 ;GET ERROR INFO
JSR PC,ERINF1 ;DATA ERROR. IN A PREVIOUS
;TEST A WRITE OF 401 WORDS
; (1 SECTOR + 1 WORD) WAS DONE
ERROR 100
```

E04

```

1957
1958
1959
1960
1961
1962
1963
1964
1965
1966 005670 005205
1967 005672 001404
1968 005674 005722
1969 005676 020227 034514
1970 005702 001363
1971
1972 005704 032701 000010
1973 005710 001030
1974 005712 062701 000012
1975 005716 000711
1976
1977
1978
1979
1980
1981
1982
1983 005720 010237 001162
1984 005724 011237 001166
1985 005730 010146
1986 005732 020227 033512
1987 005736 003001
1988 005740 005316
1989 005742 005216
1990 005744 032716 000010
1991 005750 001405
1992 005752 032716 000004
1993 005756 001402
1994 005760 062716 000004
1995 005764 012637 001170
1996 005770 000207
  
```

```

;NOW THESE 2 SECTORS WERE
;READ IN THE SECOND SECTOR
;THE FIRST WORD IS A NON-ZERO
;WORD (WHICH WAS WRITTEN BEFORE)
;THE REST OF 377 WORD
;SHOULD BE ALL ZEROS, IF
;THE WRITE WAS DONE CORRECTLY
;(& READ IS DONE CORRECTLY)

;REPORT 12 ERRORS AT MOST
5$: INC R5
   BEQ 6$
   TST (R2)+
   CMP R2,#DBUF+2000
   BNE 4$
;ALL WORDS CHECKED?
;IF NOT GO BAK

6$: BIT #10,R1
   BNE TST3
   ADD #12,R1
   BR 1$
;WERE SECTORS 10,11 READ
;YES
;FROM NEW RKDA, SEC 10
;GO BACK AND READ FROM SECS 10,11

;ERINF1
;AT THE TIME OF ENTRY:
;R2 CONTAINS ERRORING BUS ADDRESS (WHERE DATA ERROR OCCURRED).
;(R2) CONTAINS BAD DATA THAT WAS READ BACK FROM DISK.
;R1 CONTAINS DISK ADDRESS WHERE READ BEGAN.

ERINF1: MOV R2,$REG0 ;GET BUS ADDRESS OF DATA ERROR
        MOV (R2,$REG2) ;GET BAD DATA WORD (READ)
        MOV R1,-(SP)
        CMP R2,#DBUF+776 ;FIGURE OUT THE DISK ADDRESS
        BGT 1$ ;WHERE DATA ERROR OCCURRED
        DEC (SP)
1$: INC (SP)
   BIT #10,(SP)
   BEQ 2$
   BIT #4,(SP)
   BEQ 2$
   ADD #4,(SP)
2$: MOV (SP)+,$REG3
   RTS PC
  
```

```

1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008 005772 000004
2009 005774 013701 001502
2010 006000 012737 006020 001110
2011 006006 010102
2012 006010 062702 000021
2013 006014 012703 006001
2014 006020 104416
2015 006022 104420
2016 006024 104416
2017
2018
2019 006026 012737 044444 032514
2020
2021 006034 004737 006304
2022
2023 006040 104101
2024
2025 006042 004737 020112
2026 006046 104102
2027 006050 004737 020126
2028 006054 104103
2029
2030
2031 006056 004737 020230
2032 006062 104105
2033 006064 032701 000020
2034 006070 001006
2035 006072 010201
2036 006074 062702 000020
2037 006100 012703 005401
2038 006104 000745
  
```

```

*****
;TEST 3 PERFORM WRITE OF 12 SECTORS + 1 WORD
;THIS TEST CHECKS FOR ANOTHER BOUNDARY CONDITION. IT
;PERFORMS A WRITE OF 12 SECTORS + 1 WORD. RKDA RKBA
;RKWC ARE CHECKED TO SEE IF THEY ARE INCREMENTED CORRECTLY.
;VALIDITY OF THE DATA WRITTEN IS CHECKED IN THE NEXT
;TEST. DATA IS WRITTEN ON SECTORS 0-11, SURFACE 0
;CYLINDER 0 (6001TH WORD ON SECTOR 0, SURFACE 1, CYL 0).
;ALSO ON SECTORS 0-11, SURFACE 1 (6001TH WORD ON SECTOR
;0, CYL 1)
*****
TST3: SCOPE
MOV QDRV,R1 ;GET DRIVE #
MOV #15,$LPERR ;LUP ON ERROR TO '15'
MOV R1,R2
ADD #21,R2
MOV #6001,R3
1$: CON.RESET
DRV.RESET
CON.RESET

MOV #44444,DBUF ;PATTERN TO BE WRITTEN
JSR PC,DOWRITE ;GO DO WRITE
ERROR 101 ;INTERUPT DID NOT OCCUR ON
;COMPLETION OF WRITE
JSR PC,CHKCS ;CHECK IF EROR BIT IN RKCS SET
ERROR 102 ;EROR BIT IN RKCS SET ON DOING WRITE
JSR PC,CHKDA ;CHECK IF RKDA INCREMENTED RIGHT
ERROR 103 ;RKDA DID NOT INCREMENT CORRECTLY
;AFTER A WRITE OF 6001 (OCTAL) WORDS.
;(12 SECTORS + 1)
JSR PC,CHKWC ;CHECK IF RKWC OVERFLOWED TO 0
ERROR 105 ;RKWC DID NOT OVERFLOW TO 0
BIT #20,R1 ;WRITTEN ON SURFACE 1?
BNE TST4 ;YES
MOV R2,R1
ADD #20,R2 ;SURFACE 1
MOV #5401,R3 ;WORD COUNT
BR 1$ ;GO WRITE SURFACE 1
  
```

2039  
 2040  
 2041  
 2042  
 2043  
 2044  
 2045  
 2046  
 2047  
 2048  
 2049  
 2050  
 2051  
 2052  
 2053  
 2054  
 2055  
 2056  
 2057  
 2058  
 2059  
 2060  
 2061  
 2062  
 2063  
 2064  
 2065  
 2066  
 2067  
 2068  
 2069  
 2070  
 2071  
 2072  
 2073  
 2074  
 2075  
 2076  
 2077  
 2078  
 2079  
 2080  
 2081  
 2082  
 2083  
 2084  
 2085  
 2086  
 2087  
 2088  
 2089  
 2090  
 2091  
 2092  
 2093  
 2094

006106	000004		
006110	013701	001502	
006114	062701	000013	
006120	012737	006126	001110
006126	104416		
006130	104420		
006132	104416		
006134	004737	006434	
006140	012703	001000	
006144	004737	006424	
006150	104101		
006152	004737	020112	
006156	104102		
006160	012704	032514	
006164	010402		
006166	004737	020150	
006172	104104		
006174	012705	177764	
006200	022712	044444	2\$:
006204	001410		
006206	012737	044444	001164
006214	004737	005720	
006220	104100		
006222	005205		
006224	001421		
006226	005722		3\$:
006230	020227	033516	
006234	001361		
006236	005712		4\$:
006240	001407		
006242	005037	001164	
006246	004737	005720	
006252	104100		

```

*****
*TEST 4 READ & CHECK THAT 6001 WORD WRITE WAS DONE CORRECTLY
;THIS TEST CHECKS THAT THE 6001-WORD WRITE THAT WAS DONE IN THE
;PREVIOUS TEST WAS CORRECT, ESPECIALLY THE LAST 401 WORDS. THE
;FIRST WORD OF THE SECTOR ( IN WHICH THE 6001TH WORD) IS WRITTEN
;IS THE ONLY NON-ZERO WORD IN THAT SECTOR, THE REST 377 WORDS ARE
;ALL ZEROS, IF THE WRITING WAS DONE CORRECTLY.
*****
1$T4: SCOPE
MOV QDRV,R1 ;GET DRIVE #
ADD #13,R1 ;DISK ADDRESS FROM WHERE READ IS DONE
MOV #15,$LPERR
1$: CON.RESET
DRV.RESET
CON.RESET

JSR PC,CLEANBUF ;CLEAN UP THE BUFFER INTO WHICH
;READ WILL BE DONE
;SET UP RKDA
;SECTOR 11, SURFACE 0
;WORD COUNT

MOV #1000,R3

JSR PC,DOREAD ;GO READ 1000 WORDS (2 SECS)
ERROR 101 ;INTERRUPT DID NOT OCCUR AFTER
;COMPLETION OF READ
;CHECK IF EROR BIT IN RKCS SET
JSR PC,CHKCS ;EROR (RKCS) SET ON DOING READ
ERROR 102 ;STARTING BA OF DATA BUFR
MOV #DBUF,R4
MOV R4,R2
JSR PC,CHKBA ;RKBA INCREMENTED CORRECTLY?
ERROR 104 ;RKBA DID NOT INCREMENT CORRECTLY

MOV #14,R5
2$: CMP #44444,(R2) ;DATA WORD OK?
BEQ 3$
MOV #44444,$REG1
JSR PC,ERINF1
ERROR 100
;NO. GET EXPCTD DATA WORD
;GET OTHER ERROR INFO
;DATA ERROR. A WRITE OF 6001
;WORDS (12 SECS + 1 WORD) WAS DONE
;IN A PREVIOUS TEST. THE LAST TWO
;SECTORS (LAST 401 WORDS) WERE READ
;BACK. THIS ERROR INDICATES THAT
;SEC #11 (LAST BUT ONE SECTOR) GAVE
;BAD DATA WORDS
;REPORT 12 ERORS AT MOST

INC R5
BEQ 6$
3$: TST (R2)+ ;INCREMENT POINTER TO BA
CMP R2,#DBUF+1002 ;CHECKED 401 WORDS?
BNE 2$

4$: TST (R2) ;CHECK THAT THE REMAINING 377
BEQ 5$ ;WORDS OF THE LAST SECTOR (SEC #0)
CLR $REG1 ;WERE READ BACK AS 0'S
JSR PC,ERINF1
ERROR 100 ;DATA ERROR. IF WRITE WAS DONE CORRECTLY
;IN THE PREVIOUS TEST, THE LAST SECTOR
;OF THE DATA BLOCK (12 SECS + 1 WORD)
;SHOULD CONTAIN ONLY 1 (FIRST) WORD

```



```

2109      :DOWRITE
2110      :THIS ROUTINE PERFORMS A WRITE ON A DISK AT THE TIME OF ENTRY. R1 CONTAINS
2111      :DISK ADDRESS (RKDA) WHERE WRITE IS TO BE DONE, R3 CONTAINS THE WORD COUNT
2112      : (RKWC), 'DBUF' CONTAINS THE DATA TO BE WRITTEN. NOTE IBA BIT IS SET.
2113
2114      :WRITE IS DONE IN INTERRUPT MODE. IF THE INTERRUPT DOES NOT OCCUR WITHIN
2115      :A CERTAIN TIME, RETURN IS MADE TO THE ERROR MESSAGE FOLLOWING THE 'JSR'
2116      :CALL. IF THE INTERRUPT OCCURS, RETURN ADDRESS IS ADJUSTED TO SKIP OVER
2117      :THE ERROR MESSAGE.
2118
2119 006304 012777 004002 172710 DOWRITE: MOV      #4002,RKCS      ;WRITE IBA
2120 006312 010177 172712      DOXFER: MOV      R1,RKDA      ;ADDRESS THE DRIVE
2121 006316 010377 172702      MOV      R3,RKWC      ;XFER THIS # OF WORDS
2122 006322 005477 172676      NEG      RKWC
2123 006326 012777 032514 172672      MOV      #DBUF,RKBA    ;USE THIS BUS ADDRESS
2124 006334 012777 006414 172676      MOV      #3,RKVEC      ;SET UP INTERRUPT VECTOR
2125 006342 005046      CLR      -(SP)         ;NEW PSW
2126 006344 012746 006352      MOV      #1,-(SP)      ;SET NEW PC TO STACK *****
2127 006350 000002      RTI
2128 006352 052777 000101 172642 1$:  BIS      #101,RKCS     ;SET IDE, GO (WRITE,IBA/ READ
2129 006360 005037 001466      CLR      CICNT
2130 006364 012737 177760 001470      MOV      #-20,CICNT1
2131 006372 005237 001466      2$:  INC      CICNT
2132 006376 001375      BNE      -4
2133
2134 006400 005237 001470      INC      CICNT1
2135 006404 001372      BNE      2$
2136
2137 006406 004737 022032      JSR      PC,GT4RG      ;TIMED OUT, INTERRUPT DID NOT OCCUR
2138 006412 000207      RTS      PC            ;RETURN TO THE EROR MEASGE
2139
2140 006414 022626      3$:  CMP      (SP)+,(SP)+   ;RESTORE STACK POINTER
2141 006416 062716 000002      ADD      #2,(SP)       ;ADJUST RETURN ADDRESS TO SKIP OVER
2142 006422 000207      RTS      PC            ;EROR MESSAGE ON RETURN

```

2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168

006424 012777 000004 172570  
006432 000727  
  
006434 012702 177000  
006440 012705 032514  
006444 012725 022222  
006450 005202  
  
006452 001374  
006454 000207

; THIS ROUTINE PERFORMS A READ ON THE DISK AT THE TIME OF ENTRY R1 CONTAINS  
; THE DISK ADDRESS FROM WHERE THE READ IS TO BE DONE. R3 CONTAINS THE WORD  
; COUNT (RKWC). READ WILL BE DONE INTO DATA BUFFER AT 'DBUF'.  
  
; READ IS DONE IN INTERRUPT MODE. IF THE INTERRUPT DOES NOT OCCUR WITHIN  
; A CERTAIN TIME, RETURN IS MADE TO THE ERROR MESSAGE FOLLOWING THE 'JSR'  
; CALL. IF THE INTERRUPT OCCURS AS EXPECTED, RETURN ADDRESS IS ADJUSTED  
; TO SKIP OVER THE ERROR MESSAGE.  
  
DOREAD: MOV #4, DRKCS ; READ  
BR DOXFER  
  
; CLEANBUF  
; CLEANS OUT THE DATA BUFFER (ALL WORDS WRITTEN TO 177777) INTO WHICH THE  
; READ FROM THE DISK WILL BE DONE. DATA BUFFER STARTS AT 'DBUF' AND IS  
; 1000 (OCTAL) WORDS LONG.  
  
CLEANBUF: MOV #-1000, R2 ; SET COUNT  
MOV #DBUF, R5 ; INITIALIZE BA  
1\$: MOV #2222, (R5)+  
INC R2 ; DONE ALL WORDS?  
; BUFFER STARTING AT (PHYSICAL, BUS  
; ADDRESS 177000 (177000-200776).  
  
BNE 1\$  
RTS PC ; YES RETURN



1.04

2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224

006456	000004		
006460	023727	017772	002000
006466	103002		
006470	000137	007040	
006474	012737	001600	172352
006502	012737	002000	172354
006510	012737	000001	177572
006516	012737	006524	001110
006524	104416		
006526	104420		
006530	012700	000001	
006534	012701	137000	
006540	010021		
006542	005200		
006544	020027	001001	
006550	001373		
006552	013777	001502	172450
006560	012777	177000	172436
006566	012777	177000	172432
006574	012777	000003	172420
006602	104417		
006604	004737	020112	
006610	104102		
006612	004737	020204	
006616	104106		

```
*****
:TEST 5 CHECK DATA TRANSFER AROUND 32K BOUNDARY
: *THIS TEST PERFORMS A WRITE OF 2 SECTORS ON THE DISK FROM MEMORY
: *LOCATIONS AROUND THE 32K BOUNDARY, SECTORS 0,1, CYL 0, SURFACE
: *0 ARE WRITTEN. PHYSICAL BUS ADDRESSES FOR THE DATA BUFFER:
: * 177000 TO 200776 I.E. (32K-256) TO (32K+255)
:
: *CHECKING IS DONE TO SEE IF MEX BITS, RKBA,RKDA,RKWC INCREMENTED
: *CORRECTLY. THEN DATA BUFFER IS CLEARED OUT AND A READ IS DONE
: *INTO IT. A CHECK IS MADE TO SEE IF THE CORRECT DATA WAS RECIEVED.
: *ONLY 12 DATA ERRORS ARE REPORTED.
*****
†ST5: SCOPE
CMP $LSTBK, #2000 ;33K OR MORE OF MEMORY?
BHIS 1$ ;YES
JMP TST6 ;IF NOT, DONT DO THIS TEST
1$: MOV #1600, #KIPARS ;MAP 28-32K THRU PAR 5
MOV #2000, #KIPAR6 ;MAP 32-36K THRU PAR 6
MOV #1, #SRO ;TURN ON MEMORY MANAGEMENT
:
:SET UP DATA BUFFER (1000 OCTAL WORDS LONG) FOR WRITING TWO SECTORS
:ON THE DISK. THE TRANSFER IS DONE AROUND THE 32K BOUNDARY FROM
:BUS ADDRESS (PHYSICAL) 177000 TO 200776, (32K-256) TO (32+256)
:
:DATA IN THE BUFFER IS A COUNT PATTERN STARTING FROM 1 FOR THE FIRST
:WORD TO 000 FOR THE LAST WORD.
:PHYSICAL BUFFER ADDRESS: 177000 TO 200776 (32K-256 TO 32K+256)
2$: MOV #2$, $LPERR ;LUP TO 2$ ON EROR (SW 9)
CON.RESET
DRV.RESET
MOV #1, R0 ;INITIALIZE DATA PATTERN TO BE
;WRITTEN
3$: MOV #137000, R1 ;BA TO START PHYSICAL ADDRESS=177000
MOV R0, (R1) ;WRITE COUNT PATTERN (1-1000)
INC R0 ;INTO DATA BUFFER (PHYS ADDRES
CMP R0, #1001 ;177000 TO 200776)
BNE 3$
MOV QDRV, #RKDA ;SUR 0, SEC 0, CYL 0
MOV #-1000, #RKWC ;WORD COUNT =2 SECS
MOV #177000, #RKBA ;BUS ADDRESS.
MOV #3, #RKCS ;WRITE GO
CON.RDY ;WAIT FOR CNTROL RDY
JSR PC, #CHKCS ;ANY EROR IN RKCS?
ERROR 102 ;'ERR' SET IN RKCS, ON DOING A
;WRITE OF SECTORS (0,1) FROM
;(PHYSICAL) BUS ADDRESS 177000
;(177000 TO 200776)
JSR PC, #CHKMEX ;CHECK THAT RKBA OVERFLOWED INTO
;EXTENDED MEM. BIT (0) OF RKCS (BIT 4)
;EX MEM BIT 0 SET?
;GET RKCS, ER, DS, DA
;MEX BITS INCORRECT. BIT 4 OF RKCS
; (MEX BIT 0) SHOULD HAVE SET
```



M04

```

2281                                     ;AND BIT 4 OF RKCS (MEX) WILL BE SET.
2282
2283 006750 012705 1777641 6$: MOV #14,R5 ;REPORT ONLY 12 ERRORS.
2284 006754 012702 137000 MOV #137000,R2 ;STARTING ADDRESS OF BUFFER
2285                                     ; (PA=177000)
2286 006760 012701 000001 MOV #1,R1 ;INITIALIZE DATA PATTERN
2287
2288 006764 020112 7$: CMP R1,(R2) ;CORRECT DATA WORD RECVD?
2289 006766 001414 BEQ #5 ;
2290 006770 005705 TST R5 ;REPORT THIS ERROR?
2291 006772 001420 BEQ #9$ ;
2292 006774 005205 INC R5 ;COUNT ERORS
2293
2294 006776 010137 001162 MOV R1,$REG0 ;GET EXPCTED DATA WORD
2295
2296 007002 011237 001164 MOV (R2),$REG1 ;GET DATA WORD RECVD
2297 007006 104107 ERROR 107 ;DATA COMPARISON ERROR ON DOING A
2298 ;READ OF 2 SECTORS (0,1,CYL 0,SURFACE 0
2299 ;INTO DATA BUFFER (PHYSICAL ADDRESS
2300 ;177000 TO 200776)
2301
2302 007010 104421 002214 TYPMSG MSG13 ;TYPE 'PHYSICAL BUS ADDRESS'
2303 007014 004737 017774 JSR PC,TYPDB0 ;TYPE THE 6 DIGIT PHYSICAL BUS ADDRESS
2304 ;WHERE THE DATA ERROR OCCURRED.
2305
2306 007020 062702 000002 8$: ADD #2,R2 ;INCREMENT POINTER TO BA
2307 007024 005201 INC R1 ;
2308 007026 022701 001001 CMP #1001,R1 ;CHECKED THE ENTIRE BUFFER?
2309 007032 001354 BNE #7$ ;
2310
2311                                     ;**OFF
2312 007034 005037 177572 9$: CLR #SRO ;TURN OFF MEMORY MANAGEMENT

```

N04

```

2313
2314
2315
2316
2317
2318
2319
2320
2321
2322 007040 000004
2323
2324 007042 023727 017772 001740
2325 007050 103002
2326 007052 000137 007374
2327 007056 012737 001600 172352
2328 007064 012737 000001 177572
2329
2330
2331
2332
2333 007072 012737 007100 001110
2334 007100 104416
2335 007102 104420
2336 007104 012700 000001
2337 007110 012701 120000
2338 007114 010021
2339 007116 005200
2340 007120 020027 010001
2341 007124 001373
2342
2343 007126 013777 001502 172074
2344 007134 012777 170000 172062
2345 007142 012777 160000 172056
2346 007150 012777 000003 172044
2347
2348 007156 104417
2349
2350 007160 004737 020112
2351 007164 104102
2352
2353
2354
2355
2356
2357
2358
2359 007166 004737 020204
2360
2361 007172 104106
2362
2363
2364
2365 007174 012703 010000
2366 007200 012704 160000
2367 007204 004737 020150
2368 007210 104104

```

```

*****
; *TEST 6 CHECK DATA TRANSFER FROM 28K TO 32K
; *THIS TEST DOES A WRITE OF 4K WORDS FROM A BUFFER LOCATED AT 28K
; *THEN THE DATA IS READ BACK INTO THE SAME BUFFER(28K-32K) AND IS
; *CHECKED TO SEE IF IT IS CORRECT. NOTE THAT THE BUFFER IS FILLED
; *WITH ALL 1'S BEFORE DOING THE READ.
; *THE WRITE IS DONE STARTING AT CYLINDER 0, SECTOR 0, SURFACE 0.
*****
†T6: SCOPE
CMP SLSTBK, #1740 ;32K OR MORE OF MEMORY?
BHIS 15 ;YES
JMP TST7 ;IF NOT, DONT DO THIS TEST
1$: MOV #1600, @#KIPARS ;MAP 28-32K THRU PAR 5
MOV #1, @#SR0 ;TURN ON MEM MANAGEMENT
;WRITE A COUNT PATTERN (1-10000) INTO DATA BUFFER (PHYSICAL ADDRESS
;160000-177776). THIS DATA BUFFER WILL BE USED FOR WRITING ON THE DISK.
2$: MOV #25, $LPERR ;LUP TO 2$ ON EROR
CON.RESET
DRV.RESET
MOV #1, R0 ;INITIALIZE DATA PATTERN TO BE WRITTEN
MOV #120000, R1 ;INITIALIZE BA (PA=160000) 28K
3$: MOV R0, (R1)+ ;WRITE COUNT PATTERNS (1-10000)
INC R0 ;INTO DATA BUFFER (PA 160000 TO
CMP R0, #10001 ;177776, 28K-32K)
BNE 3$
MOV @DRV, @RKDA ;WRITE ON SEC 0, CYL 0, SUR1
MOV #-10000, @RKWC ;4K WORDS
MOV #160000, @RKBA ;FROM THIS BUS ADDRESS
MOV #3, @RKCS ;WRITE, GO
CON.RDY ;WAIT FOR CONTROL READY
JSR PC, CHKCS ;ANY ERROR IN RKCS?
ERROR 102 ;'ERR' BIT SET IN RKCS ON DOING
;A WRITE OF 4K WORDS FROM 160000
;(28K-32K). DISK WRITE BEGAN ON
;SEC 0, CYL 0, SUR 0. IF ALL 4K
;WORDS WERE WRITTEN RKBA SHOULD OVERFLOW
;AND CONTAIN 0. BIT 4 OF RKCS
;(MEX BIT) SHOULD BE 1
JSR PC, CHKMEX ;CHECK IF RKBA OVERFLOWED AND MEX
;BITS (4) IN RKCS WAS SET.
ERROR 106 ;MEX BIT 4 NOT SET AFTER OVERFLOW OF
;RKBA. WRITE OF 4K WORDS (28K-32K) WAS DONE.
;RETURN HERE IF NO ERROR
MOV #10000, R3 ;WORD COUNT
MOV #160000, R4 ;STARTING BUS ADDRESS
JSR PC, CHKBA ;CHECK IF RKBA INCREMENTED CORRECTLY
ERROR 104 ;RKBA DID NOT OVERFLOW TO AFTER A WRITE IF 4K

```

BUS

```

2369                                     ;WORDS (160000 TO 177776)
2370
2371 007212 004737 020230 JSR PC,CHKWC ;CHECK RKWC OVERFLOWED CORRECTLY
2372 007216 104105 ERROR 105 ;RKWC DID NOT OVEFLOW TO 0
2373 ;AFTER A WRITE OF 4K WORD (160000
2374 ;TO 177776)
2375
2376
2377 007220 012737 007226 001110 MOV #45,$LPERR ;RETURN ADDRESS FOR LUPING
2378 007226 104416 45: CON.RESET
2379 007230 104420 DRV.RESET
2380
2381 007232 012701 120000 MOV #120000,R1 ;CLEAR THE DATA BUFFER BEFORE
2382 007236 005021 55: CLR (R1)+ ;DOING A READ INTO IT
2383 007240 022701 140000 CMP #140000,R1
2384 007244 001374 BNE 55
2385
2386 007246 013777 001502 171754 MOV QDRV,ARKDA ;READ FROM SEC 0, SUR 0, CYL 0
2387 007254 012777 170000 171742 MOV #10000,ARKWC ;4K WORDS
2388 007262 012777 160000 171736 MOV #160000,ARKBA ;INTO THIS BUS ADDRESS
2389
2390 007270 012777 000005 171724 MOV #5,ARKCS ;READ, GO
2391 007276 104417 CON.RDY ;WAIT FOR CNTROL RDY
2392
2393 007300 004737 020112 JSR PC,CHKCS ;ANY ERROR IN RKCS?
2394 007304 104102 ERROR 102 ;ERROR BIT SET IN RKCS ON DOING
2395 ;A READ OF 4K WORDS INTO MEMORY
2396 ; (160000-177776)
2397 007306 012705 177764 MOV #14,R5 ;REPORT 12 ERRORS AT MOST
2398 007312 012702 120000 MOV #120000,R2 ;STARTING ADDRESS OF BUFFER
2399 ; (PA=160000)
2400 007316 012701 000001 MOV #1,R1 ;INITIALIZE DATA PATTERN
2401
2402 007322 020112 65: CMP R1,(R2) ;CORRECT DATA WORD?
2403 007324 001413 BEQ 75
2404 007326 010137 001162 MOV R1,$REG0 ;GET EXPCTD DATA WORD
2405 007332 011237 001164 MOV (R2),$REG1 ;GET DATA WORD RECVD
2406 007336 104107 ERROR 107 ;DATA ERROR ON DOING A READ OF
2407 ;4K WORDS STARTING FROM SEC 0,
2408 ;CYL 0, SUR 0 INTO MEMORY (160000-
2409 ;177776)
2410 007340 104421 002214 TYPMSG MSG13 ;TYPE 'PHYSICAL BUS ADDRESS'
2411 007344 004737 017774 JSR PC,TYPDBO ;TYPE OUT THE PHYSICAL BUS ADDRESS
2412 ;WHERE DATA
2413 007350 005205 INC R5 ;ERROR OCCURRED. R2 CONTAINS
2414 007352 001406 BEQ 85 ;THE VIRTUAL ADDRESS
2415
2416 007354 062702 000002 75: ADD #2,R2 ;POINTER TO NEXT BA
2417 007360 005201 INC R1 ;NEXT PATTERN
2418 007362 022701 001000 CMP #1000,R1 ;ALL DONE?
2419 007366 001355 BNE 65 ;NO
2420
2421 007370 005037 177572 85: CLR #SRO ;TURN OFF MEM MANAGEMENT

```

CU5

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

```
*****  
:TEST 7 PERFORM THE LARGEST POSSIBLE DATA TRANSFER  
:THIS TEST PERFORMS THE LARGEST DATA TRANSFER POSSIBLE  
:WITHIN AVAILABLE MEMORY.  
  
:1) IF KT11 IS PRESENT A WRITE OF 16K WORDS IS DONE ON  
:THE DISK (PROVIDED 29K OR MORE IS PRESENT).  
  
:2) IF KT11 IS NOT PRESENT THEN ALL BUT TOP 1.5K OF MEMORY WILL BE USED  
:FOR WRITING (RXDP MONITOR).  
  
:ALL DATA TRANSFERS ARE DONE STARTING AT BA 'PGEND'.  
:ALL WRITES ARE DONE BEGINNING AT SECTOR 0, CYLINDER 0,  
:SURFACE 0.  
  
:AFTER DOING THE 'WRITE' A 'WRITE CHECK' IS DONE  
:TO SEE IF THE WRITE WAS DONE CORRECTLY. ANY WRITE  
:CHECK ERROR IS REPORTED.  
*****
```

```
007374 000004  
007376 005737 017524  
007402 100004  
  
007404 023727 017772 001540  
007412 002034  
  
007414 013703 002054  
007420 162703 032514  
007424 000241  
007426 006003  
  
007430 005001  
007432 010346  
007434 005046  
007436 012746 000400  
007442 004737 025212  
007446 005726  
007450 001401  
007452 005201  
  
007454 062601  
  
007456 005002  
007460 162701 000014  
007464 100403  
007466 062702 000020  
007472 000772  
007474 062701 000014  
007500 050102  
007502 000404  
  
007504 012703 040000  
007510 012702 000124
```

```
†ST7: SCOPE  
TST SKT11 ;MEMORY MANAGEMENT PRESENT?  
BPL IS ;NO  
;YES  
CMP SLSTBK, #1540 ;28K OR MORE?  
BGE XFR16K ;YES  
  
1$: MOV MAXBA, R3 ;FIGURE OUT THE MAXIMUM  
SUB #PGEND, R3 ;XFER THAT CAN BE DONE  
CLC  
ROR R3 ;FROM 'PGEND' TO THE  
;'MAXBA'  
  
CLR R1  
MOV R3 -(SP) ;PUT DIVIDEND ON STACK (LO)  
CLR -(SP) ;(HIGH PART)  
MOV #400, -(SP) ;DIVISOR  
JSR PC, #SDIV ;GO TO DIVIDE ROUTINE  
TST (SP)+ ;POP OFF REMAINDER FROM STACK  
BEQ 2$  
INC R1  
  
2$: ADD (SP)+, R1 ;GET # OF SECTORS REQUIRED TO  
;DO WRITE OF # OF WORDS  
;(CONTAINED IN RD). R1 CONTAINS  
;# OF SECTORS  
CLR R2 ;FORM THE EXPECTED DISK  
SUB #14, R1 ;ADDRESS - AFTER THE WRITE  
BMI 4$ ;IS DONE.  
ADD #20, R2  
BR 3$  
4$: ADD #14, R1 ;R2 CONTAINS EXPCTD RKDA  
BIS R1, R2 ;AFTER WRITE IS DONE  
BR XFR  
  
XFR16K: MOV #40000, R3 ;# OF WORDS = 16K  
MOV #124, R2 ;RKDA SHOULD INCREMENT IC  
;THIS AFTER A TRANSFER OF 16K  
;WORDS STARTING AT DISK  
;ADDRESS = 0 (CYL 2, SUR 1, SEC 4
```

005

2478										
2479	007514	053702	001502		XFR:	BIS	QDRV,R2		:ADD THE DRIVE # TO	
2480									:EXPCD RKDA AFTER XFER	
2481	007520	012737	007526	001110		MOV	#15,\$LPERR		:LUP BAK TO '15' ON ERROR	
2482									: (SW 9)	
2483	007526	104416			15:	CON.RESET				
2484	007530	104420				DRV.RESET				
2485	007532	013777	001502	171470		MOV	QDRV,QRKDA		:ADDRESS THE DRIVE, CYL 0,SUR 0,	
2486									:SEC 0	
2487	007540	010377	171460			MOV	R3,QRKWC			
2488	007544	005477	171454			NEG	QRKWC		:WORD COUNT (IF MORE THAN	
2489									:28K IS AVAILABLE A TRANSFER	
2490									:OF 20K WILL BE DONE, IF	
2491									:LESS THAN 28K, THE	
2492									:LARGEST TRANSFER WITHIN THE	
2493									:AVAILABLE MEMORY WILL	
2494									:BE DONE, IN BOTH	
2495									:CASES, DATA TRANSFER	
2496									:WILL BE DONE STARTING	
2497									:AT 'PGEND'	
2498	007550	012777	032514	171450		MOV	#PGEND,QRKBA		:START WRITE FROM HERE	
2499	007556	012777	000003	171436		MOV	#3,QRKCS		:WRITE, GO	
2500										
2501	007564	104417				CON.RDY			:WAIT FOR CONTROL READY	
2502										
2503	007566	004737	020112			JSR	PC,CHKCS		:CHKCK RKCS FOR ERROR?	
2504	007572	104102				ERROR	102		:ERROR BIT SET IN RKCS ON	
2505									:DOING A LARGE 'WRITE'	
2506										
2507	007574	004737	020126			JSR	PC,CHKDA		:CHECK IF RKDA INCREMENTED CORRECTLY?	
2508	007600	104103				ERROR	103		:RKDA DID NOT INCREMENT	
2509									:CORRECTLY	
2510										
2511	007602	012704	032514			MOV	#PGEND,R4			
2512	007606	004737	020150			JSR	PC,CHKBA		:CHECK THAT RKBA INCREMENTED	
2513									:CORRECTLY?	
2514	007612	104104				ERROR	104		:RKBA DID NOT INCREMENT	
2515									:CORRECTLY	
2516										
2517	007614	004737	020230			JSR	PC,CHKWC		:CHECK THAT RKWC OVERFLOWED	
2518	007620	104105				ERROR	105		:RKWC DID NOT OVERFLOW TO	
2519									:ZERO AFTER A WRITE	
2520										
2521	007622	012737	007630	001110		MOV	#25,\$LPERR		:LUP BAK TO '25' ON EROR (SW 9)	
2522	007630	104416			25:	CON.RESET				
2523	007632	104420				DRV.RESET				
2524	007634	013777	001502	171366	35:	MOV	QDRV,QRKDA		:ADDRESS THE DRIVE, CYL 0,SEC 0,SUR 0	
2525	007642	010377	171356			MOV	R3,QRKWC			
2526	007646	005477	171352			NEG	QRKWC			
2527	007652	012777	032514	171346		MOV	#PGEND,QRKBA		:STARTING BA	
2528										
2529	007660	012777	000407	171334	45:	MOV	#407,QRKCS		:WRITE CHECK, SSE, GO	
2530										
2531	007666	104417				CON.RDY			:WAIT FOR CONTROL RDY	
2532										
2533	007670	004737	020112			JSR	PC,CHKCS		:ANY ERROR IN RKCS	

```

2534 007674 104102          ERROR 10?
2535
2536 007676 032777 040000 171316      BIT    #BIT14,DRKCS    ;SKIP CHECKING FOR WCE IF HE SET
2537 007704 001030          BNE    7$             ;WRITE CHECK ERROR?
2538 007706 032777 000001 171304 5$:    BIT    #WCE,DRKER     ;NO
2539 007714 001406          BEQ    6$
2540
2541 007716 004737 022032          JSR    PC,GT4RG       ;GET RKCS,ER,DS,DA
2542 007722 017737 171300 001166      MOV    DRKBA,$REG2
2543 007730 104110          ERROR 110           ;WRITE CHECK ERROR. RKDA GIVES THE DISK ADDRESS
2544                                     ;WHERE WCE OCCURRED. (NOTE THE SECTOR IN
2545                                     ;ERROR IS OBTAINED BY BACKING OFF 1 SECTOR).
2546                                     ;NOTE THAT THE DATA BUFFER WHICH WAS WRITE
2547                                     ;CHECKED IS NOT ON A SECTOR BOUNDARY
2548                                     ;(LIKE 256, 512 WORD ETC.), BUT IS SOME
2549                                     ;SECTORS & A FRACTION (LIKE 300 WORDS ETC.).
2550
2551 007732 005777 171266          6$:    TST    DRKWC
2552 007736 001413          BEQ    7$             ;WAS THE ENTIRE DATA BUFFER
2553                                     ;WRITE CHECKED?
2554 007740 017705 171260          MOV    DRKWC,R5
2555 007744 042705 177760          BIC    #177760,R5    ;FORM THE RKBA AND RKWC
2556 007750 006305          ASL    R5
2557 007752 160577 171250          SUB    R5,DRKBA      ;TO BE USED FOR DOING
2558                                     ;WRITE-CHECK IF THE REST
2559 007756 042777 000017 171240      BIC    #17,DRKWC     ;OF THE DATA BUFFER
2560
2561 007764 000735          BR     4$             ;GO DO WRT-CHK FOR
2562                                     ;REST OF THE BUFFER
2563
2564                                     ;GO CHECK THE REST OF THE DRIVES.
2565
2566 007766 000137 005366          7$:    JMP    NXTDRV
  
```



2567  
2568  
2569  
2570  
2571  
2572  
2573  
2574  
2575  
2576  
2577  
2578  
2579  
2580  
2581  
2582  
2583  
2584  
2585  
2586  
2587  
2588  
2589  
2590  
2591  
2592  
2593  
2594  
2595  
2596  
2597  
2598  
2599  
2600  
2601  
2602  
2603  
2604  
2605  
2606  
2607  
2608  
2609  
2610  
2611  
2612  
2613  
2614  
2615  
2616  
2617  
2618  
2619  
2620  
2621  
2622

007772 104416  
007774 012700 001306  
010000 005020  
010002 020027 001552  
010006 001374  
010010 105737 001253  
010014 001045  
  
010016 005020  
010020 020027 002032  
010024 001374  
  
010026 012700 001554  
010032 012701 177704  
010036 010120  
010040 010120  
010042 010120  
  
010044 012700 025730  
010050 012720 123456  
010054 012720 176543  
010060 012720 001201  
010064 012720 062465  
010070 012720 176105  
010074 012720 174532  
010100 012720 157650  
010104 012720 030753  
010110 012720 131547  
010114 012720 032070  
010120 012720 123456  
010124 012720 176543  
  
010130 005227 177777  
010134 001004  
010136 012737 000062 002056  
  
010144 000403  
010146 013737 001236 002056  
  
010154 004737 022656

.SBTTL EXERCISER PROGRAM  
  
EXRCR: CON.RESET  
1\$: MOV #KEY,RO  
CLR (RO)+  
CMP RO,#KWHR  
BNE 1\$  
TSTB FRSTRT  
BNE 3\$  
  
2\$: CLR (RO)+  
CMP RO,#PCMD  
BNE 2\$  
  
MOV #KWMIN,RO  
MOV #-60,R1  
MOV R1,(RO)+  
MOV R1,(RO)+  
MOV R1,(RO)+  
  
MOV #RSDRVL,RO  
MOV #123456,(RO)+  
MOV #176543,(RO)+  
MOV #1201,(RO)+  
MOV #62465,(RO)+  
MOV #176105,(RO)+  
MOV #174532,(RO)+  
MOV #157650,(RO)+  
MOV #30753,(RO)+  
MOV #131547,(RO)+  
MOV #32070,(RO)+  
MOV #123456,(RO)+  
MOV #176543,(RO)+  
  
3\$: INC #-1  
BNE 4\$  
MOV #50.,REPCNT  
  
4\$: BR 5\$  
MOV PCNTR,REPCNT  
  
5\$: JSR PC,CHOPRS

;BEGINING OF THE EXERCISER PART OF THE PROGRAM.  
;IF THE PROGRAM WAS RESTARTED AT 210, THEN THE STATISTICS COLLECTED  
;SO FAR WILL NOT BE CLEARED.  
  
;CLEAR UP THE LOCATIONS FROM  
;'KEY' TO 'KWHR' (EXCLUDED)  
  
;RESTARTED AT 210?  
;YES, SAVE THE STATISTICS COLLECTED  
;UPTILL NOW, DONT CLEAR THEM  
  
;CLEAR LOCATIONS FROM 'HECN'  
;TO 'PCMD' (EXCLUDED)  
  
;INITIALIZE COUNTS FOR MINS,  
;SECS, KW11  
  
;INITIALIZE PTR TO RANDOM NO. SEEDS  
;USED BY THE RANDOM NUMBER GENERATOR  
;SET UP RANDOM NUMBER SEEDS TO BE  
  
;FIRST PASS?  
;BRANCH IF NOT  
;SET UP SHORT REPITITION COUNT  
;FOR FIRST PASS.  
  
;REPETITION COUNT. WHEN THIS COUNT GOES  
;TO 0, IT'S CONSIDERED TO BE AN END OF PASS.  
;THE NEXT PASS WILL START WILL START RIGHT  
;FROM WHERE THE EXERCISER LEFT OFF.  
  
;CHECK IF ANY DRIVES ARE PRESENT  
;IF NONE, GO TO \$EOP. END OF PASS

605

2623										
2624	010160	012746	177777		MOV	#177777,-(SP)		:	PUT LOW DIVIDEND ON STACK	
2625	010164	005046			CLR	-(SP)		:	CLEAR HIGH DIVIDEND AND PUSH	
2626								:	IT ON STACK	
2627	010166	013746	001264		MOV	DRVPRS,-(SP)		:	PUSH DIVISOR ON STACK	
2628	010172	004737	025212		JSR	PC,@#SDIV		:	GO TO THE 'DIVIDE' SUBROUTINE	
2629	010176	005726			TST	(SP)+		:	DISCARD THE REMAINDER. QUOTIENT IS	
2630								:	NOW ON TOP OF THE STACK	
2631	010200	012637	001520		MOV	(SP)+,DRMAP		:	GET MAPPING FACTOR FOR DRIVES	
2632	010204	012737	000504	001522	MOV	#504,CYLMAP		:	GET MAPPING FACTOR FOR CYLINDERS	
2633	010212	012737	012527	001524	MOV	#5463.,SECMAP		:	GET MAPPING FACTOR FOR SECTORS	
2634	010220	012737	031463	001526	MOV	#13107.,FNMAP		:	GET MAPPING FACTOR FOR FUNCTION	
2635										
2636	010226	013700	002054		MOV	MAXBA,RO		:	COMPUTE THE MAXIMUM ALLOWABLE	
2637	010232	163700	002052		SUB	BASEBA,RO		:	WORD COUNT FOR DATA TRANSFERS	
2638	010236	000241			CLC					
2639	010240	006000			ROR	RO		:	CONVERT TO WORDS	
2640										
2641	010242	012746	177777		MOV	#177777,-(SP)		:	PUT LOW DIVIDEND ON STACK	
2642	010246	005046			CLR	-(SP)		:	CLEAR HIGH DIVIDEND AND PUSH	
2643								:	IT ON STACK	
2644	010250	010046			MOV	RO,-(SP)		:	PUSH DIVISOR ON STACK	
2645	010252	004737	025212		JSR	PC,@#SDIV		:	GO TO THE 'DIVIDE' SUBROUTINE	
2646	010256	005726			TST	(SP)+		:	DISCARD THE REMAINDER. QUOTIENT IS	
2647								:	NOW ON TOP OF THE STACK	
2648	010260	005216			INC	(SP)				
2649	010262	012637	001530		MOV	(SP)+,BAMAP		:	SAVE THE MAPPING FACTOR FOR BUS ADDRESS	

MUS

```

2650 ;THE ENTIRE DISK (ALL DRIVES) IS WRITTEN WITH RANDOM PATTERNS. THE FIRST
2651 ;WORD OF EACH SECTOR GIVES THE NUMBER OF WORDS (2'S COMPLEMENT) WRITTEN IN
2652 ;THAT SECTORS. REST OF THE DATA WORDS FOR THE SECTOR ARE GENERATED USING
2653 ;THE ABSOLUTE DISK ADDRESS AS THE RANDOM SEED.
2654 ;IF THE PROGRAM WAS RESTARTED AT 210 THEN CHECK IF SW 4 IS SET. IF IT IS
2655 ;THEN DO NOT REWRITE THE DISK WITH RANDOM PATTERNS. IF SW 4 IS NOT SET THEN
2656 ;ALL THE DISKS (PRESENT) ARE WRITTEN WITH RANDOM PATTERNS.
2657
2658 010266 105737 001253 WRDSK: TSTB FRSTRT ;RESTARTED AT 210?
2659 010272 001407 BEQ 1$ ;NO
2660 010274 105037 001253 CLRB FRSTRT ;YES, CLEAR THE RESTART FLAG
2661 010300 032777 000020 170632 BIT #SW4,DSWR ;SW 4 SET?
2662 010306 001401 BEQ 1$ ;NO
2663 010310 000540 BR BEGEX1 ;YES, DONT REWRITE ALL DISKS WITH
2664 ;RANDOM PATTERNS
2665 010312 012737 010330 001110 1$: MOV #2$,$LPERR ;LUP TO '2$' ON EROR, SW 9 SET!
2666 010320 012702 001254 MOV #PDR,R2 ;POINTER TO DRIVE #'S TABLE
2667 010324 013703 001264 MOV DRVPRS,R3 ;# OF DRIVES PRESENT
2668 010330 012700 011410 2$: MOV #4872,R0 ;NUMBER OF SECTORS PER DISK
2669 010334 112201 MOVB (R2)+,R1 ;GET THE FIRST AVAILABLE DRIVE
2670 010336 042701 177770 BIC #177770,R1 ;POSITION THE BITS (15,14,13)
2671 010342 010137 001502 MOV R1,QDRV
2672 010346 000241 CLC
2673 010350 006001 ROR R1
2674 010352 006001 ROR R1
2675 010354 006001 ROR R1
2676 010356 006001 ROR R1
2677 010360 010177 170644 MOV R1,ARKDA ;BASE DISK ADDRESS
2678
2679 010364 013704 002054 MOV MAXBA,R4 ;CALCULATE MAXIMUM BUFFER
2680 010370 163704 002052 SUB BASEBA,R4
2681 010374 006204 ASR R4 ;CONVERT TO WORDS
2682 010376 042704 000377 BIC #377,R4 ;KEEP ONLY WHOLE BLOCKS
2683 010402 000304 SWAB R4
2684 010404 020427 000014 CMP R4,#12. ;MAX OF 12 SECTORS
2685 010410 003402 BLE 3$
2686 010412 012704 000014 MOV #12.,R4
2687
2688 010416 010401 3$: MOV R4,R1
2689 010420 020400 CMP R4,R0
2690 010422 003401 BLE 4$
2691 010424 010001 MOV R0,R1
2692 010426 000301 4$: SWAB R1
2693 010430 005401 NEG R1
2694 010432 010137 010446 MOV R1,5$
2695 010436 010177 170562 MOV R1,ARKWC
2696 010442 004537 016704 JSR R5,GENBUF ;GENERATE RANDOM DATA BUFER
2697 010446 000000 5$: .WORD 0 ;STARTING ADDRESS OF DATA BUFER
2698 010450 032514 .WORD DBUF
2699
2700 010452 012777 032514 170546 MOV #DBUF,ARKBA ;FROM THIS BUS ADDRESS
2701 010460 012777 000003 170534 MOV #3,ARKCS ;WRITE, GO
2702
2703 010466 104417 CON.RDY ;WAIT FOR CONTROL RDY
2704 010470 004737 020112 JSR PC,CHKCS ;ANY ERROR?
2705 010474 104001 ERROR 1 ;AN ERROR OCCURED WHILE DOING WRITE
    
```



```

2738 ;THE PROGRAM IS GOING TO LOOP BACK TO THIS POINT AT THE END OF A PASS.
2739
2740 010630 104416 BEGNEX: CON.RESET ;CLEAR EVERYTHING IN DRIVES
2741 010632 012706 001100 MOV #STACK,SP ;RESET STACK
2742 010636 005737 001264 TST DRVPRS ;ANY DRIVES LEFT IN SYSTEM?
2743 010642 001402 BEQ QMNGER
2744 010644 004737 014500 JSR PC,GENBRQ ;GO GENERATE 8 COMMANDS FOR THE Q
2745
2746
2747
2748 ;CHECK IF THERE IS ANY UNFINISHED COMMAND IN THE Q, WAITING TO BE EXECUTED.
2749 ;IF THERE IS NONE, GO TO THE 'GENBRQ' AND GENERATE 8 REQUESTS (COMMANDS),
2750 ;AND PUT THEM IN THE Q. IF THERE IS AN UNFINISHED COMMAND, FIND OUT IF
2751 ;THERE IS A PRIORITY COMMAND TO BE EXECUTED IMMEDIATELY.
2752
2753 010650 013746 001244 QMNGER: MOV PPRVLV -(SP) ;NEW PSW RAISE PRIORITY
2754 010654 012746 010662 MOV #1$,-(SP) ;RETURN PC *****
2755 010660 000002 RTI
2756
2757 010662 005737 001264 1$: TST DRVPRS ;ANY DRIVES IN SYSTEM?
2758 010666 001040 BNE 2$
2759 010670 104401 010676 TYPE ,65$ ;;TYPE ASCIZ STRING
2760 010674 000432 BR ,64$ ;;GET OVER THE ASCIZ
2761 ;;65$: .ASCIZ <15><12>/HALTING - PRESS CONTINUE TO RESTART AT '200'/'15'/'12'/'12'
2762 64$:
2763 010762 000000 HALT
2764 010764 000137 003376 JMP START
2765
2766 010770 012700 001306 2$: MOV #KEY,RO
2767 010774 005720 3$: TST (RO)+ ;ANY UNFINISHED COMMAND
2768 010776 100006 BPL 4$ ;IN Q
2769 011000 020027 001326 CMP RO,#KEY+20
2770 011004 001373 BNE 3$
2771 011006 004737 014500 JSR PC,GENBRQ ;IF NOT, GO GENERATE 8 MORE COMMANDS
2772 011012 000460 BR CHFAPN
2773
2774
2775 011014 012700 001306 4$: MOV #KEY,RO
2776 011020 032710 010000 5$: BIT #BIT12,(RO) ;ANY HIGH PRIORITY COMMAND IN Q
2777 011024 001404 BEQ 6$
2778 011026 005710 TST (RO) ;FINISHED?
2779 011030 100402 BMI 6$ ;YES
2780 011032 105710 TSTB (RO) ;IN EXECUTION?
2781 011034 100165 BPL PRICMND ;NO, GO PROCESS HIGH PRIORITY
2782
2783 011036 005720 6$: TST (RO)+ ;CHECK THE ENTIRE Q
2784 011040 020027 001326 CMP RO,#KEY+20
2785 011044 001365 BNE 5$
2786
2787
2788 ;FIND OUT IF THERE IS A WRITE CHECK FUNCTION TO BE DONE IMMEDIATELY AFTER
2789 ;A WRITE, THAT WAS DONE PREVIOUSLY.
2790
2791 011046 005737 001456 TST WCFG ;IS WRITE CHECK TO FOLLOW
2792 011052 100040 BPL CHFAPN ;WRITE? IF NOT, BRANCH
2793 ;YES
    
```



```

2826 ;NO HIGH PRIORITY COMMAND WAS FOUND IN THE Q. FIND OUT THE FIRST AVAILABLE
2827 ;COMMAND IN THE Q FOR EXECUTION.
2828 CHFAFN: CLR R0 ;WAIT FOR ANY IMMEDIATE INTERRUPT
2829 CLR -(SP) ;NEW PSW
2830 MOV #1$,-(SP) ;RETURN FOR RTI
2831 RTI
2832
2833
2834 1$: INCB R0
2835 BNE .-2
2836 MOV PPRLVL, -(SP) ;NEW PSW, LOCK OUT CPU
2837 MOV #2$,-(SP) ;RETURN FOR RTI *****
2838 RTI
2839
2840
2841 2$: MOV #KEY, R0
2842 CH1: TST (R0) ;UNFINISHED COMMAND?
2843 BMI CH2
2844 TSTB (R0) ;IN PROGRESS?
2845 BMI CH2
2846 MOV (R0), R1
2847 BIC #177770, R1
2848 TSTB BUSY(R1) ;IS THIS DRIVE BUSY?
2849 BMI CH2
2850 TSTB POS(R1) ;HAS THIS DRIVE BEEN POSITIONED
;FOR ANY OTHER COMMAND. IF YES,
;SKIP. IF NOT, PROCEED
2851
2852
2853 011240 001023 BNE CH2
2854
2855 ;CHECK IF THERE IS ANY OTHER COMMAND
2856 011242 010002 MOV R0, R2 ;ON A DRIVE THAT IS NOT THE SAME
2857 ;AS THE PREVIOUS DRIVE. THIS COMMAND
2858 011244 000414 BR 2$ ;SHOULD NOT BE IN PROGRESS
2859 ;AND SHOULD NOT HAVE BEEN COMPLETED
2860 011246 005712 1$: TST (R2) ;UNFINISHED COMMAND?
2861 011250 100412 BMI 2$
2862 011252 105712 TSTB (R2) ;IN PROGRESS?
2863 011254 100410 BMI 2$
2864 011256 011203 MOV (R2), R3
2865 011260 042703 177770 BIC #177770, R3
2866 011264 105763 001426 TSTB BUSY(R3) ;IS THE DRIVE BUSY?
2867 011270 100402 BMI 2$
2868 011272 020301 CMP R3, R1 ;IS THIS COMMAND ON THE SAME DRIVE?
2869 011274 001447 BEQ POSITION ;IF YES, GO AND POSITION THE COMMAND
;POINTED TO BY R0, (BECAUSE THERE IS
;ONE MORE COMMAND THAT CAN BE PERFORMED
;ON THE SAME DRIVE)
2870 ;CHECK REST OF THE Q
2871
2872
2873 011276 005722 2$: TST (R2)+
2874 011300 020227 001326 CMP R2, #KEY+20
2875 011304 001360 BNE 1$
2876 ;IF THERE WAS NO EXECUTABLE COMMAND
2877 011306 000470 BR EXNSK ;ON ANY OTHER DRIVE, THEN EXECUTE
2878 ;COMMAND POINTED TO BY R0
2879
2880
2881 CH2: TST (R0)+ ;CHECK ENTIRE Q
CMP R0, #KEY+20

```













3088	012164	011001				POSK:	MOV	(R0),R1	:INITIAL CHECKING PRIOR TO DOING
3089	012166	042701	177770				BIC	#177770,R1	:POSITIONING COMMAND POINTED TO BY R0
3090	012172	105761	001426				TSTB	BUSY(R1)	:DRIVE BUSY?
3091	012176	100004					BPL	1\$	
3092	012200	010137	001162				MOV	R1,\$REGO	:GET DRIVE # FOR WHICH 'BUSY' IS SET
3093	012204	104002					ERROR	2	: 'BUSY' FLAG FOR THE DRIVE (CONTAINED
3094	012206	000470					BR	7\$	: IN R1) IS SET, INDICATING THAT THE DRIVE
3095									: IS 'BUSY' AND ENGAGED IN AN ACTIVITY,
3096									: STILL AN ATTEMPT WAS MADE TO INITIATE
3097									: POSITIONING ON THIS DRIVE. THIS IS AN
3098									: UNEXPECTED ERROR CONDITION.
3099	012210	105777	167006			1\$:	TSTB	ARKCS	: CONTROL READY SET?
3100	012214	100404					BMI	2\$	: YES
3101	012216	004737	022032				JSR	PC,GT4RG	: GET RKCS, ER, DS, DA
3102	012222	104003					ERROR	3	: CONTROL READY IS NOT SET. THIS IS AN
3103	012224	000454					BR	6\$	: UNEXPECTED ERROR CONDITION AT THIS
3104									: POINT OF EXECUTION, CONTROL SHOULD
3105									: BE NORMALLY READY.
3106	012226	011002				2\$:	MOV	(R0),R2	
3107	012230	000302					SWAB	R2	
3108	012232	042702	177770				BIC	#177770,R2	
3109	012236	006302					ASL	R2	
3110	012240	016203	002032				MOV	PCMD(R2),R3	: STARTING WORD OF COMMAND TABLE
3111	012244	011377	166760				MOV	(R3),ARKDA	
3112	012250	032777	000100	166740			BIT	#RWS,ARKDS	: R/W/S RDY SET?
3113	012256	001006					BNE	3\$	: YES
3114	012260	010137	001250				MOV	R1,SRDRV	: GET DRIVE # FOR TYPING SERIAL #
3115	012264	004737	022032				JSR	PC,GT4RG	: GET RKCS, ER, DS, DA
3116	012270	104004					ERROR	4	: R/W/S RDY IS NOT SET FOR THE DRIVE. A
3117	012272	000431					BR	6\$	: (POSITIONING) SEEK WAS SUPPOSED TO BE
3118									: BE EXECUTED ON THIS DRIVE. THIS IS
3119									: AN UNEXPECTED ERROR CONDITION. AT THIS
3120									: POINT OF EXECUTION R/W/S RDY SHOULD
3121									: BE NORMALLY SET.
3122	012274	110261	001426			3\$:	MOV	R2,BUSY(R1)	: SET FLAG INDICATING DRIVE BUSY
3123	012300	152761	000200	001426			BIS	#BIT7,BUSY(R1)	
3124	012306	032710	000010				BIT	#BIT3,(R0)	: IS THIS A SEEK COMMAND
3125	012312	001006					BNE	4\$	
3126	012314	112761	000377	001436			MOV	#377,POS(R1)	: SET FLAG INDICATING, THIS DRIVE POSITIONS
3127	012322	052710	000020				BIS	#BIT4,(R0)	: SET FLAG INDICATING THIS COMMAND
3128	012326	000402					BR	5\$	
3129	012330	052710	000200			4\$:	BIS	#BIT7,(R0)	: POSITIONED. SET FLAG INDICATING
3130									: FUNCTION IN PROGRESS
3131	012334	012777	012376	166676		5\$:	MOV	#INTISK,ARKVEC	: SET UP RK11 VECTOR
3132	012342	013777	001244	166672			MOV	PPRLVL,ARKSTAT	: PSW ON INTERRUPT
3133	012350	105037	001535				CLAB	INTIFL	: CLEAR FLAG INDICATING - INTERRUPT
3134	012354	000207					RTS	PC	: OCCURRED
3135									
3136	012356	004737	014426			6\$:	JSR	PC,DRVABT	: ABORT THE FUNCTION
3137	012362	004737	016540				JSR	PC,CHKDRV	: GO CHECK, DRIVE ERRORS
3138	012366	000240					NOP		
3139	012370	005726				7\$:	TST	(SP)+	: POP THE RETURN ADDRESS
3140	012372	000137	010650				JMP	QMNGER	

```

3141                                     ;AFTER ISSUING A SEEK FOR POSITIONING, AN INTERRUPT IS ALLOWED TO TAKE
3142                                     ;*PLACE. THIS HANDLER SERVICES THE FIRST INTERRUPT, WHICH OCCURS AS A RESULT
3143                                     ;OF THE SEEK BEING ACCEPTED.
3144
3145 012376 105237 001535 INTISK: INCB INTIFL ;INDICATE THAT INTERRUPT TOOK PLACE
3146 012402 053766 001244 000002 BIS PPRLVL,2(SP) ;CPU PRIORITY TO 5 ON RETURN FROM
3147                                     ;INTERRUPT
3148
3149 012410 004737 020272 JSR PC,CHKCRDY ;CONTROL READY SET?
3150 012414 104005 ERROR 5 ;CONTROL READY NOT SET AFTER THE FIRST
3151                                     ;INTERRUPT ON INITIATING SEEK
3152
3153 012416 032777 020000 166576 BIT #SCP,ARKCS ;SCP BIT SET?
3154 012424 001403 BEQ 1$
3155 012426 004737 022222 JSR PC,RG4SDR ;GET RKCS, ER, DS, DA AND DRIVE # FOR
3156                                     ;TYPING SERIAL DRIVE #
3157 012432 104012 ERROR 12 ;SCP SET AFTER FIRST INTERRUPT ON
3158                                     ;ACCEPTING SEEK. A SEEK WAS INITIATED,
3159                                     ;AS A RESULT OF WHICH (THIS) FIRST
3160                                     ;INTERRUPT OCCURRED, BUT SCP SHOULD
3161                                     ;NOT BE SET AFTER THE FIRST INTERRUPT.
3162                                     ;(IT GETS SET ONLY AFTER THE SEEK
3163                                     ;IS DONE).
3164                                     ;THIS IS THE FIRST INTERRUPT
3165                                     ;AFTER ISSUING SEEK
3166 012434 017700 166562 1$: MOV ARKCS,RO
3167 012440 042700 177760 BIC #177760,RO ;CHECK THERE IS A SEEK FUNCTION
3168 012444 022700 000010 CMP #10,RO ;IN RKCS
3169 012450 001403 BEQ 2$
3170 012452 004737 022032 JSR PC,GT4RG ;GET RKCS, ER, DS, DA
3171 012456 104006 ERROR 6 ;RKCS DOES NOT CONTAIN 'SEEK' FUNCTION.
3172                                     ;A SEEK WAS INITIATED AS A RESULT OF WHICH
3173                                     ;(THIS) FIRST INTERRUPT OCCURRED. RKCS
3174                                     ;SHOULD CONTAIN THE SEEK FUNCTION BITS.
3175                                     ;BUT IT DID NOT
3176
3177 012460 017700 166544 2$: MOV ARKDA,RO ;GET THE DRIVE # FROM RKDA
3178 012464 042700 017777 BIC #17777,RO
3179 012470 000241 CLC
3180 012472 006100 ROL RO
3181 012474 006100 ROL RO
3182 012476 006100 ROL RO
3183 012500 006100 ROL RO
3184 012502 010001 MOV RO,R1
3185 012504 105761 001426 TSTB BUSY(R1) ;CHECK THIS DRIVE NO. WAS BUSY
3186 012510 100403 BMI 3$ ;OK IF IT WAS
3187 012512 010137 001162 MOV R1,$REGD ;GET DRIVE # (FROM RKDA)
3188 012516 104007 ERROR 7 ;'BUSY' FLAG FOR THE DRIVE WAS NOT SET.
3189                                     ;THE DRIVE # (IN RKDA) WHICH CAUSED (?)
3190                                     ;THIS (FIRST) INTERRUPT SHOULD HAVE BEEN
3191                                     ;'BUSY' (SOFTWARE FLAG). NOTE WHENEVER
3192                                     ;ANY OPERATION IS INITIATED ON ANY DRIVE
3193                                     ;'BUSY' FLAG FOR THAT DRIVE IS SET.
3194 012520 116100 001426 3$: MOVB BUSY(R1),RO
3195 012524 042700 177600 BIC #177600,RO ;FORM THE ADDRESS OF
3196 012530 062700 001306 ADD #KEY,RO ;THIS KEY
    
```

3197										
3198	012534	032710	000020		BIT	#BIT4,(R0)				; IS THE KEY ACTIVE FOR 'POSITIONING'?
3199	012540	001003			BNE	4\$				
3200	012542	010137	001162		MOV	R1,\$REGO				; GET DRIVE NUMBER
3201	012546	104010			ERROR	10				; POSITIONING FLAG (IN THE KEY) IS NOT
3202										; SET FOR THE INTERRUPTING DRIVE (GIVEN
3203										; IN EROR MESSAGE)
3204										
3205	012550	105761	001436	4\$:	TSTB	POS(R1)				; IS THE 'POSITIONING' FLAG SET?
3206	012554	001003			BNE	5\$				
3207	012556	010137	001162		MOV	R1,\$REGO				; GET DRIVE # FROM RKDA
3208	012562	104010			ERROR	10				; THIS INTERRUPT IS SUPPOSED TO BE AS
3209										; A RESULT OF INITIATING A (POSITIONING)
3210										; SEEK. WHENEVER A SEEK IS INITIATED
3211										; TO POSITION THE HEADS THE POSITIONING
3212										; FLAG (POS#) IS SET. OCCURANCE OF THIS
3213										; ERROR INDICATED THAT THE DRIVE #
3214										; (FROM RKDA)GIVING THIS INTERRUPT DIC
3215										; NOT HAVE ITS POSITIONING FLAG SET.
3216										
3217	012564	005777	166432	5\$:	TST	JRKCS				; ANY ERROR?
3218	012570	100044			BPL	8\$				; NO - RETURN
3219										
3220	012572	032737	000040	001474	BIT	#BITS,ERCODE				
3221	012600	001012			BNE	6\$				; SAME ERROR
3222	012602	042737	000040	001474	BIC	#BITS,ERCODE				
3223	012610	001026			BNE	7\$				
3224	012612	052737	000040	001474	BIS	#BITS,ERCODE				
3225										
3226	012620	004737	022222		JSR	PC,RG4SDR				; GET RKCS, ER, DS, DA AND DRIVE # FOR
3227										; TYPING SERIAL DRIVE #
3228	012624	104011			ERROR	11				; BIT 15, 'ERR' SET IN RKKCS AFTER FIRST
3229										; INTERRUPT - WHICH OCCURRED AFTER A
3230										; POSITIONING SEEK WAS INITIATED. RKER
3231										; WILL CONTAIN ERROR BIT/S
3232	012626	042710	000020	6\$:	BIC	#BIT4,(R0)				; CLEAR 'POSITIONING' BIT
3233	012632	105061	001426		CLRB	BUSY(R1)				; CLEAR 'BUSY' FLAG
3234	012636	105061	001436		CLRB	POS(R1)				; CLEAR 'POSITIONING' FLAG
3235										
3236	012642	004737	016006		JSR	PC,CLRERR				; CLEAR THE ERROR
3237	012646	052710	010000		BIS	#BIT12,(R0)				; INDICATE HIGH PRIORITY ON
3238	012652	105261	001446		INCB	RETRY(R1)				; INCREMENT RETRY COUNT
3239	012656	126127	001446	000003	CMPB	RETRY(R1),#3				; DONE RETRIES?
3240	012664	003406			BLE	8\$				; NO
3241										
3242	012666	004737	014426	7\$:	JSR	PC,DRVABT				; ABORT THE FUNCTION
3243	012672	005061	001446		CLR	RETRY(R1)				
3244	012676	005037	001474		CLP	ERCODE				
3245										
3246	012702	000002		8\$:	RTI					

3247  
3248  
3249  
3250  
3251  
3252  
3253  
3254  
3255  
3256  
3257  
3258  
3259  
3260  
3261  
3262  
3263  
3264  
3265  
3266  
3267  
3268  
3269  
3270  
3271  
3272  
3273  
3274  
3275  
3276  
3277  
3278  
3279  
3280  
3281  
3282  
3283  
3284  
3285  
3286  
3287  
3288  
3289  
3290  
3291  
3292  
3293  
3294  
3295  
3296  
3297  
3298  
3299  
3300  
3301  
3302

012704 004737 020272  
012710 104013  
  
012712 017746 166300  
012716 004737 022170  
  
012722 012601  
012724 105761 001426  
012730 100403  
012732 004737 022032  
012736 104014  
  
012740 116100 001426  
012744 042700 177600  
012750 062700 001306  
012754 032710 000020  
012760 001003  
012762 010137 001162  
012766 104010  
  
012770 105761 001436  
012774 001003  
012776 010137 001162  
013002 104010  
  
013004 004737 016540  
013010 000501  
  
013012 017777 166200 166210  
  
013020 032777 000100 166170  
013026 001005  
013030 004737 022032  
013034 010137 001250  
013040 104015  
  
013042 032777 001000 166146  
013050 001007

SKCMP: JSR PC,CHKCRDY  
ERROR 13  
  
1\$: MOV DRKDS, -(SP)  
JSR PC,CROTFL  
  
MOV (SP)+, R1  
TSTB BUSY(R1)  
BMI 2\$  
JSR PC,GT4RG  
ERROR 14  
  
2\$: MOVB BUSY(R1), RO  
BIC #177600, RO  
ADD #KEY, RO  
BIT #BIT4, (RO)  
BNE 3\$  
MOV R1, \$REGO  
ERROR 10  
  
3\$: TSTB POS(R1)  
BNE 4\$  
MOV R1, \$REGO  
ERROR 10  
  
4\$: JSR PC,CHKDRV  
BR PFTERR  
  
MOV DRKDS, DRKDA  
  
BIT #RWS, DRKDS  
BNE 5\$  
JSR PC,GT4RG  
MOV R1, SRDRV  
ERROR 15  
  
5\$: BIT #SIN, DRKDS  
BNE PSINER

: THIS ROUTINE IS ENTERED UPON COMPLETION  
: OF A SEEK OPERATION UNDER INTERRUPT MODE  
  
: CONTROL READY SET?  
: CONTROL RDY NOT SET, AFTER A SEEK  
: DONE INTERRUPT!  
  
: GET DRIVE # THAT INTERRUPTED  
: ROTATE BITS 15,14,13 TO POSITION  
: 0,1,2  
: GET THE ROTATED BITS (DRIVE ID BITS)  
: CHECK THAT DRIVE WAS BUSY  
  
: 'BUSY' FLAG FOR THE INTERRUPTING DRIVE  
: (RKDS) WAS NOT SET. DRIVE # (15,14,13 - RKDS  
: INTERRUPTED AFTER SEEK WAS COMPLETE, BUT  
: 'BUSY' FLAG CORRESPONDING TO THAT DRIVE  
: WAS CLEAR. IF THE CORRECT DRIVE WAS  
: INTERRUPTING 'BUSY' FLAG SHOULD HAVE  
: BEEN SET.  
: FORM THE ADDRESS OF THE  
: KEY  
  
: CHECK THAT POSITION BIT WAS SET  
  
: GET DRIVE NUMBER  
: POSITIONING FLAG (IN KEY) IS NOT SET  
: SET FOR INTERRUPTING DRIVE (GIVEN IN ERCP  
: MESSAGE)  
: CHECK POSITIONING FLAG WAS SET  
  
: 'POSITIONING' FLAG WAS CLEAR. WHEN  
: DOING A (POSITIONING) SEEK, THE FLAG  
: 'POS#' IS SET TO INDICATE THAT DRIVE  
: IS POSITIONING HEADS (SEEKING). THIS  
: ERROR INDICATES THAT FOR THE INTERRUPTING  
: DRIVE (RKDS) POSITIONING FLAG WAS  
: NOT SET.  
: GO, CHECK FOR DRV ERRORS (DPL, DRU, DRY, WPS,  
: IF THERE IS A DRV ERROR, RETURN HERE  
: THE DRIVE HAS BEEN Deselected, ALL  
: FURTHER COMMANDS ON THAT DRIVE ABORTED  
: RETURN HERE IF NO DRIVE ERRORS  
: GET THE DRIVE # FROM DRIVE ID BITS  
: IN RKDS AND ADDRESS THE DRIVE  
: R/W/S RDY SET?  
: YES  
: GET RKCS, ER, DS, DA  
: GET DRIVE # FOR TYPING SERIAL #  
: R/W/S RDY NOT SET FOR INTERRUPTING  
: DRIVE (RKDS). AFTER SEEK WAS COMPLETED.  
: 'SIN' ERROR?  
: YES, BRANCH



3303	013052	032777	040000	166142	BIT	#HE, DRKCS	; ANY HARD ERROR?
3304	013060	001060			BNE	PORKER	; YES
3305							; NO ERRORS DETECTED ON POSITIONING
3306	013062	105061	001426		CLRB	BUSY(R1)	; CLEAR BUSY FLAG FOR THIS DRIVE
3307	013066	000207			RTS	PC	

```

3308
3309
3310 013070 004737 022032 PSINER: JSR PC,GT4RG ;THERE WAS A 'SIN' ERROR ON
3311 013074 010137 001250 MOV R1,SRDRV ;DOING POSITIONING (SEEK)
3312 013100 104016 ERROR 16 ;GET DRIVE #, FOR TYPING SERIAL #
3313 013102 042710 000020 BIC #BIT4,(R0) ;'SIN' ON DOING (POSITIONING) SEEK
3314 013106 105061 001436 CLRB POS(R1) ;CLEAR 'POSITIONING' BIT
3315 013112 105061 001426 CLRB BUSY(R1) ;CLEAR POSITIONING FLAG
3316 013116 004737 016152 JSR PC,CLRSIN ;CLEAR BUSY FLAG FOR THIS DRIVE
3317 ;CLEAR 'SIN' ERROR
3318 013122 004737 016264 JSR PC,SINCNT ;KEEP COUNT OF 'SIN' ERRORS. IF MORE
3319 ;THAN 'MAX' DESELECT THE DRIVE
3320 013126 000432 BR PFTERR ;RETURN HERE IF COUNT=MAX
3321
3322 ;RETURN HERE IF COUNT LESS THAN MAX
3323 013130 105261 001446 PSRTY: INCB RETRY(R1) ;INCREMENT RETRY COUNT
3324 013134 105061 001436 CLRB POS(R1) ;CLEAR POSITION FLAG
3325 013140 105061 001426 CLRB BUSY(R1) ;CLEAR BUSY FLAG
3326 013144 122761 000003 001446 CMPB #3,RETRY(R1) ;DONE ALL RETRIES?
3327 013152 001403 BEQ 1$ ;YES
3328 013154 052710 010000 BIS #BIT12,(R0) ;SET HI PRIORITY ON RETRY
3329 013160 000207 RTS PC ;RETURN
3330
3331 013162 052710 104000 1$: BIS #104000,(R0) ;INDICATE COMMAND ABORTED
3332 013166 105061 001446 CLRB RETRY(R1) ;CLEAR RETRY COUNT FOR FUTURE USE
3333 013172 010102 MOV R1,R2
3334 013174 006302 ASL R2
3335 013176 005262 001672 INC ABORT(R2) ;INCREMENT ABORT COUNT
3336 013202 042710 010000 BIC #BIT12,(R0) ;CLR HI PRIORITY BIT
3337 013206 104421 002165 TYPMSG MSG10 ;PRINT ABORT MESSAGE
3338 013212 000207 RTS PC
3339
3340 013214 042710 010000 PFTERR: BIC #BIT12,(R0) ;CLEAR HI PRIORITY BIT IF SET
3341 013220 000207 RTS PC
3342
3343 013222 004737 022032 PORKER: JSR PC,GT4RG ;IF ANY ERROR BIT IN RKCS SET
3344 013226 010137 001250 MOV R1,SRDRV ;GET DRIVE #, FOR TYPING SERIAL #
3345 013232 104017 ERROR 17 ;ON DOING POSITIONING (SEEK)
3346 ;ENTER HERE
3347
3348 013234 004737 016006 JSR PC,CLRERR ;GO CLEAR THE HARD ERROR
3349 013240 042710 000020 BIC #BIT4,(R0) ;CLEAR POSITIONING BIT IN KEY
3350 013244 105061 001436 CLRB POS(R1) ;CLEAR POSITIONING FLAG FOR THIS DRIVE
3351 013250 105061 001426 CLRB BUSY(R1) ;CLEAR BUSY FLAG FOR THIS DRIVE
3352 013254 000725 BR PSRTY

```

3353									
3354									
3355									
3356	013256	105037	001534			INTHND: CLRB	INTFLG		
3357									
3358	013262	013766	001244	000002		MOV	PPRLVL,2(SP)		
3359	013270	004737	020272			JSR	PC,CHKCRDY		
3360	013274	104024				ERROR	24		
3361									
3362	013276	032777	020000	165716	1\$:	BIT	#SCP,DRKCS		
3363	013304	001403				BEQ	2\$		
3364	013306	004737	012704			JSR	PC,SKCMP		
3365	013312	000002				RTI			
3366	013314	017746	165710		2\$:	MOV	DRKDA,-(SP)		
3367									
3368									
3369	013320	004737	022170			JSR	PC,CROTFL		
3370									
3371	013324	012605				MOV	(SP)+,R5		
3372	013326	105765	001426			TSTB	BUSY(R5)		
3373	013332	100403				BMI	3\$		
3374	013334	010537	001162			MOV	R5,\$REGD		
3375	013340	104007				ERROR	7		
3376									
3377									
3378									
3379									
3380									
3381									
3382									
3383	013342	105065	001436		3\$:	CLRB	POS(R5)		
3384	013346	116500	001426			MOVB	BUSY(R5),R0		
3385	013352	042700	177760			BIC	#177760,R0		
3386	013356	062700	001306			ADD	#KEY,R0		
3387	013362	032710	000010			BIT	#BIT3,(R0)		
3388	013366	001401				BEQ	4\$		
3389	013370	104020				ERROR	20		
3390									
3391									
3392									
3393									
3394									
3395									
3396									
3397	013372	105710			4\$:	TSTB	(R0)		
3398	013374	100403				BMI	5\$		
3399	013376	010537	001162			MOV	R5,\$REGD		
3400	013402	104031				ERROR	31		
3401									
3402									
3403									
3404									
3405	013404	011002			5\$:	MOV	(R0),R2		
3406	013406	042702	177770			BIC	#177770,R2		
3407	013412	020502				CMP	R5,R2		
3408	013414	001405				BEQ	6\$		

```
;ENTER THIS ROUTINE WHEN AN INTEPRUPT
;OCCURS. NOTE THERE IS ONE OTHER
;INTERRUPT ROUTINE- INT15K
;CLEAR FLAG TO INDICATE THAT
;AN INTERRUPT OCCURED.
;CPU PRIORITY TO 5 ON RTI
;CONTROL RPY CLEAR AFTER INTRUPT ON COMPLETION
;OF FUNCTION, IT SHOULD BE SET
;SCP SET? SEARCH COMPLETE?

;GO TO SEEK COMPLETE ROUTINE

;GET THE DRIVE # TO WHICH
;THE COMMAND WAS ISSUED UNDER
;INTERRUPT MODE
;ROTATE BITS 15,14,13 TO POSITION
;0,1,2
;POP OFF THE DRIVE # FROM THE STACK
;CHECK THAT DRIVE WAS BUSY

;'BUSY' FLAG FOR THE INTERRUPTING DRIVE
;WAS NOT SET. WHENEVER ANY ACTIVITY
;IS INITIATED ON A DRIVE, THE (SOFTWARE)
;'BUSY' FLAG IS SET TO INDICATE THAT
;DRIVE IS BUSY DOING SOMETHING.
;OCCURANCE OF THIS ERROR INDICATES
;THAT THE 'BUSY' FLAG FOR THE INTERRUPTING
;DRIVE (RKDA) IS CLEAR!
;CLEAR THE POSITION FLAG
;GET THE ADDRESS OF THE
;COMMAND KEY

;CHECK IT'S NOT A SEEK FUNCTION

;(SOFTWARE) FLAG FOR THE INTERRUPTING
;DRIVE (RKDA) INDICATES THAT A SEEK
;FUNCTION WAS BEING EXECUTED ON THE
;DRIVE. IF THAT'S THE CASE, SCP BIT SHOULD
;HAVE SET ON SEEK DONE INTRUPT, BUT IT
;WAS NOT. NOTE, HERE THERE IS A CHANGE
;OF MULTIPLE ERRORS OR ERROR
;MISINTERPRETATION
;CHECK THAT FUNCTION IN
;PROGRESS BIT WAS SET
;GET DRIVE NUMBER
;IF NOT, ERROR
;'FUNCTION IN PROGRESS' FLAG (IN THE KEY)
;WAS NOT SET FOR THE INTERRUPTING DRIVE.
;IF THIS DRIVE WAS BUSY DOING THE
;FUNCTION, THE ABOVE FLAG SHOULD BE SET.
;CHECK THAT THE INTERRUPTING
;DRIVE (IN RKDA) IS THE SAME AS
;THE DRIVE IN THE KEY
```

3409	013416	010537	001162		MOV	R5,\$REGO	:GET EXPCD DRIVE NO.
3410	013422	010237	001164		MOV	R2,\$REGI	:GET DRIVE NO. RECVD
3411	013426	104032			ERROR	32	:UNEXPECTED DRIVE NO. INTERRUPTED
3412	013430	006302		6\$:	ASL	R2	
3413	013432	011003			MOV	(R0),R3	
3414	013434	010037	001536		MOV	R0,\$AVKEY	
3415	013440	000303			SWAB	R3	
3416	013442	042703	177770		BIC	#177770,R3	:GET POSITION OF THE PARAMETER
3417	013446	006303			ASL	R3	:LIST FROM THE KEY
3418	013450	017704	165546		MOV	\$RKCS,R4	
3419	013454	042704	177761		BIC	#177761,R4	:CLEAR ALL BUT FUNCTION CODE
3420	013460	016305	002032		MOV	PCMD(R3),R5	:CHECK THAT THE FUNCTION IN
3421							:RKCS AND THE KEY ARE SAME.
3422	013464	126504	000002		CMPB	2(R5),R4	
3423							
3424	013470	001406			BEQ	7\$	:IF NOT, ERROR
3425	013472	016537	000002	001162	MOV	2(R5),\$REGO	:GET EXPCD FUNCTION CODE IN RKCS
3426	013500	010437	001164		MOV	R4,\$REGI	:GET CODE RECVD
3427	013504	104033			ERROR	33	:FUNCTION CODE THAT IS IN RKCS AFTER THIS
3428							:INTERRUPT OCCURED IS NOT THE EXPECTED ONE
3429	013506	042710	000200		BIC	#BIT7,(R0)	:CLEAR 'FUNCTION IN PROGRESS' BIT
3430	013512	142761	000200	001426	BICB	#BIT7,BUSY(R1)	:CLEAR BUSY FLAG FOR THIS DRIVE
3431	013520	004737	016540		JSR	PC,CHKDRV	:CHECK FOR DRIVE ERRORS
3432	013524	000002			RTI		:IF THERE WAS ANY DRIVE ERROR
3433							:DESELECT THE DRIVE AND EXIT
3434							:IF NO ERROR, RETURN HERE
3435	013526	032777	001000	165462	BIT	#SIN,\$RKCS	:SIN ERROR?
3436	013534	001426			BEQ	10\$	
3437	013536	004737	022222		JSR	PC,RG4SDR	:GET RKCS, ER, DS, DA AND DRIVE # FOR
3438							:TYPING SERIAL DRIVE #
3439	013542	104016			ERROR	16	:SIN ERROR
3440	013544	004737	016152		JSR	PC,CLRSIN	
3441	013550	004737	016264		JSR	PC,SINCNT	:HELP COUNT OF SIN'S. IF MORE
3442							:THAN MAXM ALLOWABLE, DESELECT THE DRIVE
3443	013554	000002			RTI		:RETURN HERE IF COUNT=MAX
3444							:RETURN HERE IF LESS THAN MAX
3445	013556	105261	001446		INCB	RETRY(R1)	
3446	013562	122761	000003	001446	CMPB	#3,RETRY(R1)	
3447	013570	001405			BEQ	8\$	
3448	013572	052710	010000		BIS	#BIT12,(R0)	
3449	013576	042710	000020		BIC	#BIT4,(R0)	
3450	013602	000002			RTI		
3451							
3452	013604	004737	014426		JSR	PC,DRVABT	:ABORT THIS FUNCTION
3453	013610	000002			RTI		
3454							
3455	013612	032777	040000	165402	BIT	#HE,\$RKCS	:HARD ERROR?
3456	013620	001076			BNE	HRDERR	
3457	013622	032777	100000	165372	BIT	#ERR,\$RKCS	:SOFT ERROR?
3458	013630	001002			BNE	SFTERR	:YES
3459	013632	000137	014300		JMP	NOEROR	:NO

```

3460                                     ;THERE WAS A SOFT ERROR
3461
3462
3463 013636 032777 000001 165354 SFTErr: BIT    #WCE,DRKER    ;WCE?
3464 013644 001417                BEQ    1$
3465
3466 013646 032737 000001 001474    BIT    #BIT0,ERCODE ;ALREADY WORKING ON A WC ERROR?
3467 013654 001054                BNE    3$           ;YES - IGNORE THIS ONE
3468 013656 042737 000001 001474    BIC    #BIT0,ERCODE ;ANY OTHER ERROR?
3469 013664 001052                BNE    4$           ;YES - ABORT THIS FUNCTION
3470 013666 052737 000001 001474    BIS    #BIT0,ERCODE ;SET THE WC ERROR BIT
3471
3472 013674 005262 001632                INC    WCECN(R2)   ;INCREMENT WCE COUNT FOR
3473 013700 104421 002070                TYPMSG ,MSG2      ;THIS DRIVE
3474
3475 013704 032777 000002 165306 1$:  BIT    #CSE,DRKER    ;CSE?
3476 013712 001417                BEQ    2$
3477
3478 013714 032737 000002 001474    BIT    #BIT1,ERCODE ;ALREADY WORKING ON A CS ERROR?
3479 013722 001031                BNE    3$           ;YES - IGNORE THIS ONE
3480 013724 042737 000002 001474    BIC    #BIT1,ERCODE ;ANY OTHER ERROR?
3481 013732 001027                BNE    4$           ;YES - ABORT THIS FUNCTION
3482 013734 052737 000002 001474    BIS    #BIT1,ERCODE ;SET THE CS ERROR BIT
3483
3484 013742 005262 001652                INC    CSECN(R2)  ;INCREMENT CSE COUNT FOR THIS DRIVE
3485 013746 104421 002076                TYPMSG ,MSG3
3486 013752 104421 002120                TYPMSG ,MSG5
3487 013756 004737 021736                JSR    PC,TYPFN   ;GO TYPE OUT THE FUNCTION THAT GAVE ERROR
3488 013762 004737 022064                JSR    PC,GETINF
3489 013766 004737 022226                JSR    PC,GTSDRV
3490 013772 104021                ERROR 21         ;SAVE DRIVE #, FOR TYPING SERIAL #
3491                                     ;SOFT ERROR ON PERFORMING A DATA
3492                                     ;TRANSFER RKDA PRINTED OUT IN ERROR
3493                                     ;MESSAGE IS BROKEN DOWN INTO DRIVE #.
3494 013774 022704 000004                CMP    #4,R4      ;CYL # SEC # SUR #
3495 014000 001002                BNE    3$         ;WAS IT A READ FUNCTION?
3496 014002 004737 017110                JSR    PC,DATCHK ;GO CHECK THE DATA READ
3497
3498 014006 000137 014160                3$:  JMP    CMNERR
3499 014012 000137 014216                4$:  JMP    CMNABT

```

NO6

```

3500                                     ; IF THERE WAS A HARD
3501                                     ; ERROR DURING THE DATA
3502                                     ; TRANSFER ENTER HERE
3503
3504 014016 032777 010000 165174 HRDERR: BIT      #SKE,DRKER      ;SKE?
3505 014024 001421                                     BEQ      15
3506
3507 014026 032737 000004 001474 BIT      #BIT2,ERCODE    ;ALREADY WORKING ON A SK ERROR?
3508 014034 001051                                     BNE     CMNERR        ;YES - IGNORE THIS ONE
3509 014036 042737 000004 001474 BIC     #BIT2,ERCODE    ;ANY OTHER ERROR?
3510 014044 001064                                     BNE     CMNABT        ;YES - ABORT THIS FUNCTION
3511 014046 052737 000004 001474 BIS      #BIT2,ERCODE    ;SET THE SK ERROR BIT
3512
3513 014054 005262 001602                                     INC     SKECN(R2)     ;INCREMENT COUNT FOR SKE
3514 014060 005262 001562                                     INC     HECN(R2)     ;ON THIS DRIVE,
3515
3516 014064 104421 002062                                     TYPMSG  ,MSG1
3517
3518 014070 032777 167740 165122 15: BIT      #167740,DRKER    ;ANY HE BESIDES SKE?
3519 014076 001417                                     BEQ     25
3520
3521 014100 032737 000010 001474 BIT      #BIT3,ERCODE    ;ALREADY WORKING ON A HE ERROR?
3522 014106 001024                                     BNE     CMNERR        ;YES - IGNORE THIS ONE
3523 014110 042737 000010 001474 BIC     #BIT3,ERCODE    ;ANY OTHER ERROR?
3524 014116 001037                                     BNE     CMNABT        ;YES - ABORT THIS FUNCTION
3525 014120 052737 000010 001474 BIS      #BIT3,ERCODE    ;SET THE HE ERROR BIT
3526
3527 014126 005262 001562                                     INC     HECN(R2)     ;INCREMENT HE COUNT FOR
3528 014132 104421 002104                                     TYPMSG  ,MSG4       ;THIS DRIVE
3529
3530 014136 104421 002120 25: TYPMSG  ,MSG5      ;PRINT 'ON DOING'
3531 014142 004737 021736 JSR     PC,TYPFN     ;GO PRINT OUT THE FUNCTION
3532                                     ;WHICH GAVE THIS ERROR
3533
3534 014146 004737 022064 JSR     PC,GETINF
3535 014152 004737 022226 JSR     PC,GTSDRV
3536 014156 104022 ERROR   22          ;SAVE DRIVE #, FOR TYPING SERIAL #
3537                                     ;HARD ERROR ON DOING DATA XFER
3538 014160 105261 001446 CMNERR: INCB   RETRY(R1)    ;INCREMENT RETRY COUNT
3539 014164 122761 000003 001446 CMPB   #3,RETRY(R1)  ;3 TRIES IN ALL?
3540 014172 001411 BEQ     CMNABT        ;YES, ABORT THIS FUNCTION
3541
3542 014174 052777 010000 165334 BIS      #BIT12,JSVKEY  ;INDICATE HIGH PRIORITY ON RETRY
3543 014202 042777 000020 165326 BIC     #BIT4,JSVKEY  ;CLEAR 'POSITIONING HEADS'. IF SET
3544 014210 004737 016006 JSR     PC,CLRERR    ;CLEAR THIS ERROR
3545 014214 000002 RTI
3546
3547 014216 005037 001474 CMNABT: CLR     ERCODE    ;CLEAR ERROR CODE
3548 014222 104421 002532 TYPMSG  MSG27
3549 014226 004737 014426 JSR     PC,DRVABT    ;ABORT THE FUNCTION
3550 014232 032777 010000 164760 BIT      #SKE,DRKER    ;SEEK ERROR?
3551 014240 001416 BEQ     35          ;NO
3552 014242 104416 CON.RESET ;RESET THE CONTROLLER
3553 014244 010137 001502 MOV     R1,QDRV     ;SET DRIVE NUMBER
3554 014250 000241 CLC     ;CLEAR THE 'C' BIT
3555 014252 006037 001502 ROR     QDKV        ;LEFT JUSTIFY DRIVE NUMBER

```

3556	014256	006037	001502	ROR	QDRV	:LEFT JUSTIFY DRIVE NUMBER
3557	014262	006037	001502	ROR	QDRV	:LEFT JUSTIFY DRIVE NUMBER
3558	014266	006037	001502	ROR	QDRV	:LEFT JUSTIFY DRIVE NUMBER
3559	014272	104420		DRV.RESET		:RESET THE DRIVE
3560	014274	104416		CON.RESET		:RESET THE CONTROLLER
3561	014276	000002	35:	RTI		:THIS DATA TRANSFER

```

3562                                     ; IF THERE WAS NO HARD OR
3563                                     ; SOFT ERROR ON DATA TRANSFER
3564                                     ; ENTER HERE
3565 014300 105761 001446      NOEROR: TSTB  RETRY(R1)
3566 014304 001406              BEQ    1$
3567 014306 104421 002604      TYPMSG MSG28
3568 014312 116146 001446      MOVVB RETRY(R1),-(SP)
3569 014316 104403              TYPOS
3570 014320          002          .BYTE  2
3571 014321          000          .BYTE  0
3572
3573 014322 005037 001474      1$:  CLR    ERCODE
3574 014326 105061 001446      CLRB  RETRY(R1)
3575 014332 042777 010000      BIC   #BIT12,JSVKEY ; CLEAR PRIORITY BIT IF SET
3576 014340 004737 021006      JSR   PC,STATSC    ; GO, COLLECT STATISTICS ON THIS
                                     ; DATA TRANSFER
3577
3578
3579 014344 022704 000004      CMF   #4,R4        ; WAS IT A 'READ' FUNCTION?
3580 014350 001002              BNE   2$
3581 014352 004737 017110      JSR   PC,DATCHK    ; IF IT WAS 'READ', CHECK THE DATA
3582
3583
3584
3585 014356 022704 000002      2$:  CMP   #2,R4        ; WAS IT A 'WRITE' FUNCTION?
3586 014362 001011              BNE   3$
3587 014364 032777 000040      BIT   #BITS,JSVKEY ; IS 'WRT CHK' TO FOLLOW THIS
3588 014372 001412              BEQ   4$            ; 'WRT' FUNCTION?
3589 014374 052703 100000      BIS   #BIT15,R3    ; SET FLAG BIT TO INDICATE
                                     ; THAT WRITE CHECK SHOULD
                                     ; FOLLOW THIS WRITE
3590
3591
3592 014400 010337 001456      MOV   R3,WCFG      ; SET UP WC FLAG TO INDICATE
                                     ; THE ABOVE. THE LOWER BYTE
3593
3594
3595
3596
3597 014404 000407              BR    5$            ; CONTAINS THE POINTER TO THE
                                     ; COMMAND LIST, WHICH WILL
3598
3599
3600
3601 014406 022704 000006      3$:  CMP   #6,R4        ; WRT CHK FUNCTION?
3602 014412 001002              BNE   4$
3603 014414 005037 001456      CLR   WCFLG        ; CLEAR WR CHK FLAG
3604
3605
3606 014420 052710 100000      4$:  BIS   #BIT15,(R0) ; SET FLAG TO INDICATE THIS
3606 014424 000002              RTI   ; FUNCTION IS COMPLETED
    
```



```

3607                                     ;ABORT THE FUNCTION ON DRIVE POINTED TO BY R1
3608                                     ;CLEAR ASSOCIATED FLAGS
3609                                     ;DROP THE DRIVE IF MORE THAN 8. ABORTS
3610
3611 014426 010246                 DRVABT: MOV     R2, -(SP)           ;SAVE R2
3612 014430 105061 001446         CLRB    RETRY(R1)
3613 014434 052710 104000         BIS     #104000, (R0)       ;INDICATE THAT FUNCTION IS ABORTED
3614 014440 042710 010000         BIC     #BIT12, (R0)       ;CLEAR HIGH PRIORITY BIT IF SET
3615 014444 010102                 MOV     R1, R2
3616 014446 006302                 ASL     R2
3617 014450 005262 001672         INC     ABORT(R2)
3618 014454 026227 001672 0J0010  CMP     ABORT(R2), #10
3619 014462 003402                 BLE    1$
3620 014464 000137 016312         JMP     DSELECT             ;DROP THE DRIVE
3621 014470 012602                 1$:   MOV     (SP)+, R2       ;RESTORE R2
3622 014472 104401 002165         TYPE   ,MSG10              ;TYPE "ABORTED"
3623 014476 000207                 RTS     PC                  ;RETURN
  
```

3624  
 3625  
 3626  
 3627  
 3628  
 3629  
 3630  
 3631  
 3632  
 3633  
 3634 014500 104407  
 3635 014502 005337 002056  
 3636 014506 001025  
 3637 014510 013737 001236 002056  
 3638 014516 104401 014524  
 3639 014522 000407  
 3640  
 3641 014542  
 3642 014542 013746 001100  
 3643 014546 005216  
 3644 014550 104405  
 3645 014552 004737 021116  
 3646 014556 000137 022704  
 3647  
 3648 014562 032777 000400 164350 1\$:  
 3649 014570 001422  
 3650 014572 032777 000001 164340  
 3651 014600 001410  
 3652 014602 005727  
 3653 014604 000000 7\$:  
 3654 014606 001013  
 3655 014610 005237 014604  
 3656 014614 004737 026660  
 3657 014620 000406  
 3658 014622 005037 014604 8\$:  
 3659 014626 104401 001213  
 3660 014632 004737 021116  
 3661 014636 032777 000040 164274 2\$:  
 3662 014644 001401  
 3663 014646 000000  
 3664  
 3665 014650 004737 022656 3\$:  
 3666 014654 012700 001306 4\$:  
 3667 014660 010004  
 3668 014662 005001  
 3669 014664 010120 5\$:  
 3670 014666 062701 000400  
 3671 014672 022701 004000  
 3672 014676 001372  
 3673 014700 012700 001326  
 3674 014704 010005  
 3675 014706 012701 177740  
 3676 014712 005020 6\$:  
 3677 014714 005201  
 3678 014716 001375  
 3679

: THIS ROUTINE GENERATES THE 8 COMMAND REQUESTS AND PUT THEM IN THE QUEUE. THE  
 : FOLLOWING PARAMETERS ARE GENERATED RANDOMLY:  
 : 1. DRIVE NUMBER ON WHICH THE COMMAND WILL BE PERFORMED.  
 : 2. FUNCTION TO BE PERFORMED.  
 : 3. DISK ADDRESS - CYLINDER, SURFACE, SECTOR.  
 : 4. STARTING BUS ADDRESS.  
 : 5. WORD COUNT FOR DATA TRANSFER.

: THE QUEUE IS LOCATED AT 'CMND' (TO 'CMNDB').

GENBRQ: CKSWR  
 DEC REPCNT ; DECREMENT REPETITION COUNT  
 BNE 1\$ ; CONTINUE IF NOT 0  
 MOV PCNTR, REPCNT ; REESTABLISH REPETITION COUNT FOR EXERCISER  
 TYPE 5\$ ; TYPE ASCIZ STRING  
 BR 64\$ ; GET OVER THE ASCIZ  
 : 65\$: .ASCIZ <15><12>/END OF PASS/  
 64\$:  
 MOV \$PASS, -(SP)  
 INC (SP)  
 TYPDS  
 JSR PC, REPSTAT  
 JMP \$EOP  
 BIT #SWB, @SWR  
 BEQ 2\$  
 BIT #SW00, @SWR ; SWITCH 0 SET ?  
 BEQ 8\$ ; BR IF NOT  
 TST (PC)+ ; CHECK INDICATOR  
 0 ; TYPE TIME INDICATOR  
 BNE 2\$ ; BR IF TIME ALREADY TYPED  
 INC 7\$ ; INCREMENT THE INDICATOR  
 JSR PC, TIMTYP ; TYPE THE TIME (IF SWR 03 SET)  
 BR 2\$ ; CONTINUE  
 CLR 7\$ ; CLEAR THE INDICATOR  
 TYPE \$CRLF  
 JSR PC, REPSTAT  
 BIT #SW5, @SWR ; HALT?  
 BEQ 3\$ ; NO  
 HALT ; YES, PRESSING CONTINUE RESUMES PROG EXECUTION  
 JSR PC, CHDPRS ; CHECK IF ANY DRIVES PRESENT  
 MOV #KEY, R0  
 MOV R0, R4  
 CLR R1  
 MOV R1, (R0)+ ; CLEAR THE 8 COMMAND KEYS  
 ADD #400, R1 ; BITS 8, 9, 10 INDICATE THEIR  
 CMP #4000, R1 ; POSITION 0, 1, 2, 3, 4, 5, 6, 7  
 BNE 5\$  
 MOV #CMND, R0 ; CLEAR THE PARAMETER TABLE  
 MOV R0, R5  
 MOV #-40, R1 ; FOR THE 8 COMMANDS IN Q  
 CLR (R0)+ ; (BX4) WORDS IN ALL  
 INC R1  
 BNE 6\$

3680	014720	004737	015770			JSR	PC, CLRFLGS	: CLEAR THE FLAGS PERTAINING TO THE 8 : COMMANDS IN THE Q : R4 CONTAINS 'KEY' : R5 CONTAINS 'CMND' : ONLY 1 DRV PRESENT? : NO : YES
3681								
3682								
3683								
3684	014724	022737	000001	001264	GEN1:	CMP	#1 DRVPRS	
3685	014732	001002				BNE	22\$	
3686	014734	005000				CLR	RO	
3687	014736	000420				BR	3\$	
3688	014740	004737	025576		22\$:	JSR	PC, \$RAND	: GENERATE A RANDOM NUMBER : GET A RANDOM DRIVE NUMBER : FROM THE AVAILABLE DRIVES
3689								
3690								
3691	014744	025730					RSDRVL	
3692								
3693	014746	013746	025732			MOV	RSDRVH, -(SP)	: PUT LOW DIVIDEND ON STACK
3694	014752	005046				CLR	-(SP)	: CLEAR HIGH DIVIDEND AND PUSH : IT ON STACK
3695								
3696	014754	013746	001520			MOV	DRMAP, -(SP)	: PUSH DIVISOR ON STACK
3697	014760	004737	025212			JSR	PC, Q#\$DIV	: GO TO THE 'DIVIDE' SUBROUTINE
3698	014764	005726				TST	(SP)+	: DISCARD THE REMAINDER. QUOTIENT IS : NOW ON TOP OF THE STACK
3699								
3700	014766	012600				MOV	(SP)+, RO	
3701	014770	020037	001264			CMP	RO, DRVPRS	: MAKE SURE CORRECT MAPPING IS DONE
3702	014774	001001				BNE	3\$	
3703	014776	005300				RO		: 'QDRVE' CONTAINS RANDOM DRIVE NO.
3704	015000	005037	001502		3\$:	CLR	QDRV	
3705	015004	116037	001254	001502		MOVB	PDR(RO), QDRV	
3706	015012	105737	001502			TSTB	QDRV	: TEST IF TYPE F DRIVE
3707	015016	100016				BPL	32\$	: NOT IF POSITIVE
3708								
3709	015020	142737	000200	001502		BICB	#200, QDRV	: CLEAR THE FLAG BIT
3710	015026	132737	000001	001502		BITB	#1 QDRV	: ODD OR EVEN DRIVE ADDRESS
3711	015034	001404				BEQ	31\$	
3712								
3713	015036	005737	015724			TST	ODDEVN	: MUST HAVE BEEN AN ODD ONE
3714	015042	001730				BEQ	GEN1	: INSURE THAT ODDS WHAT WE WANT
3715	015044	000403				BR	32\$	
3716								
3717	015046	005737	015724		31\$:	TST	ODDEVN	: MUST HAVE BEEN AN EVEN ONE
3718	015052	001332				BNE	22\$	: NO GOOD - TRY AGAIN
3719								
3720	015054	004737	025576		32\$:	JSR	PC, \$RAND	: GENERATE A RANDOM NUMBER
3721	015060	025734				RSFUNL		
3722								
3723	015062	013746	025736			MOV	RSFUNH, -(SP)	: PUT LOW DIVIDEND ON STACK
3724	015066	005046				CLR	-(SP)	: CLEAR HIGH DIVIDEND AND PUSH : IT ON STACK
3725								
3726	015070	013746	001526			MOV	FNMAP, -(SP)	: PUSH DIVISOR ON STACK
3727	015074	004737	025212			JSR	PC, Q#\$DIV	: GO TO THE 'DIVIDE' SUBROUTINE
3728	015100	005726				TST	(SP)+	: DISCARD THE REMAINDER. QUOTIENT IS : NOW ON TOP OF THE STACK
3729								
3730	015102	021627	000003			CMP	(SP), #3	: MAKE SURE CORRECT FUNCTION IS SELCTD
3731	015106	001001				BNE	20\$	
3732	015110	005316				DEC	(SP)	
3733	015112	012637	001512		20\$:	MOV	(SP)+, QFNC	: THE FUNCTION THAT CAN BE PERFORMED
3734								
3735	015116	004737	025576			JSR	PC, \$RAND	: GENERATE A RANDOM NUMBER

```

3736 015122 025740 RSCYLL
3737
3738 015124 013746 025742 MOV RSCYLH,-(SP) ;PUT LOW DIVIDEND ON STACK
3739 015130 005046 CLR -(SP) ;CLEAR HIGH DIVIDEND AND PUSH
3740 ;IT ON STACK
3741 015132 013746 001522 MOV CYLMAP,-(SP) ;PUSH DIVISOR ON STACK
3742 015136 004737 025212 JSR PC,@#SDIV ;GO TO THE 'DIVIDE' SUBROUTINE
3743 015142 005726 TST (SP)+ ;DISCARD THE REMAINDER. QUOTIENT IS
3744 ;NOW ON TOP OF THE STACK
3745 015144 012637 001504 MOV (SP)+,QCYL ;(FROM 0-312)
3746 015150 022737 000313 001504 CMP #313,QCYL
3747 015156 001002 BNE 4$
3748 015160 005337 001504 DEC QCYL ;'QCYL' CONTAINS RANDOM CYLINDER NO.
3749
3750 015164 4$:
3751
3752 015164 013746 025742 MOV RSCYLH,-(SP) ;PUT LOW DIVIDEND ON STACK
3753 015170 005046 CLR -(SP) ;CLEAR HIGH DIVIDEND AND PUSH
3754 ;IT ON STACK
3755 015172 013746 001524 MOV SECMAP,-(SP) ;PUSH DIVISOR ON STACK
3756 015176 004737 025212 JSR PC,@#SDIV ;GO TO THE 'DIVIDE' SUBROUTINE
3757 015202 005726 TST (SP)+ ;DISCARD THE REMAINDER. QUOTIENT IS
3758 ;NOW ON TOP OF THE STACK
3759 015204 012637 001510 MOV (SP)+,QSEC ;(BETWEEN 0-13/8)
3760 015210 022737 000014 001510 CMP #14,QSEC
3761 015216 001002 BNE 5$
3762 015220 005337 001510 DEC QSEC ;'QSEC' CONTAINS RANDOM SEC #
3763
3764 015224 022737 077777 025742 5$: CMP #77777,RSCYLH ;SELECT SURFACE # RANDOMLY
3765 015232 101005 BHI 6$
3766 015234 012737 000020 001506 MOV #BIT4,QSUR ;SURFACE 1
3767 ;R1 CONTAINS THE MAXM WORD COUNT BASED ON
3768 ;AVAILABLE DISK SPACE.
3769 ;R2 CONTAINS THE MAXM WORD COUNT BASED ON
3770 ;AVAILABLE MEMORY SPACE.
3771 015242 005001 CLR R1
3772 015244 000404 BR 7$
3773
3774 015246 005037 001506 6$: CLR QSUR ;SURFACE 0
3775 015252 012701 000014 MOV #14,R1 ;14 SECTORS ON SUR 0
3776
3777 ;ASSUMING THAT ENOUGH MEMORY IS AVAILABLE THE MAXIMUM TRANSFER THAT CAN
3778 ;TAKE PLACE IS 20K WORDS. IF THE RANDOM CYLINDER # > OR = TO 307 AND THE
3779 ;SURFACE IS 1, CHANCES ARE THAT THE NUMBER OF WORDS TO BE TRANSFERRED MAY
3780 ;BE GREATER THAN THE SPACE AVAILABLE ON THE DISK. IN THAT CASE, THE WORDS
3781 ;COUNT IS TO BE SELECTED IN SUCH A WAY THAT THIS OVERFLOW DOES NOT OCCUR.
3782
3783 015256 023727 001504 000306 7$: CMP QCYL,#306 ;IS THE RANDOM CYL # GREATER THAN 306?
3784 015264 002003 BGE 9$
3785 015266 012701 177777 8$: MOV #177777,R1 ;IF YES, MAKE SURE THAT THE
3786 015272 000431 BR 21$ ;WORD COUNT IS SELECTED PROPERLY
3787 ;COMPUTE MAXM WORD COUNT BASED ON
3788 ;AVAILABLE DISK SPACE
3789 015274 012700 000014 9$: MOV #14,R0 ;COMPUTE # OF SECTORS AVAILABLE ON
3790 015300 163700 001510 SUB QSEC,R0 ;CYLINDERS SELECTED
3791 015304 060001 ADD R0,R1
    
```

3792	015306	012703	000312		MOV	#312,R3	: COMPUTE # OF SECTORS AVAILABLE
3793	015312	163703	001504		SUB	QCYL,R3	: BEYOND THE CYLINDER SELECTED
3794	015316	012746	000030		MOV	#30,-(SP)	:: PUT THE MULTIPLIER ON THE STACK
3795	015322	010346			MOV	R3,-(SP)	:: PUT THE MULTIPLICAND ON THE STACK
3796	015324	004737	025100		JSR	PC,@#SMULT	:: CALL THE MULTIPLY ROUTINE
3797	015330	012616			MOV	(SP)+,(SP)	:: DISREGARD THE MSB'S
3798	015332	012603			MOV	(SP)+,R3	:: GET THE LSB'S OF THE PRODUCT
3799	015334	060103			ADD	R1,R3	: COMPUTE TOTAL # OF SECTORS AVAILABLE
3800	015336	012746	000400		MOV	#400,-(SP)	:: PUT THE MULTIPLIER ON THE STACK
3801	015342	010346			MOV	R3,-(SP)	:: PUT THE MULTIPLICAND ON THE STACK
3802	015344	004737	025100		JSR	PC,@#SMULT	:: CALL THE MULTIPLY ROUTINE
3803	015350	012616			MOV	(SP)+,(SP)	:: DISREGARD THE MSB'S
3804	015352	012603			MOV	(SP)+,R3	:: GET THE LSB'S OF THE PRODUCT
3805	015354	010301			MOV	R3,R1	: COMPUTE TOTAL # OF WORDS-SPACE
3806							: AVAILABLE ON DISK FROM THE SELECTED
3807							: CYL #
3808							: COMPUTE MAXM WORD COUNT BASED ON
3809							: AVAILABLE MEMORY SPACE.
3810	015356	004737	025576	215:	JSR	PC,\$RAND	: GENERATE RANDOM NUMBER
3811	015362	025744			RSBAL		
3812							
3813	015364	013746	025746		MOV	RSBAH,-(SP)	: PUT LOW DIVIDEND ON STACK
3814	015370	005046			CLR	-(SP)	: CLEAR HIGH DIVIDEND AND PUSH
3815							: IT ON STACK
3816	015372	013746	001530		MOV	BAMAP,-(SP)	: PUSH DIVISOR ON STACK
3817	015376	004737	025212		JSR	PC,@#SDIV	: GO TO THE 'DIVIDE' SUBROUTINE
3818	015402	005726			TST	(SP)+	: DISCARD THE REMAINDER. QUOTIENT IS
3819							: NOW ON TOP OF THE STACK
3820	015404	012637	001514		MOV	(SP)+,QBUSAD	: BE USED
3821	015410	006337	001514		ASL	QBUSAD	
3822	015414	063737	002052	001514	ADD	BASEBA, QBUSAD	: FORM THE RANDOM BUS-ADDRESS
3823							: BY ADDING RANDOM OFFSET TO
3824							: THE BASE BUS-ADDRESS
3825	015422	013703	002054		MOV	MAXBA,R3	: COMPUTE # OF WORDS THAT
3826	015426	163703	001514		SUB	QBUSAD,R3	: CAN BE TRANSFERRED, USING THE
3827	015432	000241			CLC		
3828	015434	006003			ROR	R3	: ABOVE GENERATED BUS-ADDRESS WITHOUT
3829	015436	010302			MOV	R3,R2	: CAUSING A NXM
3830							
3831	015440	020201		105:	CMP	R2,R1	: SELECT SMALLER OF THE TWO
3832	015442	103401			BLO	115	: WORD-COUNTS THAT WILL BE
3833							: USED FOR GENERATING A RANDOM
3834	015444	010103			MOV	R1,R3	: WORD COUNT)
3835							
3836							: R3 CONTAINS THE MAXM WORD COUNT
3837							: POSSIBLE. COMPUTE THE WORD COUNT
3838							: MAPPING FACTOR FROM THIS.
3839	015446			115:			
3840							
3841	015446	012746	177777		MOV	#177777,-(SP)	: PUT LOW DIVIDEND ON STACK
3842	015452	005046			CLR	-(SP)	: CLEAR HIGH DIVIDEND AND PUSH
3843							: IT ON STACK
3844	015454	010346			MOV	R3,-(SP)	: PUSH DIVISOR ON STACK
3845	015456	004737	025212		JSR	PC,@#SDIV	: GO TO THE 'DIVIDE' SUBROUTINE
3846	015462	005726			TST	(SP)+	: DISCARD THE REMAINDER. QUOTIENT IS
3847							: NOW ON TOP OF THE STACK

```

3848 015464 005216 INC (SP)
3849 015466 012637 001532 MOV (SP)+,WCMAP ;WORD COUNT MAPPING FACTOR
3850
3851 015472 004737 025576 JSR PC,$RAND ;GENERATE A RANDOM NUMBER
3852 015476 025750 RSWCL
3853
3854 015500 013746 025752 MOV RSWCH,-(SP) ;PUT LOW DIVIDEND ON STACK
3855 015504 005046 CLR -(SP) ;CLEAR HIGH DIVIDEND AND PUSH
3856 ;IT ON STACK
3857 015506 013746 001532 MOV WCMAP,-(SP) ;PUSH DIVISOR ON STACK
3858 015512 004737 025212 JSR PC,$DIV ;GO TO THE 'DIVIDE' SUBROUTINE
3859 015516 005726 TST (SP)+ ;DISCARD THE REMAINDER. QUOTIENT IS
3860 ;NOW ON TOP OF THE STACK
3861 015520 012637 001516 MOV (SP)+,QWRCNT ;'QWRCNT' CONTAINS THE RANDOM
3862 ;WORD COUNT THAT WILL BE USED
3863 015524 005737 001516 TST QWRCNT ;MAKE SURE THE WORD COUNT IS
3864 015530 003004 BGT 12$ ;NOT 0
3865 015532 005437 001516 NEG QWRCNT ;TAKE CARE OF ZERO AND NEG NUMBRS
3866 015536 005237 001516 INC QWRCNT
3867 015542 113700 001502 12$: MOV# QDRV,RO
3868 015546 000241 CLC
3869 015550 006000 ROR RO
3870 015552 006000 ROR RO ;POSITION THE DRIVE NUMBER IN
3871 015554 006000 ROR RO ;BITS 15,14,13
3872 015556 006000 ROR RO
3873 015560 013701 001504 MOV QCYL,R1
3874 015564 000301 SWAB R1
3875 015566 000241 CLC
3876 015570 006001 ROR R1 ;POSITION THE CYLINDER NUMBER
3877 015572 006001 ROR R1 ;IN BITS 12-5
3878 015574 006001 ROR R1
3879 015576 050100 BIS R1,RO
3880 015600 053700 001510 BIS QSEC,RO ;RO CONTAINS THE COMPLETE DISK
3881 015604 053700 001506 BIS QSUR,RO ;ADDRESS
3882 015610 010025 MOV RO,(R5)+ ;INSERT RKDA IN THE PARAMETER TABLE
3883 ; (FOR THE 8 COMMANDS)
3884 ; WHICH FUNCTION?
3885 ; 0-READ CHECK, 1-READ, 2-WRITE
3886 015612 022737 000001 001512 CMP #1,QFNC
3887 015620 001412 BEQ 2$ ;READ
3888 015622 003014 BGT 14$ ;READ CHECK
3889 015624 012725 000002 15$: MOV #2,(R5)+ ;WRITE FUNCTION CODE
3890 015630 023727 025752 077777 CMP RSWCH,#77777 ;SHOULD WRITE CHECK BE DONE
3891 015636 101010 BHI 15$ ;AFTER THE WRITE?
3892 015640 052714 000040 BIS #BITS,(R4) ;SET FLAG IN KEY TO INDICATE
3893 ; THAT WRITE CHECK SHOULD BE
3894 ; DONE FOLLOWING THE WRITE
3895 015644 000405 BR 15$
3896
3897 015646 012725 000004 2$: MOV #4,(R5)+ ;READ FUNCTION CODE
3898 015652 000402 BR 15$
3899
3900 015654 012725 000012 14$: MOV #12,(R5)+ ;READ CHECK FUNCTION CODE
3901
3902 015660 013715 001516 15$: MOV QWRCNT,(R5) ;INSERT THE WORD COUNT (RWKC)
3903 015664 005425 NEG (R5)+ ;(2'S COMPLEMENT)

```

3904	015666	013725	001514			MOV	QBUSAD,(R5)+	: INSERT THE BUS ADDRESS (RKBA) FOR : THIS COMMAND IN THE PARAMETER : TABLE
3905								
3906	015672	053724	001502	16\$:		BIS	QDRV,(R4)+	: SET THE DRIVE NUMBER INSIDE : THE KEY FOR THIS COMMAND
3907								: GENERATED 8 COMMANDS IN : THE QUEUE?
3908	015676	020427	001326			CMP	R4,#KEY+20	
3909								
3910	015702	001402				BEQ	17\$	
3911	015704	000137	014724			JMP	GEN1	: IF NOT, GO BACK AND GENERATE : THE NEXT COMMAND AND THE : PARAMETERS (RKWC, BA, DA)
3912								: CHANGE FROM ODD/EVEN TO EVEN/ODD
3913								: KEEP ONLY ONE BIT
3914	015710	005237	015724	17\$:		INC	ODDEVN	: ALL 8 COMMANDS HAVE BEEN : GENERATED IN THE TASK-QUEUE
3915	015714	042737	177776	015724		BIC	#177776,ODDEVN	
3916	015722	000207				RTS	PC	
3917								
3918								
3919	015724	000000					ODDEVN: 0	

K07

```

3920
3921
3922
3923
3924 015726 004737 016006      EXCR LUP: JSR      PC, CLRERR      ;WAIT FOR OTHER DRIVES TO GET DONE
3925                                     ;THEN ISSUE A CONTROL RESET
3926 015732 010146      MOV      R1, -(SP)
3927 015734 012701 001306      MOV      #KEY, R1      ;CLEAR OUT THE COMMAND-KEYS
3928 015740 042721 114220      1$: BIC      #114220, (R1)+
3929 015744 020127 001326      CMP      R1, #KEY+20
3930 015750 001373      BNE      1$
3931 015752 012601      MOV      (SP)+, R1
3932 015754 004737 015770      JSR      PC, CLRFLGS      ;CLEAR OUT THE VARIOUS FLAGS PERTAINING
3933                                     ;TO THE 8 COMMANDS IN THE Q
3934 015760 012706 001100      MOV      #STACK, SP      ;REESTABLISH STACK POINTER
3935 015764 000137 010650      JMP      QMNGER          ;START OVER AGAIN, PROCESS THE COMMANDS
3936                                     ;IN THE Q AGAIN. NOTE THAT ON LOOPING
3937                                     ;(ON ERROR, WITH SW 9) AN ATTEMPT IS MADE
3938                                     ;TO RECREATE THE SET OF EVENTS WHICH LED TO
3939                                     ;ERROR.
3940
3941
3942
3943
3944
3945      ;CLRFLGS
3946      ;THIS ROUTINE CLEARS OUT THE VARIOUS FLAGS USED FOR THE 8 COMMANDS IN THE QUEUE.
3947 015770 012700 001426      CLRFLGS: MOV      #BUSY, R0      ;CLEAR THE 8 BUSY FLAGS
3948 015774 005020      1$: CLR      (R0)+
3949 015776 020027 001462      CMP      R0, #QSCNT+2      ;ALL DONE?
3950 016002 001374      BNE      1$              ;NO
3951 016004 000207      RTS      PC
  
```



```

3952 ;CLRERR
3953 ;THIS ROUTINE IS ENTERED WHEN A HARD ERROR HAS OCCURRED AND IT HAS TO BE
3954 ;CLEARED. THE DRIVES THAT HAVE BEEN BUSY ARE CHECKED TO SEE IF THEIR R/W/S
3955 ;RDY BIT HAS SET. WHEN R/W/S IS SET, CHECKING IS DONE FOR ANY ERROR. IF A
3956 ;ERROR OCCURED IT IS REPORTED, IF NOT, APPROPRIATE FLAGS ARE SET AND
3957 ;CLEARED FOR THAT DRIVE. AFTER ABOVE IS DONE FOR ALL DRIVES THAT HAVE BEEN
3958 ;SEEKING, A CONTROL RESET IS ISSUED TO CLEAR THE HARD ERROR.
3959
3960 016006 010446 CLRERR: MOV R4,-(SP) ;SAVE R4, R4 ON THE STACK
3961 016010 010546 MOV R5,-(SP)
3962 016012 005005 CLR R5
3963 016014 005077 163210 CLR @RKDA
3964
3965 016020 105765 001426 1$: TSTB BUSY(R5) ;WAS THIS DRIVE BUSY SEEKING?
3966 016024 100035 BPL 4$ ;NO
3967 016026 005037 001472 CLR TIMER
3968 016032 032777 000100 163156 2$: BIT #RWS,@RKDS ;R/W/S SET?
3969 016040 001015 BNE 3$ ;YES
3970 016042 005237 001472 INC TIMER ;KEEP TIME
3971 016046 001371 BNE 2$ ;WAIT FOR R/W/S RDY
3972 016050 004737 022222 JSR PC,RG4SDR ;GET RKCS, ER, DS, DA AND DRIVE # FOR
3973 ;TYPING SERIAL DRIVE #
3974 016054 104004 ERROR 4 ;R/W/S READY DID NOT SET
3975 ;FOR THIS DRIVE, WAITED LONG ENOUGH.
3976 016056 032777 001000 163132 BIT #SIN,@RKDS ;SIN ERROR ON THIS DRIVE?
3977 016064 001403 BEQ 3$
3978 016066 004737 022222 JSR PC,RG4SDR ;GET RKCS, ER, DS, DA AND DRIVE # FOR
3979 ;TYPING SERIAL DRIVE #
3980 016072 104016 ERROR 16 ;SIN OCCURED ON THIS DRIVE
3981
3982 016074 116504 001426 3$: MOVB BUSY(R5),R4 ;FORM THE ADDRESS OF THE
3983 016100 042704 177760 BIC #177760,R4 ;KEY WHICH MADE THIS DRIVE
3984 016104 062704 001306 ADD #KEY,R4 ;BUSY
3985
3986 016110 042714 010000 BIC #10000,(R4) ;CLEAR HIGH PRIORITY BIT, IF SET
3987 016114 105065 001426 CLRB BUSY(R5) ;MAKE THIS DRIVE FREE, AVAILABLE
3988
3989
3990 016120 062777 020000 163102 4$: ADD #20000,@RKDA ;ADDRESS THE NEXT POSSIBLE DRIVE
3991 016126 005205 INC R5 ;INCREMENT COUNT
3992 016130 022705 000010 CMP #10,R5 ;ALL DONE
3993 016134 001331 BNE 1$ ;NO
3994
3995 016136 004737 020340 JSR PC,CRCMND ;SAVE INFO ABOUT THE PAST & PRESENT COMAND
3996 016142 104416 CON.RESET
3997 016144 012605 MOV (SP)+,R5 ;RESTORE R4,R5
3998 016146 012604 MOV (SP)+,R4
3999 016150 000207 RTS PC ;RETURN

```

```

4000 ;CLRSIN
4001 ;THIS ROUTINE IS ENTERED WHEN THERE IS A 'SIN' ERROR. AT TIME OF ENTRY
4002 ;RKDA CONTAINS THE DRIVE # THAT GAVE 'SIN'. A DRIVE RESET IS DONE ON THAT
4003 ;DRIVE. AFTER IT IS DONE, ROUTINE 'CLRHE' IS ENTERED, TO WAIT FOR THE
4004 ;OTHER DRIVES THAT HAVE BEEN DOING SEEKS. WHEN ALL THE DRIVES GIVE
4005 ;'R/W/S RDY' A CONTROL RESET IS DONE, RETURN IS MADE BACK TO THIS
4006 ;ROUTINE-'CLRSIN'- AND FINALLY CONTROL IS TRANSFERRED BACK TO THE MAIN
4007 ;PROGRAM.
4008
4009 016152 017737 163052 001516 CLRSIN: MOV QWKDA,QWRCNT ;SAVE DISK ADDRESS
4010 016160 004737 016006 JSR PC,CLARR ;GO, WAIT FOR OTHER DRIVES TO COMPLETE
4011 ;THEIR SEEKS(IF THEY ARE DOING ANY)
4012 ;THEN DO CON.RESET TO CLR THE EROR.
4013 016164 013777 001516 163036 MOV QWRCNT,QWKDA ;ADDRESS THE DRIVE AGAIN
4014 016172 004737 020314 JSR PC,DRCMND ;SAVE INFO ABOUT THE PAST & PRESENT COMAND
4015 016176 012777 000015 163016 MOV #15,QWKCS ;DO DRIVE RESET ON THE
4016 ;DRIVE
4017 ;INDICATED IN RKDA
4018 016204 104417 CON.RDY
4019 016206 005037 001472 CLR TIMER
4020 016212 032777 000100 162776 1$: BIT #RWS,QWKDS ;WAIT FOR R/W/S RDY TO SET
4021 016220 001015 BNE 2$
4022 016222 005237 001472 INC TIMER
4023 016226 001371 BNE 1$
4024 016230 004737 022222 JSR PC,RG4SDR ;GET RKCS, ER, DS, DA AND DRIVE # FOR
4025 ;TYPING SERIAL DRIVE #
4026 016234 104004 ERROR 4 ;R/W/S RDY DID NOT SET AFTER
4027 ;DOING DRIVE RESET, TIMED OUT
4028 016236 032777 001000 162752 BIT #SIN,QWKDS
4029 016244 001403 BEQ 2$
4030 016246 004737 022222 JSR PC,RG4SDR ;GET RKCS, ER, DS, DA AND DRIVE # FOR
4031 ;TYPING SERIAL DRIVE #
4032 016252 104016 ERROR 16 ;A DRIVE RESET WAS DONE ON THIS DRIVE
4033 ;TO CLEAR 'SIN', BUT 'SIN' DID NOT GET
4034 ;CLEARED
4035
4036 016254 004737 020340 2$: JSR PC,CRCMND ;SAVE INFO ABOUT THIS COMMAND
4037 016260 104416 CON.RESET ;DO IT TO CLEAR OUT MASK F/FS
4038 016262 000207 RTS PC ;EXIT FROM THIS ROUTINE
    
```

```

4039 ;SINCNT
4040 ;THIS ROUTINE IS ENTERED WHEN A 'SIN' ERROR OCCURS. THE 'SIN' COUNT FOR
4041 ;THE DRIVE GIVING 'SIN' IS INCREMENTED. IF MORE THAN 5 'SIN' ERRORS
4042 ;OCCURRED THE DRIVE IS DESELECTED. AT THE TIME OF ENTRY R1 CONTAINS THE
4043 ;DRIVE NUMBER THAT GAVE 'SIN' ERROR.
4044 ;CALL: JSR PC,SINCNT
4045 ;----- RETURN HERE IF SIN COUNT (MAXIMUM ALLOWABLE)
4046 ; WAS EXCEEDED.
4047 ;----- RETURN HERE IF TOTAL SIN COUNT LESS THAN MAXIMUM
4048 ; ALLOWABLE.
4049
4050
4051 016264 105261 001622 SINCNT: INCB SINCN(R1) ;INCREMENT 'SIN' COUNT FOR THIS DRIVE
4052 016270 122761 000005 001622 CMPB #5,SINCN(R1) ;5 ERRORS OCCURRED?
4053 016276 101403 BLOS 1$ ;YES
4054 016300 062716 000002 ADD #2,(SP) ;ADJUST PC FOR RETURN TO THE RIGHT POINT
4055 016304 000207 RTS PC ;RETURN
4056
4057 016306 000137 016312 1$: JMP DSELECT ;5 ERRORS OCCURRED, GO DESELECT

```

608

4058  
 4059  
 4060  
 4061  
 4062  
 4063  
 4064  
 4065  
 4066  
 4067  
 4068  
 4069 016312 012705 001253  
 4070 016316 013702 001264  
 4071 016322 062702 001253  
 4072 016326 005205  
 4073 016330 111503  
 4074 016332 042703 177600  
 4075 016336 020301  
 4076 016340 001403  
 4077 016342 020502  
 4078 016344 103770  
 4079 016346 000472  
 4080 016350 111502  
 4081 016352 020527 001264  
 4082 016356 001406  
 4083 016360 010504  
 4084 016362 005205  
 4085 016364 112524  
 4086 016366 022704 001263  
 4087 016372 001374  
 4088 016374 105065 177777  
 4089  
 4090  
 4091  
 4092  
 4093 016400 104401 002336  
 4094 016404 010146  
 4095 016406 104403  
 4096 016410 001  
 4097 016411 000  
 4098 016412 004737 026766  
 4099  
 4100 016416 005337 001264  
 4101  
 4102 016422 004737 022656  
 4103  
 4104  
 4105 016426 012746 177777  
 4106 016432 005046  
 4107  
 4108 016434 013746 001264  
 4109 016440 004737 025212  
 4110 016444 005726  
 4111  
 4112 016446 012637 001520  
 4113

```

:DSELECT
:THIS ROUTINE IS ENTERED WHEN A DRIVE IS TO BE DESELECTED (TAKEN
:OUT OF SELECTION LIST), BECAUSE THE FATAL ERRORS ON THAT DRIVE
:HAS REACHED A MAXIMUM COUNT. R1 CONTAINS THE DRIVE NUMBER THAT
:THAT IS TO BE DESELECTED. THE DRIVE IS DESELECTED IF 1. TOTAL SIN COUNT
:FOR THAT DRIVE REACHES THE MAXIMUM ALLOWABLE 2. IF A FATAL ERROR
:LIK F DRIVE UNSAFE, DRIVE POWER LOW OCCURS. 3. IF WPS GETS SET, OR DRY
:IS CLEAR.

DSELECT: MOV      #PDR-1,R5
          MOV      DRVPRS,R2
          ADD      #PDR-1,R2
1$:      INC      R5
          MOV      (R5),R3
          BIC      #177600,R3
          CMP      R3,R1
          BEQ      R5,R2
          CMP      R5,R2
          BLO      1$
          BR      11$
2$:      MOV      (R5),R2
5$:      CMP      R5,#PDR+10
          BEQ      4$
          MOV      R5,R4
          INC      R5
3$:      MOV      (R5)+(R4)+
          CMP      #PDR+7,R4
          BNE      3$
4$:      CLRB     -1(R5)

          ;NUMBER OF DRIVES BEING TESTED
          ;FOR END ADDRESS OF TABLE
          ;LOCATE THE DRIVE (TO BE
          ;DESELECTED) IN THE TABLE
          ;DROP THE F FLAG
          ;IS THIS THE ONE
          ;CONTAINING AVAILABLE DRIVES
          ;FINISHED ?
          ;BR IF NOT
          ;DRIVE WAS NOT FOUND IN TABLE, EXIT
          ;GET THE DRIVE NUMBER
          ;IS THIS DRIVE # THE LAST ENTRY IN TABLE?
          ;YES

          ;IF NOT, TAKE OUT THIS DRIVE # FROM
          ;THE MIDDLE AND PUSH UP THE
          ;REST OF THE ENTRIES

          ;CLEAR LAST ENTRY IN TABLE

          ;THE DRIVE # TYPED OUT WAS DESELECTED
          ;BECAUSE ERROR COUNT EXCEEDED THE
          ;MAXIMUM ALLOWABLE
          ;TYPE "DRIVE DROPPED"
          ;PUSH DRIVE NUMBER ON STACK
          ;TYPE IT ON THE TERMINAL

          ;GO TYPE OUT SERIAL NO OF THE DRIVE.
          ;IF SW 1 IS SET.
          ;DECREMENT THE TOTAL NUMBER OF
          ;DRIVES PRESENT
          ;CHECK IF ANY DRIVES PRESENT
          ;IF NONE GOT TO END OF PASS. $EOP

          ;PUT LOW DIVIDEND ON STACK
          ;CLEAR HIGH DIVIDEND AND PUSH
          ;IT ON STACK
          ;PUSH DIVISOR ON STACK
          ;GO TO THE 'DIVIDE' SUBROUTINE
          ;DISCARD THE REMAINDER. QUOTIENT IS
          ;NOW ON TOP OF THE STACK
          ;TO BE USED FOR GENERATING RANDOM
          ;DRIVE NUMBERS.
    
```

```

4114 016452 012704 001306      MOV      #KEY,R4      ;DESELECT THE COMMANDS
4115 016456 011405      6$: MOV      (R4),R5    ;IN THE 'COMMAND Q' CORRESPONDING
4116 016460 042705 177770      BIC      #177770,R5  ;TO THE DESELECTED DRIVE
4117 016464 020105      CMP      R1,R5
4118 016466 001002      BNE      7$
4119 016470 052714 104000      BIS      #104000,(R4) ;INDICATE COMMAND DESELECTED
4120 016474 005724      7$: TST      (R4)+     ;(AND COMPLETED)
4121 016476 022704 001326      CMP      #KEY+20,R4
4122 016502 001365      BNE      6$
4123
4124 016504 105702      8$: TSTB   R2        ;'F' TYPE DRIVE ?
4125 016506 100012      BPL      11$        ;NO - JUST EXIT
4126 016510 032701 000001      BIT      #1,R1     ;ODD OR EVEN DRIVE NUMBER
4127 016514 001403      BEQ      9$
4128 016516 042701 000001      BIC      #1,R1
4129 016522 000402      BR       10$
4130 016524 052701 000C      9$: BIS      #1,R1
4131 016530 000137 016312      10$: JMF     DSELCT   ;DROP CORRESPONDING DRIVE
4132
4133
4134 016534 000137 010630      11$: JMP     BEGNEY  ;GO RESTART EXERSISOR PART OF TEST
    
```

```

4135 ;CHKDRV
4136 ;THIS ROUTINE CHECKS FOR FATAL ERRORS OF THE DRIVE LIKE DPL, DRV
4137 ;WPS. IF ANY ONE OF THESE ERRORS OCCUR THE DRIVE IS DESELECTED
4138 ;AND NO MORE FUNCTIONS WILL BE PERFORMED ON THAT DRIVE. RI
4139 ;CONTAINS THE DRIVE NUMBER TO BE CHECKED.
4140
4141 ;*NOTE 1: IN THE ERROR MESSAGE WHERE RKDA IS PRINTED OUT, IT GIVES
4142 ;*ONLY THE DRIVE NUMBER (NOT CYLINDER, SURFACE AND SECTOR).
4143
4144 ;CALL: JSR PC,CHKDRV
4145 ;----- RETURN HERE IF ANY FATAL ERROR OCCURED
4146 ;----- RETURN HERE IF THERE WAS NO FATAL ERROR
4147
4148 016540 017746 162464 CHKDRV: MOV DRKDA, -(SP) ;SAVE RKDA
4149 016544 010146 MOV RI, -(SP) ;GET DRIVE #
4150 016546 000241 CLC
4151 016550 006016 ROR (SP)
4152 016552 006016 ROR (SP)
4153 016554 006016 ROR (SP)
4154 016556 006016 ROR (SP)
4155 016560 012677 162444 MOV (SP)+, DRKDA ;ADDRESS THE DRIVE TO BE CHECKED
4156 016564 032777 010000 162424 BIT #DPL, DRKDS ;DRIVE POWER LOW?
4157 016572 001403 BEQ 1$
4158 016574 004737 022222 JSR PC, RG4SDR ;GET RKCS, ER, DS, DA AND DRIVE #
4159 ;FOR TYPING SERIAL NUMBER
4160 016600 104035 ERROR 35 ;DRIVE POWER LO, *NOTE 1 ABOVE
4161 016602 032777 002000 162406 1$: BIT #DRU, DRKDS ;DRIVE
4162 016610 001403 BEQ 2$
4163 016612 004737 022222 JSR PC, RG4SDR ;GET RKCS, ER, DS, DA AND DRIVE #
4164 ;FOR TYPING SERIAL NUMBER
4165 016616 104036 ERROR 36 ;DRIVE UNSAFE BIT IS SET
4166 ;*NOTE 1 ABOVE
4167 016620 032777 000040 162370 2$: BIT #WPS, DRKDS ;WRITE PROTECT SET?
4168 016626 001403 BEQ 3$
4169 016630 004737 022222 JSR PC, RG4SDR ;GET RKCS, ER, DS, DA AND DRIVE #
4170 ;FOR TYPING SERIAL NUMBER
4171 016634 104037 ERROR 37 ;WPS SET, CHECK WRITE PROTECT SWITCH ON DRIVE
4172 ;*NOTE 1 ABOVE
4173 016636 032777 000200 162352 3$: BIT #DRY, DRKDS ;DRIVE READY CLEAR?
4174 016644 001004 BNE 4$
4175 016646 004737 022222 JSR PC, RG4SDR ;GET RKCS, ER, DS, DA AND DRIVE #
4176 ;FOR TYPING SERIAL NUMBER
4177 016652 104034 ERROR 34 ;DRIVE READY CLEAR, SHOULD BE SET
4178 ;*NOTE 1 ABOVE
4179 016654 000411 BR 5$
4180
4181 016656 032777 012040 162332 4$: BIT #12040, DRKDS ;ANY ERROR?
4182 016664 001005 BNE 5$ ;YES
4183 016666 012677 162336 MOV (SP)+, DRKDA ;RESTORE RKDA
4184 016672 062716 000002 ADD #2, (SP) ;ADJUST RETURN ADDRESS
4185 016676 000207 RTS PC
4186
4187 016700 000137 016312 5$: JMP DSELECT
    
```

```

4188 :GENBUF
4189 :THIS ROUTINE GENERATES A BUFFER FULL OF RANDOM DATA WORDS. THIS BUFFER
4190 :IS THEN USED TO WRITE DATA ON THE DISK. AT THE TIME OF ENTRY, RKDA
4191 :CONTAINS THE DISK ADDRESS WHERE WRITE WILL BE DONE. SEED WORDS USED FOR
4192 :THE RANDOM NUMBERS ARE:
4193 :   1) ABSOLUTE DISK ADDRESS (DRIVE #, CYL #, SEC #, SUR #) - $HNUM
4194 :   2) COMPLEMENT OF THE ABOVE WORD
4195 :CALL: JSR      R5,GENBUF
4196 :       X
4197 :       Y
4198 :
4199 :
4200 016704 104414      GENBUF: SAVREG      :SAVE REGISTERS
4201 016706 016504 000002  MOV      2(R5),R4    :GET STARTING ADDRESS OF BUFFER
4202 016712 011503      MOV      (R5),R3    :GET WORD COUNT (# OF WORDS TO
4203 :                               :BE GENERATED)
4204
4205 016714 017702 162310      MOV      @RKDA,R2
4206 016720 010237 025756 1$:  MOV      R2,R50TH    :GET THIS RANDOM SEED
4207 016724 010237 025754      MOV      R2,R50TL
4208 016730 005137 025754      COM      R50TL    :GET LO RANDOM SEED
4209
4210 016734 022703 177400      CMP      #-400,R3  :IF THE BUFFER IS MORE THAN
4211 016740 003003          BGT      2$        :ONE SECTOR (400 WORDS) LONG.
4212 016742 010305          MOV      R3,R5    :GENERATE THE BUFFER IN SUCH
4213 016744 005003          CLR      R3        :A WAY THAT EACH SECTOR
4214 016746 000404          BR       3$        :BEGINS WITH RANDOM DATA
4215
4216 016750 062703 000400 2$:  ADD      #400,R3    :WORDS GENERATED USING THAT
4217 016754 012705 177400      MOV      #-400,R5  :SECTOR ADDRESS AS THE RANDOM
4218 :                               :SEED
4219 016760 010524          3$:  MOV      R5,(R4)+  :FIRST WORD OF EVERY SECTOR IS
4220 :                               :A WORD COUNT (2'S COMP) INDICATING #
4221 :                               :OF WORDS ACTUALLY WRITTEN IN THAT SECTOR
4222 :                               :ALL DONE?
4222 016762 005205          INC      R5
4223 016764 001427          BEQ      8$
4224
4225 016766 004737 025576          4$:  JSR      PC,$RAND    :GENERATE DATA WORDS
4226 016772 025754          RSDTL
4227 016774 012737 000002 017104  MOV      #2,RCNT
4228 017002 013737 025755 017106  MOV      R50TH,RNUM
4229 017010 000406          BR       6$
4230 017012 005337 017104          5$:  DEC      RCNT
4231 017016 001763          BEQ      4$
4232 017020 013737 025754 017106  MOV      R50TL,RNUM
4233 017026 005737 017106          6$:  TST      RNUM
4234 017032 001767          BEQ      5$
4235 017034 013724 017106          MOV      RNUM,(R4)+
4236 017040 005205          INC      R5
4237 017042 001351          BNE      4$
4238 :
4239 017044 005703          8$:  TST      R3
4240 017046 001412          BEQ      10$
4241 :
4242 017050 010246          MOV      R2,-(SP)
4243 017052 042716 177760          BIC      #177760,(SP)  :(ABSOLUTE DISK ADDRESS & ITS

```

4244	017056	022726	000013		CMP	#13,(SP)+	:COMPLEMENT) TO USE FOR
4245	017062	001002			BNE	9\$	:GENERATING DATA WORDS
4246	017064	062702	000004		ADD	#4,R2	:OF THE NEXT BLOCK
4247							
4248	017070	005202		9\$:	INC	R2	;HI RANDOM SEED
4249	017072	000712			BR	1\$	
4250							
4251	017074	104415		10\$:	RESREG		:RESTORE REGISTERS
4252	017076	062705	000004		ADD	#4,R5	:ADJUST RETURN ADDRESS
4253	017102	000205			RTS	R5	:RETURN
4254							
4255	017104	000000		RCNT:	0		
4256	017106	000000		RNUM:	0		



603

```

4257 :DATCHK
4258 :THIS ROUTINE IS ENTERED WHEN THE DATA THAT WAS READ FROM THE DISK IS TO
4259 :BE CHECKED. AT THE TIME OF ENTRY R3 CONTAINS THE OFFSET OF POINTER TO
4260 :THE ADDRESS OF THE PARAMETER LIST (RKCS,DA,WC,BA). DATA IS CHECKED IN
4261 :BLOCKS OF 1 SECTOR (400 WORDS). EACH BLOCK IS GENERATED USING THE SECTOR
4262 :ADDRESS (AND ITS COMPLEMENT) AS RANDOM SEEDS. WHEN A DATA MISCOMPARISON
4263 :OCCURS THE BUS ADDRESS, EXPECTED AND RECEIVED DATA ARE REPORTED.
4264
4265 017110 104414          DATCHK: SAVREG          ;SAVE R0-R5
4266 017112 016303 002032  MOV      PCMD(R3),R3    ;GET ADDRESS OF THE PARAMETER
4267                                ;TABLE
4268 017116 016304 000004  MOV      4(R3),R4      ;GET WORD COUNT (2'S COMP)
4269 017122 016305 000006  MOV      6(R3),R5      ;GET BUS ADDRESS
4270 017126 011301          MOV      (R3),R1       ;GET DISK ADDRESS
4271
4272 017130 010146          MOV      R1,-(SP)
4273 017132 004737 022170  JSR      PC,CROTFL     ;ROTATE BITS 15,14,13 TO
4274                                ;0,1,2
4275 017136 012602          MOV      (SP)+,R2      ;POP OFF DRIVE # FROM THE STACK
4276 017140 006302          ASL      R2
4277 017142 062702 001712  ADD      #DATER,R2    ;FORM THE ADDRESS OF DATA ERROR COUNT
4278                                ;FOR THIS DRIVE
4279 017146 012737 177764 001540  MOV      #-14,ECOUNT  ;CREATE RANDOM SEEDS TO
4280 017154 011337 025756          MOV      (R3),RSDTH   ;BE USED FOR CHECKING DATA
4281 017160 013737 025756 025754  MOV      RSDTH,RSDTL
4282 017166 005137 025754          COM      RSDTL
4283
4284 017172 022704 177400  1$:      CMP      #-400,R4  ;DATA IS CHECKED IN 1 SECTOR
4285 017176 003003          BGT      2$          ;BLOCKS. EACH SECTOR IS GENERATED
4286 017200 010403          MOV      R4,R3      ;USING THAT SECTOR ADDRESS
4287 017202 005004          CLR      R4          ;AS THE RANDOM SEED
4288 017204 000404          BR       3$
4289
4290 017206 062704 000400  2$:      ADD      #400,R4
4291 017212 012703 177400          MOV      #-400,R3
4292 017216 012500          3$:      MOV      (R5)+,R0
4293                                ;SAVE THE FIRST WORD OF THE SECTOR.
4294                                ;FIRST WORD OF EVERY SECTOR IS A COUNT
4295                                ;(2'S COMP) INDICATING # OF WORDS ACTLALY
4296                                ;WRITTEN IN THAT SECTOR
4297                                ;INCREMENT COUNT OF # OF WORDS (WRITEN)
4298                                ;IN THE SECTOR
4299                                ;INCREMENT COUNT OF DATA WORDS TO BE CHECKED
4300                                ;BRANCH, IF DONE
4301 017226 004737 025576  4$:      JSR      PC,$RAND    ;GENERATE RANDOM DATA WORD
4302 017232 025754          RSDTL
4303 017234 012737 000002 017104  MOV      #2,RCNT
4304 017242 013737 025756 017106  MOV      RSDTH,RNUM
4305 017250 000406          BR       10$
4306 017252 005337 017104  9$:      DEC      RCNT
4307 017256 001763          BEQ      4$
4308 017260 013737 025754 017106  MOV      RSDTL,RNUM
4309 017266 005737 017106  10$:     TST      RNUM
4310 017272 001767          BEQ      9$
4311
4312 017274 005700          TST      R0
    
```

4313	017276	001401				BEQ	5\$		
4314	017300	005200				INC	RO		
4315									
4316	017302	023715	017106		5\$:	CMP	RNUM,(R5)	:EXPCTD WORD = RECD WORD?	
4317	017306	001431				BEQ	8\$	:YES	
4318									
4319	017310	005700				TST	RO		
4320	017312	001005				BNE	6\$		
4321	017314	005715				TST	(R5)		
4322	017316	001425				BEQ	8\$		
4323	017320	005037	001164			CLR	\$REG1		
4324	017324	000403				BR	7\$		
4325									
4326	017326	013737	017106	001164	6\$:	MOV	RNUM,\$REG1	:SAVE EXPCTD DATA WORD	
4327	017334	005212			7\$:	INC	(R2)	:INCRMT DATA EROR COUNT FOR THIS DRIVE	
4328	017336	005737	001540			TST	ECOUNT	:STORE ONLY 12 (DEC) DATA ERRORS	
4329	017342	001413				BEQ	8\$	:IF MORE EXIT	
4330	017344	010537	001162			MOV	R5,\$REG0	:SAVE ERROR BUS ADDRESS	
4331	017350	011537	001166			MOV	(R5),\$REG2	:SAVE ERROR DATA WORD	
4332	017354	010137	001170			MOV	R1,\$REG3		
4333	017360	004737	022226			JSR	PC,GTSDRV	:SAVE DRIVE #, FOR TYPING SERIAL #	
4334	017364	104023				ERROR	23	:DATA (COMPARISON) ERROR ON DOING	
4335								:READ FROM DISK NORMALLY ONLY 12 DATA	
4336								:ERRORS WILL BE REPORTED. THROUGH	
4337								:CHECKING WILL BE DONE, ERRORS	
4338								:EXCEEDING 12 WON'T BE REPORTED. IF	
4339								:YOU WANT MORE, CHANGE 'ECOUNT' TO	
4340								:WHATEVER # OF ERRORS YOU WANT REPORTED	
4341	017366	005237	001540			INC	ECOUNT		
4342									
4343	017372	005725			8\$:	TST	(R5)+		
4344	017374	005203				INC	R3	:DONE CHECKING?	
4345	017376	001313				BNE	4\$		
4346									
4347	017400	005704			14\$:	TST	R4	:ANY MORE SECTOR BLOCKS	
4348	017402	001427				BEQ	17\$	:TO CHECK? IF NOT, EXIT	
4349									
4350	017404	010146				MOV	R1,-(SP)		
4351								:GET THE NEW RANDOM SEEDS	
4352	017406	042716	177760			BIC	#177760,(SP)	:ABSOLUTE DISK ADDRESS & ITS COMPLEMENT	
4353	017412	022726	000013			CMP	#13,(SP)+	:TO USE FOR GENERATING DATA WORDS	
4354	017416	001002				BNE	15\$	:OF THE NEXT BLOCK	
4355	017420	062701	000004			ADD	#4,R1		
4356									
4357	017424	005201			15\$:	INC	R1		
4358	017426	032777	000002	161564		BIT	#CSE,ARKER	:IF THERE WAS A CSE THEN CHECK	
4359	017434	001403				BEQ	16\$	:ONLY THOSE SECTORS THAT WERE READ	
4360	017436	020177	161566			CMP	R1,ARKDA		
4361	017442	001407				BEQ	17\$		
4362	017444	010137	025756		16\$:	MOV	R1,RSDTH		
4363	017450	010137	025754			MOV	R1,RSDTL		
4364	017454	005137	025754			COM	RSDTL		
4365	017460	000644				BR	1\$		
4366	017462	104415			17\$:	RESREG		:RESTORE R0-R5	
4367	017464	000207				RTS	PC		

```

4368 .SBTTL ROUTINE TO SIZE MEMORY
4369
4370 ;*****
4371 ;CALL:
4372 ; JSR PC,$SIZE
4373 ; RETURN
4374 ;$LSTAD WILL CONTAIN:
4375 ; WITH KT11 OPTION -- LAST VIRTUAL ADDRESS OF THE LAST BANK
4376 ; WITHOUT KT11 OPTION -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
4377 ;$LST3K WILL CONTAIN THE LAST BANK AS A SAF
4378 ;$KT11 IS THE MEMORY MANAGEMENT KEY
4379 ;BIT07 = 0 DON'T USE MEMORY MANAGEMENT
4380 ; MUST BE SETUP BEFORE THE CALL
4381 ;BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
4382 ;*
4383 ;* DETERMINED BY ROUTINE
4384
4384 017466 010046 $SIZE: MOV R0,-(SP) ;;SAVE R0 ON THE STACK
4385 017470 010146 MOV R1,-(SP) ;;SAVE R1 ON THE STACK
4386 017472 010246 MOV R2,-(SP) ;;SAVE R2 ON THE STACK
4387 017474 010346 MOV R3,-(SP) ;;SAVE R3 ON THE STACK
4388 017476 013746 000004 MOV @#ERRVEC-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
4389 017502 013746 000006 MOV @#ERRVEC+2,-(SP)
4390 017506 010600 MOV SP,R0 ;;SAVE THE STACK POINTER
4391 ;;SET THE ERRVEC PS TO THE PRESENT PS
4392 017510 104400 TRAP ;;PUSH OLD PSW AND PC ON STACK
4393 017512 012637 000006 MOV (SP)+,@#ERRVEC+2 ;;SAVE THE PSW IN @#ERRVEC+2
4394 017516 012701 003776 MOV #3776,R1 ;;SETUP ADDRESS
4395 017522 105727 TSTB (PC)+ ;;USE MEMORY MANAGEMENT?
4396 017524 000200 $KT11: .WORD 200 ;;SET TO USE MEMORY MANAGEMENT
4397 017526 100062 BPL SCORE ;;BR IF NO
4398 017530 012737 017666 000004 MOV #SKTNEX,@#ERRVEC ;;SET FOR TIMEOUT
4399 017536 005737 177572 TST @#SR0 ;;KT11 ARE YOU THERE?
4400 017542 052737 100000 017524 BIS #100000,$KT11 ;;YES--SET KT11 KEY
4401 017550 005046 CLR -(SP) ;;INITIALIZE FOR "PAR" LOADING
4402 017552 012702 172340 MOV #KIPAR0,R2 ;;ADDRESS OF FIRST "PAR"
4403 017556 012703 000010 MOV #108,R3 ;;LOAD EIGHT "PAR.'S" AND EIGHT "PDR.'S"
4404 017562 012762 077406 177740 1$: MOV #77406,-40(R2) ;;PDR = 4K UP, READ/WRITE
4405 017570 011622 MOV (SP),(R2)+ ;;LOAD "PAR"
4406 017572 062716 000200 ADD #200,(SP) ;;UPDATE FOR NEXT "PAR"
4407 017576 077307 SOB R3,1$ ;;LOOP UNTIL ALL EIGHT ARE LOADED
4408 017600 012742 177600 MOV #177600,-(R2) ;;SETUP KIPAR7 FOR I/O
4409 017604 005042 CLR -(R2) ;;SETUP KIPAR6 FOR TESTING
4410 017606 012737 017624 000004 MOV #2$,@#ERRVEC ;;CATCH TIMEOUT IF NO SR3
4411 017614 012737 000020 172516 MOV #20,@#SR3 ;;ENABLE 22 BIT MODE
4412 017622 000401 BR 3$ ;;THIS POP-11 HAS A SR3 REGISTER
4413 017624 022626 2$: CMP (SP)+,(SP)+ ;;CLEAN OFF THE STACK--NO SR3
4414 017626 005237 177572 3$: INC @#SR0 ;;TURN ON MEMORY MANAGEMENT
4415 017632 012737 017656 000004 MOV #SKTOUT,@#ERRVEC ;;SET FOR TIME OUT
4416 017640 005737 143776 4$: TST @#143776 ;;TRAP ON NON-EX-MEM
4417 017644 062712 000040 ADD #40,(R2) ;;MAKE A 1K STEP
4418 017650 023712 172356 CMP @#KIPAR7 (R2) ;;LAST ONE?
4419 017654 101371 BHI 4$ ;;NO--TRY IT
4420 017656 011202 $KTOUT: MOV (R2),R2 ;;GET LAST BANK+1
4421 017660 005037 177572 CLR @#SR0 ;;TURN OFF MEMORY MANAGEMENT
4422 017664 000421 BR $SIZEX
4423 017666 042737 100000 017524 $KTNEX: BIC #100000,$KT11 ;;KT11 NON-EXISTENT

```

```

4424 017674 012737 017724 000004 SCORE: MOV    #SCROUT,2#ERRVEC ;; SET FOR TIMEOUT
4425 017702 005002          CLR    R2          ;; SET UP BANK
4426 017709 062701 004000 1$:    ADD    #4000,R1    ;; INCREMENT BY 1K
4427 017710 062702 000040          ADD    #40,R2      ;; 1K STEP
4428 017714 005711          TST    (R1)        ;; TRAP ON TIME OUT
4429 017716 022701 177776          CMP    #177776,R1  ;; LAST ONE
4430 017722 001370          BNE    1$         ;; NO--TRY AGAIN
4431 017724 162701 004000 $SCROUT: SUB   #4000,R1
4432 017730 162702 000040 $SIZEX: SUB   #40,R2      ;; DROP BACK
4433 017734 010006          MOV    R0,SP       ;; RESTORE THE STACK
4434 017736 012637 000006          MOV    (SP)+,2#ERRVEC+2 ;; RESTORE ERROR VECTOR
4435 017742 012637 000004          MOV    (SP)+,2#ERRVEC
4436 017746 010137 017770          MOV    R1,$LSTAD  ;; LAST ADDRESS
4437 017752 010237 017772          MOV    R2,$LSTBK  ;; LAST BANK
4438 017756 012603          MOV    (SP)+,R3   ;; RESTORE R3
4439 017760 012602          MOV    (SP)+,R2   ;; RESTORE R2
4440 017762 012601          MOV    (SP)+,R1   ;; RESTORE R1
4441 017764 012600          MOV    (SP)+,R0   ;; RESTORE R0
4442 017766 000207          RTS    PC
4443 017770 000000          $LSTAD: .WORD    0 ;; CONTAINS THE LAST ADDRESS
4444 017772 000000          $LSTBK: .WORD    0 ;; CONTAINS THE LAST BANK
    
```

```

4445 ;TYPDBO
4446 ;THIS ROUTINE CONVERTS A VIRTUAL ADDRESS TO PHYSICAL ADDRESS AND TYPES
4447 ;OUT THE 6 DIGIT PHYSICAL ADDRESS. R2 CONTAINS VIRTUAL ADDRESS AT THE TIME
4448 ;OF ENTRY.
4449 ;TYPE OUT IS INHIBITED IF SW 13 IS SET.
4450
4451 017774 032777 020000 161136 TYPDBO: BIT #SW13,2SWR ;INHIBIT TYPEOUT?
4452 020002 001042 BNE 2$ ;YES
4453 020004 010346 MOV R3,-(SP)
4454 020006 010446 MOV R4,-(SP)
4455 020010 010546 MOV R5,-(SP)
4456
4457 020012 010246 MOV R2,-(SP) ;PUSH VA ON STACK
4458 020014 004737 022170 JSR PC,CROTLF ;ROTATE BITS 15,14,13 INTO 2,1,0
4459 020020 012603 MOV (SP)+,R3 ;FORM OFFSET TO BE USED
4460 020022 006303 ASL R3 ;FOR KIPAR
4461
4462 020024 016304 172340 MOV KIPAR0(R3),R4 ;GET THE BASE PAGE ADDRESS FROM
4463 ;KIPAR
4464 020030 005003 CLR R3 ;ROTATE LEFT 6 TIMES (MULTIPLY
4465 020032 017705 177772 MOV #6,R5 ;BY 100 OCTAL) TO GET THE
4466 020036 006304 1$: ROL R4 ;BASE BUS ADDRESS (PHYSICAL)
4467 020040 005205 INC R5 ;R3 CONTAINS MSB-2 BITS
4468 020042 001375 BNE 1$
4469
4470
4471 020044 010246 MOV R2,-(SP) ;STRIP OFF TOP 3 BITS FROM VA &
4472
4473 020046 042716 160000 BIC #160000,(SP) ;GET THE OFFSET INSIDE THE PAGE
4474 020052 062604 ADD (SP)+,R4 ;FORM THE ENTIRE PHYSICAL
4475 020054 005503 ADC R3 ;ADDRESS. R4 CONTAINS LOWER 16 BITS
4476 ;R3 CONTAINS TOP 2 BITS
4477 020056 010437 001162 MOV R4,$REG0 ;SAVE LOWER 16 BITS OF PA
4478 020062 010337 001164 MOV R3,$REG1 ;SAVE TOP 2 BITS OF PA
4479 020066 012746 001162 MOV #5REG0,-(SP) ;PUSH POINTER TO PA ON STACK
4480 020072 004737 024464 JSR PC,2$SDB20 ;CONVERT THE 18 BIT BINARY
4481 ;ADDRESS TO OCTAL ASCII NUMBERS ON RETURN
4482 ;POINTER TO THE FIRST ASCII CHARACTERS
4483 ;IS ON STACK
4484 020076 004737 025014 JSR PC,2$SUPRS ;TYPE OUT THE OCTAL 6 DIGIT
4485 ;PHYSICAL ADDRESS.
4486
4487 020102 012605 MOV (SP)+,R5
4488 020104 012604 MOV (SP)+,R4
4489 020106 012603 MOV (SP)+,R3
4490
4491 020110 000207 2$: RTS PC

```

```

4492 :CHKCS
4493 :THIS ROUTINE CHECKS IF BIT 15 OF RKCS WAS SET. IF IT WAS RETURN IS MADE TO
4494 :THE ERROR MESSAGE FOLLOWING THE JSR CALL. IF NOT, THE ERROR MADE TO SKIP
4495 :OVER THE ERROR MESSAGE.
4496
4497 020112 005777 161104 CHKCS: TST      2RKCS      ;BIT 15 SET?
4498 020116 100073          BPL      COMRET      ;NO
4499 020120 004737 022032 JSR      PC,GT4RG    ;YES, GET RKCS. ER, DS, DA
4500 020124 000207          RTS      PC          ;RETURN TO THE ERROR MESSAGE
4501
4502 :CHKDA
4503 :THIS ROUTINE CHECKS IF RKDA INCREMENTED CORRECTLY. IF NOT, RETURN IS MADE
4504 :TO THE ERROR MESSAGE FOLLOWING THE JSR CALL. IF YES, RETURN IS MADE TO
4505 :SKIP OVER THE ERROR MESSAGE.
4506 :AT THE TIME OF ENTRY, R2 CONTAINS THE EXPECTED RKDA.
4507
4508 020126 020277 161076 CHKDA: CMP      R2,2RKDA  ;DID RKDA INCREMENT CORRECTLY?
4509 020132 001465          BEQ      COMRET      ;YES
4510 020134 010237 001162 MOV      R2,$REG0    ;GET EXPCTD RKDA
4511 020140 017737 161064 001164 MOV      2RKDA,$REG1 ;GET RKDA RECVD
4512 020146 000207          RTS      PC          ;RETURN TO THE ERROR MESSAGE
4513
4514 :CHKBA
4515 :THIS ROUTINE CHECKS IF RKBA INCREMENTED CORRECTLY. IF NOT, RETURN IS MADE
4516 :TO THE ERROR MESSAGE FOLLOWING THE JSR CALL. IF YES, RETURN IS MADE TO
4517 :SKIP OVER THE ERROR MESSAGE.
4518 :AT THE TIME OF ENTRY, R3 CONTAINS THE WORD COUNT (# OF WORDS TRANSFERRED),
4519 :R4 CONTAINS THE BUS ADDRESS WHERE THE TRANSFER STARTED.
4520
4521 020150 000241          CHKBA: CLC
4522 020152 006103          ROL      R3          ;FORM THE EXPCTD BUS ADDRESS
4523 020154 060304          ADD     R3,R4
4524 020156 000241          CLC
4525 020160 006003          ROR     R3
4526 020162 020477 161040 CMP      R4,2RKBA    ;DID RKBA INCREMENT CORRECTLY?
4527 020166 001447          BEQ     COMRET      ;YES
4528 020170 010437 001162 MOV      R4,$REG0    ;GET EXPCTD RKBA
4529 020174 000207          RTS     PC          ;RETURN TO THE EROR MESSAGE
4530
4531 020176 017737 161024 001164 MOV      2RKBA,$REG1 ;GET RKBA RECVD
4532
4533 :CHKMEX
4534 :THIS ROUTINE CHECKS THAT RKBA OVERFLOWED AND MEX BIT WAS SET IN RKCS (BIT 4).
4535 :IF RKBA OVERFLOWED CORRECTLY, THE RETURN ADDRESS IS ADJUSTED TO SKIP THE
4536 :ERROR MESSAGE ON RETURN.
4537
4537 020204 017746 161012 CHKMEX: MOV      2RKCS,-(SP) ;GET RKCS
4538 020210 042716 177717 BIC     #177717,(SP) ;GET MEX BITS 4,5
4539 020214 022726 000020 CMP     #BIT4,(SP)+ ;CHECK BIT 4 SET?
4540 020220 001432          BEQ     COMRET      ;YES, OK
4541 020222 004737 022032 JSR     PC,GT4RG    ;SAVE RKCS,ER,DS,DA
4542 020226 000207          RTS     PC          ;RETURN
    
```

```

4543 ;CHKWC
4544 ;THIS ROUTINE CHECKS IF BKWC OVERFLOWED CORRECTLY AFTER A DATA TRANSFER
4545 ;IF IT DID NOT, RETURN IS MADE TO THE ERROR MESSAGE. IF IT DID, RETURN IS
4546 ;MADE TO SKIP OVER THE ERROR MESSAGE.
4547
4548 020230 005777 160770 CHKWC: TST      BKWC      ;BKWC OVERFLOWED?
4549 020234 001424          BEQ      COMRET   ;YES
4550 020236 017737 160766 001162 MOV     BKDA,$REG0
4551 020244 017737 160754 001164 MOV     BKWC,$REG1
4552 020252 000207          RTS      PC      ;RETURN TO THE EROR MESSAGE
4553
4554
4555
4556 ;CHKRWS
4557 ;THIS ROUTINE CHECKS IF R/W/S RDY BIT IN RKDS IS SET. IF IT IS NOT SET RETURN
4558 ;IS MADE TO THE ERROR MESSAGE FOLLOWING THE JSR CALL. IF IT IS, THE RETURN
4559 ;ADDRESS IS ADJUSTED TO SKIP OVER THE ERROR MESSAGE ON RETURN.
4560
4561 020254 032777 000100 160734 CHKRW: BIT     #RWS,BKDS ;RWS RDY SET?
4562 020262 001011          BNE     COMRET   ;YES
4563 020264 004737 022032 JSR     PC,GT4RG ;GET RKCS, ER, DS, DA
4564 020270 000207          RTS      PC      ;RETURN TO THE EROR MESSAGE
4565
4566
4567 ;CHKCRDY
4568 ;THIS ROUTINE CHECKS IF CONTROL READY BIT IN RKCS IS SET. IF IT IS NOT,
4569 ;RETURN IS MADE TO THE ERROR MESSAGE FOLLOWING THE JSR CALL. IF IT IS,
4570 ;RETURN ADDRESS IS ADJUSTED TO SKIP OVER THE ERROR MESSAGE.
4571
4572 020272 105777 160724 CHKCRDY: TSTB   BKCS      ;CONTROL READY SET?
4573 020276 100403          BMI     COMRET   ;YES
4574 020300 004737 022032 JSR     PC,GT4RG ;GET RKCS, ER, DS, DA
4575 020304 000207          RTS      PC      ;RETURN TO THE EROR MESSAGE
4576
4577 020306 062716 000002 COMRET: ADD     #2,(SP) ;ADJUST RETURN ADDRESS TO SKIP OVER MESSAGE
4578 020312 000207          RTS      PC

```

```

4579 ; THIS ROUTINE KEEPS A HISTORY OF THE COMMANDS THAT ARE BEING EXECUTED
4580 ; ON THE RK11. 'PRSFNC' CONTAINS INFORMATION ABOUT THE PRESENT COMMAND
4581 ; WHICH IS ABOUT TO BE INITIATED. 'PSTFNC' CONTAINS INFORMATION ABOUT THE
4582 ; COMMAND THAT WAS EXECUTED BEFORE THIS NEW ONE. THERE ARE MULTIPLE POINTS
4583 ; OF ENTRY DEPENDING ON THE TYPE OF COMMAND BEING PRESENTLY INITIATED:
4584
4585 ; DRCMND - ENTERED WHEN A DRIVE RESET IS BEING INITIATED. DRIVE # IS SAVED
4586 ; IN BITS 0-2 OF 'PRSCMND' AND BIT 8 IS SET.
4587
4588 ; CRCMND - ENTERED WHEN A CONTROL RESET IS BEING INITIATED. BIT 14 OF
4589 ; 'PRSCMND' IS SET.
4590
4591 ; POSCMND - ENTERED WHEN A POSITIONING SEEK IS BEING INITIATED. BITS 0-2
4592 ; CONTAIN THE DRIVE NUMBER ON WHICH THE POSITIONING SEEK WAS DONE. ALSO
4593 ; BIT 7 IS SET.
4594
4595 ; IN ALL ABOVE CASES BIT 15 OF 'PRSCMND' IS SET.
4596
4597 ; FNCMND - ENTERED WHEN A COMMAND OTHER THAN ANY ONE OF THE ABOVE IS BEING
4598 ; INITIATED (EX: READ, WRITE, ETC).
4599 ; THE OFFSET TO THE COMMAND KEY (BASE=KEY) IS SAVED IN BITS 0-3 OF 'PRSCMND'.
4600
4601 ; IT SHOULD BE NOTED THAT CONTENTS OF 'PRSFNC' ARE PUSHED INTO 'PSTFNC'
4602 ; AND SAVED, BEFORE PUTTING INFO ABOUT THE PRESENT COMMAND IN 'PRSFNC'.
4603
4604 ; RD CONTAINS ADDRESS OF THE COMMAND KEY, AT THE TIME OF ENTRY.
4605
4606 020314 012737 100400 001512 DRCMND: MOV #BIT15+BIT8,QFNC
4607 020322 017746 160702 MOV @RKDA,-(SP) ;SAVE DRIVE #
4608 020326 004737 022170 JSR PC,CROTFL
4609 020332 052637 001512 BIS (SP)+,QFNC
4610 020336 000424 BR P2
4611
4612 020340 012737 140000 001512 CRCMND: MOV #BIT15+BIT14,QFNC
4613 020346 000420 BR P2
4614
4615 020350 011037 001512 POSCMND: MOV (RD),QFNC
4616 020354 042737 177770 001512 BIC #177770,QFNC ;GET DRIVE NO.
4617 020362 052737 100200 001512 BIS #BIT15+BIT7,QFNC
4618 020370 000407 BR P2
4619
4620 020372 005037 001512 FNCMND: CLR QFNC
4621 020376 010046 P1: MOV RD,-(SP)
4622 020400 162716 SUB #KEY,(SP)
4623 020404 052637 001512 BIS (SP)+,QFNC
4624
4625 020410 013737 001462 001464 P2: MOV PRSFNC,PSTFNC
4626 020416 013737 001512 001462 MOV QFNC,PRSFNC
4627 020424 000207 RTS PC

```



```

4628
4629
4630
4631
4632
4633
4634
4635 020426 010046
4636 020430 010146
4637
4638 020432 012700 001462
4639 020436 104401 020746
4640 020442 104401 020772
4641 020446 104401 020762
4642 020452 005710
4643 020454 100053
4644 020456 105710
4645 020460 100427
4646 020462 032710 040000
4647 020466 001014
4648
4649 020470 104401 020476
4650 020474 000410
4651
4652 020516
4653 020516 000425
4654
4655 020520
4656 020520 104401 020526
4657 020524 000404
4658
4659 020536
4660 020536 000463
4661
4662 020540
4663 020540 104401 020546
4664 020544 000412
4665
4666 020572
4667 020572 011046
4668 020574 042716 177770
4669 020600 104402
4670 020602 000441
4671
4672 020604 011001
4673 020606 016101 002032
4674 020612 016104 000002
4675 020616 004737 021736
4676 020622 104401 020630
4677 020626 000403
4678
4679 020636
4680 020636 011146
4681 020640 104402
4682 020642 104401 020650
4683 020646 000403
    
```

```

:HISTRY
:THIS ROUTINE TYPES OUT INFORMATION ABOUT THE FUNCTION THAT WAS
:BEING PERFORMED ON THE RK AT THE TIME OF ERROR AND THE FUNCTION
:THAT WAS PERFORMED JUST BEFORE THAT FUNCTION (WHICH LED TO
:THE ERROR). THIS ROUTINE IS CALLED WHEN AN ERROR OCCURS AND SW 12
:IS SET.

HISTRY: MOV    RO,-(SP)
        MOV    R1,-(SP)

        MOV    @PRSFNC,RO
        TYPE   ,MH1
        TYPE   ,MH3
        TYPE   ,MH2
65:     TST    (RO)
        BPL    3$ ;READ, READ CHECK, WRITE, WRITE CHECK, SEEK
        TSTB   (RO)
        BMI    2$ ;POSITIONING (SEEK)
        BIT    @BIT14,(RO)
        BNE    1$ ;CONTROL RESET

        TYPE   65$ ;:TYPE ASCIZ STRING
        BR     64$ ;:GET OVER THE ASCIZ
::65$: .ASCIZ /DRESET ON DRV /
64$:   BR     7$

1$:    TYPE   67$ ;:TYPE ASCIZ STRING
        BR     66$ ;:GET OVER THE ASCIZ
::67$: .ASCIZ /CRESET/
66$:   BR     4$

2$:    TYPE   69$ ;:TYPE ASCIZ STRING
        BR     68$ ;:GET OVER THE ASCIZ
::69$: .ASCIZ /POSITIONING DRIVE /
68$:   MOV    (RO),-(SP)
7$:    BIC    @177770,(SP) ;TYPE DRIVE NO.
        TPOC
        BR     4$

3$:    MOV    (RO),R1
        MOV    PCMD(R1),R1 ;GO TYPE OUT THE FUNCTION
        MOV    2(R1),R4 ;BEING PERFORMED
        JSR    PC,TYPFN
        TYPE   71$ ;:TYPE ASCIZ STRING
        BR     70$ ;:GET OVER THE ASCIZ
::71$: .ASCIZ <15><12>/DA=/
70$:   MOV    (R1),-(SP) ;TYPE OUT DISK ADDRESS
        TPOC
        TYPE   73$ ;:TYPE ASCIZ STRING
        BR     72$ ;:GET OVER THE ASCIZ
    
```

4684					73\$:	.ASCIZ	/ BA=	
4685	020656				72\$:	MOV	6(R1),-(SP)	
4686	020656	016146	000006			TYPOC		
4687	020662	104402				TYPE	75\$	::TYPE ASCIZ STRING
4688	020664	104401	020672			BR	74\$	::GET OVER THE ASCIZ
4689	020670	000403			75\$:	.ASCIZ	/ WC=	
4690					74\$:	MOV	4(R1),-(SP)	
4691	020700					TYPOC		
4692	020700	016146	000004					
4693	020704	104402						
4694								
4695	020706	020027	001464		4\$:	CMP	RO,#PSTFNC	
4696	020712	001410				BEQ	5\$	
4697	020714	005720				TST	(RO)+	
4698	020716	104401	020746			TYPE	,MH1	
4699	020722	104401	020775			TYPE	,MH4	
4700	020726	104401	020762			TYPE	,MH2	
4701	020732	000647				BR	6\$	
4702	020734	104401	001213		5\$:	TYPE	\$CRLF	
4703	020740	012601				MOV	(SP)+,R1	
4704	020742	012600				MOV	(SP)+,RO	
4705	020744	000207				RTS	PC	
4706	020746	005015	052506	041516	MH1:	.ASCIZ	<15><12>/FUNCTION /	
4707	020754	044524	047117	000040				
4708	020762	042440	051122	051117	MH2:	.ASCIZ	/ ERROR /	
4709	020770	000040						
4710	020772	052101	000		MH3:	.ASCIZ	/AT/	
4711	020775	120	044522	051117	MH4:	.ASCIZ	/PRIOR TO/	
4712	021002	052040	000117					
4713							.EVEN	

009

```

4714      :STATSTC
4715      :AT THE TIME OF ENTRY R1 CONTAINS THE DRIVE NUMBER FOR WHICH THE STATISTIC
4716      :IS TO BE OBTAINED. R5 CONTAINS THE POINTER TO THE PARAMETER TABLE, FOR
4717      :THE COMMAND EXECUTED ON THE ABOVE DRIVE. R4 CONTAINS THE FUNCTION CODE
4718      : (WRITE, READ, ETC) FOR WHICH STATISTICS ARE TO BE TAKEN.
4719
4720 021006 010046      STATSTC:MOV  R0,-(SP)      ;PUSH R0, R2, R3 ONTO THE
4721 021010 010246      MOV  R2,-(SP)      ;STACK
4722 021012 010346      MOV  R3,-(SP)
4723
4724 021014 005002      CLR  R2
4725 021016 005701      TST  R1              ;DRIVE 0?
4726 021020 001404      BEQ  2$              ;FORM THE OFFSET FOR THE
4727 021022 062702 000004 1$:  ADD  #4,R2           ;'WORDS XFERRED COUNTS'-
4728 021026 005301      DEC  R1              ;NWRTL, NRDL
4729 021030 001374      BNE  1$
4730 021032 016506 000004 2$:  MOV  4(R5),R0        ;GET WORD COUNT (RKWC) FROM
4731 021036 005400      NEG  R0              ;THE PARAMETER TABLE
4732
4733 021040 005777 160156      TST  DRKCS           ;ANY ERROR DURING THE XFER?
4734 021044 100004      BPL  3$
4735
4736 021046 017703 160152      MOV  DRKWC,R3        ;YES,
4737 021052 005403      NEG  R3              ;GET THE # OF WORDS THAT
4738 021054 160300      SUB  R3,R0           ;WERE ACTUALLY X-FERRED
4739
4740 021056 022704 000002 3$:  CMP  #2,R4           ;WRITE FUNCTION?
4741 021062 001005      BNE  5$
4742
4743 021064 060062 001732 4$:  ADD  R0,NWRTL(R2)    ;YES, ADD THE # OF WORDS
4744 021070 005562 001734      ADC  NWRTL(R2)       ;XFERRED (WRITE)
4745 021074 000404      BR   6$              ;NOTE IT'S 2-WORD COUNT LO. HI
4746
4747 021076 060062 001772 5$:  ADD  R0,NRDL(R2)     ;ADD THE # OF WORDS READ
4748                                ;NOTE THAT WRT CHK,
4749                                ;READ CHK ARE ALSO CON-
4750                                ;SIDERED TO BE 'READ'
4751 021102 005562 001774      ADC  NRDL(R2)        ;CARRY OVER TO THE HI WORD
4752
4753 021106 012603 6$:  MOV  (SP)+,R3        ;POP R3,R2,R0 FROM THE STACK
4754 021110 012602      MOV  (SP)+,R2
4755 021112 012600      MOV  (SP)+,R0
4756 021114 000207      RTS  PC

```

E09

4757  
 4758  
 4759  
 4760 021116 104401 002377  
 4761 021122 013700 001264  
 4762 021126 012701 001254  
 4763  
 4764 021132 104401 001213  
 4765 021136 112102  
 4766 021140 042702 177770  
 4767 021144 010246  
 4768 021146 104403  
 4769 021150 003  
 4770 021151 000  
 4771 021152 104401 002662  
 4772  
 4773 021156 005004  
 4774 021160 010203  
 4775 021162 001404  
 4776 021164 062704 000004  
 4777 021170 005303  
 4778 021172 001374  
 4779  
 4780 021174 104401 002664  
 4781 021200 010446  
 4782 021202 062716 001732  
 4783 021206 004737 024604  
 4784 021212 004737 025014  
 4785 021216 104401 002664  
 4786 021222 010446  
 4787 021224 062716 001772  
 4788 021230 004737 024604  
 4789 021234 004737 025014  
 4790  
 4791 021240 006302  
 4792  
 4793 021242 104401 002664  
 4794 021246 016246 001652  
 4795 021252 104405  
 4796  
 4797 021254 104401 002664  
 4798 021260 016246 001632  
 4799 021264 104405  
 4800  
 4801 021266 104401 002664  
 4802 021272 016246 001712  
 4803 021276 042716 100000  
 4804 021302 104405  
 4805  
 4806 021304 104401 002664  
 4807 021310 016246 001562  
 4808 021314 104405  
 4809  
 4810 021316 005300  
 4811 021320 001304  
 4812 021322 104401 002474

:REPSTAT  
 ;THIS ROUTINE REPORTS ERROR STATISTICS AND DATA-TRANSFER STATISTICS.  
 REPSTA: TYPE MSG26  
 MOV DRVPRS,R0  
 MOV #PDR,R1  
 1\$: TYPE \$CRLF  
 MOVB (R1)+,R2  
 BIC #177770,R2  
 MOV R2,-(SP)  
 TYPOS  
 .BYTE 3  
 .BYTE 0  
 TYPE ,BLNKS3  
 CLR R4  
 MOV R2,R3  
 BEQ 3\$  
 2\$: ADD #4,R4  
 DEC R3  
 BNE 2\$  
 3\$: TYPE ,BLNKS1  
 MOV R4,-(SP)  
 ADD #N\$RTL,(SP)  
 JSR PC,\$DB20  
 JSR PC,SUPRS  
 TYPE ,BLNKS1  
 MOV R4,-(SP)  
 ADD #N\$DL,(SP)  
 JSR PC,\$DB20  
 JSR PC,SUPRS  
 ASL R2  
 TYPE ,BLNKS1  
 MOV (SECN(R2)),-(SP)  
 TYPDS  
 TYPE ,BLNKS1  
 MOV (CECN(R2)),-(SP)  
 TYPDS  
 TYPE ,BLNKS1  
 MOV (ATER(R2)),-(SP)  
 BIC #100000,(SP) :DONT TYPE A NEGATIVE NO.  
 TYPDS  
 TYPE ,BLNKS1  
 MOV (ECN(R2)),-(SP)  
 TYPDS  
 DEC R0 :FINISHED WITH THE DRIVES ?  
 BNE 1\$ :BR IF NOT  
 TYPE ,MSG26A :REST OF SUMMARY MESSAGE

F09

4813	021326	013700	001264	MOV	DRVPRS,R0	:NUMBER OF DRIVES
4814	021332	012701	001254	MOV	#PDR,R1	: 'DRIVES PRESENT' TABLE ADDRESS
4815	021336	104401	001213	45: TYPE	\$CR LF	: CR-LF
4816	021342	112102		MOVB	{R1},R2	: DRIVE ADDRESS
4817	021344	042702	177770	BIC	#177770,R2	: LEAVE ONLY DRIVE NUMBER
4818	021350	010246		MOV	R2,-(SP)	: PUT ON STACK FOR TYPEOUT
4819	021352	104403		TYPOS		: TYPE IT IN OCTAL
4820	021354	003		.BYTE	3	: TYPE 3 CHARACTERS
4821	021355	000		.BYTE	0	: SUPPRESS LEADING ZEROS
4822	021356	104401	002662	TYPE	BLNKS3	: 3 BLANKS
4823	021362	006302		ASL	R2	: CONVERT TO A WORD TABLE INDEX
4824	021364	104401	002664	TYPE	BLNKS1	
4825	021370	016246	001602	MOV	\$KECN(R2),-(SP)	
4826	021374	104405		TYPDS		
4827						
4828						
4829	021376	104401	002664	TYPE	BLNKS1	
4830	021402	016246	001672	MOV	ABORT(R2),-(SP)	
4831	021406	104405		TYPDS		
4832						
4833	021410	006202		ASR	R2	
4834	021412	104401	002664	TYPE	BLNKS1	
4835	021416	116246	001622	MOVB	\$INCN(R2),-(SP)	
4836	021422	104405		TYPDS		
4837						
4838	021424	005300		DEC	R0	
4839	021426	001343		BNE	45	
4840	021430	004737	025660	JSR	PC,TIMTYP	:TYPE THE TIME
4841	021434	000207		RTS	PC	

4842  
4843  
4844  
4845  
4846  
4847  
4848  
4849  
4850  
4851  
4852  
4853  
4854  
4855  
4856  
4857  
4858  
4859 021436 005046  
4860 021440 012746 021446  
4861 021444 000002  
4862  
4863  
4864  
4865  
4866  
4867  
4868  
4869  
4870 021446 005237 001460  
4871 021452 001456  
4872 021454 105777 157542  
4873 021460 100514  
4874  
4875 021462 005037 001466  
4876 021466 012737 177761 001470  
4877  
4878 021474 105737 001534  
4879  
4880  
4881  
4882  
4883  
4884 021500 001507  
4885 021502 005237 001466  
4886 021506 001372  
4887 021510 005237 001470  
4888 021514 001367  
4889  
4890  
4891  
4892 021516 113700 001534  
4893 021522 042700 177760  
4894 021526 010003  
4895 021530 062700 001306  
4896 021534 011037 001172  
4897 021540 042737 177770 001172

```

:STATUS
:THIS ROUTINE IS NORMALLY ENTERED WHEN THE PROGRAM IS WAITING FOR THE
:CONTROLLER TO FINISH WHAT IT IS DOING. THERE ARE TWO DOUBLE PRECISION
:COUNTS KEPT IN THIS ROUTINE.
:CICNT,CICNT1
:THIS COUNT KEEPS TRACK OF HOW LONG THE CONTROLLER HAS BEEN BUSY AFTER
:A COMMAND WAS INITIATED. THE CONTROLLER SHOULD FINISH WHATEVER IT IS DOING
:BEFORE THIS COUNT EXPIRES. IF IT DOES NOT, THERE IS AN ERROR CONDITION AND
:IT IS SO REPORTED.

:QSCNT
:THIS COUNT IS INITIALIZED EVERY TIME 8 COMMANDS ARE GENERATED. THE COUNT
:IS INCREMENTED EVERY TIME THIS ROUTINE IS ENTERED. ALL THE 8 COMMANDS
:SHOULD BE DONE BEFORE THIS COUNT EXPIRES. IF THEY DO NOT AN ERROR CONDITION
:IS REPORTED. THIS COUNT HAS BEEN KEPT PRIMARILY TO INSURE THAT THE PROGRAM
:DOES NOT GET CAUGHT IN AN INDEFINITE LOOP, BECAUSE OF AN ERROR CONDITION.

STATUS: CLR      -(SP)      ;DROP PRIORITY AND WAIT FOR INT
        MOV      #15,-(SP)  ;RETURN FOR RTI
        RTI

;NOTE THAT THE INTERRUPTS ARE ALLOWED ONLY
;AT CERTAIN PLACES IN THE PROGRAM, BECAUSE
;IT MAKES TROUBLESHOOTING OF FAILURES EASY.
;OTHER PLACES WHERE INTERRUPTS ARE ALLOWED
;TO TAKE PLACE:
; 'CHFAFN', FIRST INTERRUPT AFTER ISSUING
; A SEEK FUNCTION.

15:    INC      QSCNT
        BEQ     QEROR
        TSTB   #RKCS
        BMI     CNOBSY

        CLR     CICNT
        MOV     #-17,CICNT1

CBSY:  TSTB     INTFLG

;FOR A NON-SEEK COMAND:
;INTFLG- BIT 7 IS SET, BITS 0-3 CONTAIN
;OFFSET TO THE COMMAND KEY (FROM KEY),
;FOR WHICH THIS INTERUPT IS EXPCTD..
;WHEN THE INTERUPT OCCURS & 'INTHND' IS
;ENTERED 'INTFLG' IS CLEARED.

        BEQ     SEXIT
        INC     CICNT
        BNE     CBSY
        INC     CICNT1
        BNE     CBSY

;TIMED OUT WHILE WAITING FOR THE INTRUPT.
;ONE OF THE COMMANDS DID NOT INTERPLT

NIEROR: MOVB    INTFLG,RO
        BIC     #177760,RO
        MOV     RO,R3
        ADD     #KEY,RO
        MOV     (RO),$REG4
        BIC     #177770,$REG4
    
```

```

4898 021546 013737 001172 001250      MOV      $REG4,SRDRV      ;GET DRIVE #, FOR TYPING SERIAL #
4899
4900 021554 104421 002245      TYPMSG  ,MSG15           ;PRINT 'DRVE # DIDN'T INTRUPT AFTER'
4901 021560 016305 002032      MOV      PCMD(R3),R5
4902 021564 016504 000002      MOV      2(R5),R4
4903 021570 004737 021736      JSR      PC,TYPFN
4904 021574 004737 022032      JSR      PC,GT4RG
4905 021600 104025      ERROR    25              ;COMMAND TYPED OUT IN EROR MESSAGE DID
4906                                     ;NOT INTERRUPT ON COMPLETION.
4907 021602 052710 104000      BIS      #BIT15+BIT11,(R0) ;INDICATE THAT FUNCTION IS ABORTEC
4908 021606 000444      BR
4909
4910
4911 021610 005037 001460      QEROR:  CLR      QSCNT      ;REESTABLISH COUNT
4912 021614 004737 022032      JSR      PC,GT4RG
4913 021620 104026      ERROR    26
4914
4915                                     ;ALL 8 COMMANDS SHOULD BE DONE BY NOW, TIMED
4916                                     ;OUT. THE PROGRAM IS WAITING FOR ONE OF THE
4917                                     ;COMMANDS IN THE Q TO BE FINISHED AND THIS
4918                                     ;DID NOT HAPPEN OR FOR SOME OTHER REASON THE
4919                                     ;'FINISHED' FLAG (BIT 15) OF ONE OF THE 8
4920                                     ;COMMAND KEYS WAS NOT SET. VARIOUS FLAGS 'POS',-7
4921                                     ;'BUSY'(-7), 'KEY'(-8) CONTAIN INFORMATION
4922                                     ;ABOUT THE STATUS OF THE SYSTEM.
4922 021622 032777 020000 157310      BIT      #SW13,JSWR      ;INHIBIT TYPEOUT?
4923 021630 001024      BNE      25              ;YES
4924 021632 104401 002305      TYPE,   MSG16
4925 021636 012700 001306      MOV      #KEY,R0
4926 021642 012701 001426      MOV      #BUSY,R1
4927 021646 012702 177770      MOV      #-10,R2
4928 021652 104401 001213      15:    TYPE  $CRLF
4929 021656 012046      MOV      (R0)+,-(SP)    ;TYPE OUT CONTENTS OF ALL KEYS
4930 021660 104402      TYPCC   ;KEY-KEYS
4931 021662 104401 002662      TYPE   ,BLNKS3
4932 021666 005046      CLR     -(SP)
4933 021670 112116      MOVSB  (R1)+,(SP)      ;TYPE OUT CONTENTS OF ALL BUSY FLAGS
4934 021672 104403      TYPOS  ;BUSY-BUSY?
4935 021674 003      .BYTE  3
4936 021675 000      .BYTE  0
4937 021676 005202      INC    R2              ;DONE?
4938 021700 001364      BNE    15              ;NO
4939
4940 021702 004737 016006      25:    JSR      PC,CLRERR  ;MAKE SURE THERE IS NO HEAD MOVEMENT ON
4941                                     ;ANY DRIVE & THEN DO CONTROL RESE
4942 021706 000137 010630      JMP     BEGNEX         ;GO, BAK AND CONTINUE
4943
4944 021712 005004      CNOBSY: CLR    R4
4945 021714 005204      INC    R4
4946 021716 001376      BNE    -2
4947 021720 013746 001244      SEXIT:  MOV    PPRVL,-(SP)
4948 021724 012746 021732      MOV    #RTIPC7,-(SP)  ;RETURN FOR RTI *****
4949 021730 000002      RTI
4950
4951 021732 000137 010650      RTIPC7: JMP    QMNGER

```

```

4952 ;TYPFN
4953 ;ROUTINE TO TYPE OUT THE FUNCTION (READ,WRITE,ETC) ;R4 CONTAINS THE
4954 ;FUNCTION CODE AT THE TIME OF ENTRY.
4955 ;SW 13, IF SET INHIBITS TYPEOUT.
4956
4957 021736 032777 020000 157174 TYPFN: BIT #SW13,@SWR ;INHIBIT TYPEOUT?
4958 021744 001031 BNE 5$ ;YES
4959 021746 020427 000002 CMP R4,#2 ;WRITE?
4960 021752 001002 BNE 1$
4961 021754 104401 002133 TYPE ,MSG6
4962 021760 022704 000004 1$: CMP #4,R4 ;READ?
4963 021764 001002 BNE 2$
4964 021766 104401 002141 TYPE ,MSG7
4965 021772 022704 000012 2$: CMP #12,R4 ;READ CHECK?
4966 021776 001002 BNE 3$
4967 022000 104401 002156 TYPE ,MSG9
4968 022004 022704 000006 3$: CMP #6,R4 ;WRITE CHECK?
4969 022010 001002 BNE 4$
4970 022012 104401 002146 TYPE ,MSG8
4971 022016 022704 000010 4$: CMP #10,R4 ;SEEK?
4972 022022 001002 BNE 5$
4973 022024 104401 002201 TYPE ,MSG11
4974 022030 000207 5$: RTS PC
  
```



J09

```

4975 ;GT4RG
4976 ;GET CONTENTS OF RKCS, RKER, RKDS, RKDAA
4977
4978 022032 017737 157172 001170 GT4RG: MOV @RKDA,$REG3
4979 022040 017737 157156 001162 GT3RG: MOV @RKCS,$REG0
4980 022046 017737 157146 001164 MOV @RKER,$REG1
4981 022054 017737 157136 001166 MOV @RKDS,$REG2
4982 022062 000207 RTS PC
4983
4984
4985
4986
4987
4988
4989

```

```

;GETINF
;THIS ROUTINE GETS CONTENTS OF RKCE, RKER, RKDS. THEN IT BREAKS DOWN THE
;CONTENTS OF RKDA INTO ITS COMPONENT: CYLINDER, SECTOR, SURFACE AND DRIVE
;NUMBER.

```

```

4990 022064 004737 022040 GETINF: JSR PC,GT3RG
4991 022070 010046 MOV RO,-(SP)
4992 022072 010146 MOV R1,-(SP)
4993 022074 010246 MOV R2,-(SP)
4994 022076 012700 001200 MOV #REG6+2,R0
4995 022102 017701 157122 MOV @RKDA,R1
4996 022106 010102 MOV R1,R2
4997 022110 042702 177760 BIC #177760,R2
4998 022114 010240 MOV R2,-(R0)
4999 022116 006201 ASR R1
5000 022120 006201 ASR R1
5001 022122 006201 ASR R1
5002 022124 006201 ASR R1
5003 022126 010102 MOV R1,R2
5004 022130 042702 177776 BIC #177776,R2
5005 022134 010240 MOV R2,-(R0)
5006 022136 006201 ASR R1
5007 022140 010102 MOV R1,R2
5008 022142 042702 177400 BIC #177400,R2
5009 022146 010240 MOV R2,-(R0)
5010 022150 000301 SWAB R1
5011 022152 042701 177770 BIC #177770,R1
5012 022156 010140 MOV R1,-(R0)
5013 022160 012602 MOV (SP)+,R2
5014 022162 012601 MOV (SP)+,R1
5015 022164 012600 MOV (SP)+,R0
5016 022166 000207 RTS PC

```

K09

```

5017 ;CROTFL
5018 ;CALL: MOV #NO, -(SP) ;PUSH NO. TO BE ROTATED ON STACK
5019 ; JSR PC, CROTFL
5020 ;THIS ROUTINE ROTATES BITS 15, 14, 13 OF A WORD INTO BITS 2, 1, 0. THE
5021 ;REST OF THE BITS OF THE ROTATED WORD ARE CLEARED.
5022
5023
5024 022170 042766 017777 000002 CROTFL: BIC #17777, 2(SP)
5025 022176 000241 CLC
5026 022200 006166 000002 ROL 2(SP)
5027 022204 006166 000002 ROL 2(SP)
5028 022210 006166 000002 ROL 2(SP)
5029 022214 006166 000002 ROL 2(SP)
5030 022220 000207 RTS PC
5031
5032
5033 ;RG4SDRV
5034 ;CALL: JSR PC, RG4SDRV
5035 ;THIS ROUTINE GETS THE CONTENTS OF RKDS, RKER, RKCS, RKDA. THEN
5036 ;IT SAVES THE DRIVE NUMBER FROM RKDA IN 'SRDRV'.
5037 022222 004737 022032 RG4SDR: JSR PC, GT4RG ;GET RKCS, ER, DS, DA
5038
5039
5040 ;GTSDRV
5041 ;CALL: JSR PC, GTSDRV
5042 ;THIS ROUTINE EXTRACTS THE DRIVE # FROM RKDA (BITS 15,14,13) AND SAVES
5043 ;IT IN "SRDRV" (BITS 0,1,2)
5044
5045 022226 017746 156776 GTSDRV: MOV #RKDA, -(SP) ;GET BITS 15,14,13 FROM RKDA
5046 022232 004737 02217C JSR PC, CROTFL
5047 022236 012637 001250 MOV (SP)+, SRDRV ;SAVE THE DRIVE #
5048 022242 000207 RTS PC
    
```

109

```

5049          .SBTTL  DRV.RESET - DRIVE RESET ROUTINE
5050          ;DRV.RESET - DRIVE RESET ROUTINE
5051          ;IF R/W/S RDY DOES NOT SET WITHIN A CERTAIN TIME OF DOING DRIVE RESET
5052          ;AN ERROR IS REPORTED.
5053
5054          022244 005037 022354          DR.RST: CLR          TIMOUT
5055          022250 013777 001502 156752          MOV          QDRV, @RKDA
5056          022256 012777 000015 156736          MOV          #15, @RKCS
5057          022264 104417          CON.RDY
5058          022266 032777 000100 156722 1$:          BIT          #100, @RKDS          ;DID R/W/S RDY SET?
5059          022274 001026          BNE          2$          ;YES
5060          022276 012746 177760          MOV          #-20, -(SP)          ;NO, WAIT FOR R/W/S
5061          022302 005216          INC          (SP)
5062          022304 001376          BNE          -2
5063          022306 005726          TST          (SP)+
5064          022310 005237 022354          INC          TIMOUT
5065          022314 001364          BNE          1$
5066          022316 032777 020000 156614          BIT          #SW13, @SWR          ;INHIBIT TYPEOUT?
5067          022324 001012          BNE          2$          ;YES
5068          022326 104401 001213          TYPE          , $CRLF          ;TIMED OUT, R/W/S RDY DID NOT SET
5069          022332 104401 027746          TYPE          , EM4          ;REPORT ERROR
5070          022336 104401 002206          TYPE          , MSG12
5071          022342 011646          MOV          (SP), -(SP)
5072          022344 162716 000002          SUB          #2, (SP)
5073          022350 104402          TYP0C
5074          022352 000002          2$:          RTI
5075          022354 000000          TIMOUT: 0
  
```

```

5076 SBTTL CON.RESET - CONTROL RESET ROUTINE
5077 ;CON.RESET
5078 ;CONTROL RESET ROUTINE
5079 ;CON.RDY
5080 ;CONTROL READY ROUTINE
5081
5082 022356 012777 000001 156636 CN.RST: MOV #1,ARKCS
5083 022364 005037 001472 CN.RDY: CLR TIMER
5084 022370 105777 '56626 1$: TSTB ARKCS ;DID CONTROL RDY SET?
5085 022374 100451 BMI 2$ ;YES
5086 022376 012746 177750 MOV #30,-(SP) ;WAIT FOR CNTRL RDY
5087 022402 005216 INC (SP)
5088 022404 001376 BNE -2
5089 022406 005726 TST (SP)+
5090 022410 005237 001472 INC TIMER
5091 022414 001365 BNE 1$
5092 022416 032777 020000 156514 BIT #SW13,ASWR ;INHIBIT TYPEOUT?
5093 022424 001035 BNE 2$ ;YES
5094 022426 104401 002206 TYPE MSG12 ;CNTRL RDY DID NOT SET, REPORT ERROR
5095 022432 011646 MOV (SP),-(SP)
5096 022434 162716 000002 SUB #2,(SP)
5097 022440 104402 TYPOC
5098 022442 104401 022450 TYPE ,65$ ;:TYPE ASCIZ STRING
5099 022446 000421 BR 64$ ;:GET OVER THE ASCIZ
5100 ;:65$: .ASCIZ '<15><12>/CONTROLLER NOT READY - ARKCS='
5101 64$:
5102 022512 017746 156504 MOV ARKCS,-(SP)
5103 022516 104402 TYPOC
5104 022520 000002 RTI

```

```

5105 .SBTTL TYPMSG - TYPE MESSAGE ROUTINE (SW13)
5106 ;TYPMSG
5107 ;THIS ROUTINE IS USED FOR MESSAGE TYPEOUTS. IF SW 13 IS SET THE TYPEOUT
5108 ;IS SKIPPED.
5109 ;CALL: TYPMSG
5110 ; POINTER ; POINTER TO THE ASCII MESSAGE STRING
5111
5112 022522 032777 020000 156410 TY.MSG: BIT #SW13,2SWR ;INHIBIT TYPEOUT?
5113 022530 001005 BNE 2$ ;YES
5114 022532 017637 000000 022542 MOV @2(SP),1$ ;GET POINTER TO ASCII STRING
5115 022540 104401 TYPE
5116 022542 000000 1$: .WORD 0
5117
5118 022544 062716 000002 2$: ADD #2,(SP) ;ADJUST RETURN ADDRESS TO SKIP OVER POINTER
5119 022550 000002 RTI

```

```

5120 .SBTTL KWSRVE - KWILL CLOCK SERVICE ROUTINE
5121 ; THIS ROUTINE SERVICES THE INTERRUPT FROM THE KWILL LINE CLOCK
5122 ; AND KEEPS TRACK OF ELAPSED TIME.
5123 ; KWCOUNT- CONTAINS CYCLES (PER SECOND) (2'S COMPLEMENT)
5124 ; KWSEC- CONTAINS SECONDS (2'S COMPLEMENT)
5125 ; KWMIN- CONTAINS MINUTES (2'S COMPLEMENT)
5126 ; KWHR- CONTAINS HOURS (2'S COMPLEMENT)
5127
5128 022552 005237 001560 KWSRVE: INC KWCOUNT ;COUNT 60 CPS
5129 022556 001401 BEQ 1$ ;OVERFLOWED?
5130 022560 000002 RTI
5131 022562 012737 177704 001560 1$: MOV #-60.,KWCOUNT ;RESET 60 CPS COUNT
5132 022570 005237 001556 INC KWSEC ;COUNT SECONDS
5133 022574 001401 BEQ 2$ ;OVERFLOWED?
5134 022576 000002 RTI ;RETURN
5135 022600 012737 177704 001556 2$: MOV #-60.,KWSEC ;RESET "SECONDS" COUNT
5136 022606 005237 001554 INC KWMIN ;COUNT MINUTES
5137 022612 001005 BNE 3$ ;OVERFLOWED?
5138 022614 012737 177704 001554 MOV #-60.,KWMIN ;RESET "MINUTES" COUNT
5139 022622 005237 001552 INC KWHR ;COUNT HOURS
5140
5141 022626 000002 3$: RTI ;RETURN
5142
5143 ;WATIME
5144 ;ROUTINE PROVIDES SOME WAITING TIME.
5145
5146
5147 022630 013746 022652 WATIME: MOV 2$,-(SP) ;COUNTER VALUE
5148 022634 005237 022654 1$: INC 3$ ;COUNT
5149 022640 001375 BNE 1$ ;HANG IN THERE UNTIL COUNT WRAPS AROUND
5150 022642 005216 INC (SP) ;COUNT AGAIN
5151 022644 001373 BNE 1$ ;GO THROUGH MINOR LOOP AGAIN
5152 022646 005726 TST (SP)+ ;RESTORE THE STACK POINTER
5153 022650 000207 RTS PC ;RETURN
5154 022652 177730 2$: .WORD 177730 ;VALUE FOR APPROX 15 SEC DELAY
5155 022654 000000 3$: .WORD 0 ;'MINOR' LOOP COUNTER
    
```

S156  
 S157  
 S158  
 S159  
 S160  
 S161  
 S162  
 S163  
 S164  
 S165  
 S166  
 S167  
 S168  
 S169  
 S170  
 S171

022656 005737 001264  
 022662 001401  
 022664 000207  
 022666 104401 002225  
 022672 004737 022630  
 022676 012706 00110C  
 022702 000400

```

:CHPDRS
:THIS ROUTINE CHECKS IF THERE ANY DRIVES PRESENT (ON LINE), IF THERE
:ARE, A RETURN IS MADE. IF THERE ARE NONE PRESENT, A MESSAGE IS PRINTED OUT.
:THE STACK POINTER IS RE-INITIATED TO 1100 AND CONTROL IS TRANSFERRED
:TO THE END OF PASS ROUTINE, $EOP. BEFORE PASSING CONTROL TO $EOP, SOME
:TIME IS KILLED (WATIME), THIS IS DONE TO KEEP THE NUMBER OF MESSAGES
:(END OF PASS #X) TO A SMALL AMOUNT.

CHDPRS: TST      DRVPRS      ;ANY DRIVES PRESENT?
        BEQ      1$          ;NO
        RTS      PC          ;YES, EXIT
1$:     TYPE     MSG14       ;NO, GIVE A MESSAGE
        JSR      PC,WATIME  ;KILL SOME TIME
        MOV      #STACK,SP  ;REINITIALIZE STACK
        BR       $EOP       ;GO TO END OF PASS ROUTINE
    
```

```

5172          .SBTTL  END OF PASS ROUTINE
5173
5174          ;*****
5175          ;*INCREMENT THE PASS NUMBER ($PASS)
5176          ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
5177          ;*IF THERES A MONITOR GO TO IT
5178          ;*IF THERE ISN'T JUMP TO BEGNEX
5179
5180          $EOP:
5181          022704 000004          SCOPE
5182          022706 005037 001102  CLR          $STNM          ;; ZERO THE TEST NUMBER
5183          022712 005237 001100  INC          $PASS          ;; INCREMENT THE PASS NUMBER
5184          022716 042737 100000 001100  BIC          #100000,$PASS ;; DON'T ALLOW A NEG. NUMBER
5185          022724 005327          DEC          (PC)+          ;; LOOP?
5186          022726 000001          $EOPCT: .WORD 1
5187          022730 003013          BGT          $DOAGN          ;; YES
5188          022732 012737          MOV          (PC)+,2(PC)+  ;; RESTORE COUNTER
5189          022734 000001          $ENDCT: .WORD 1
5190          022736 022726          $GET42: $EOPCT
5191          022740 013700 000042  MOV          2#42,R0          ;; GET MONITOR ADDRESS
5192          022744 001405          BEQ          $DOAGN          ;; BRANCH IF NO MONITOR
5193          022746 000005          RESET
5194          022750 004710          $ENDAD: JSR          PC,(R0)  ;; GO TO MONITOR
5195          022752 000240          NOP          ;; SAVE ROOM
5196          022754 000240          NOP          ;; FOR
5197          022756 000240          NOP          ;; ACT11
5198          022760          $DOAGN:
5199          022760 000137          JMP          2(PC)+          ;; RETURN
5200          022762 010630          $RTNAD: .WORD  BEGNEX
    
```



FIU

```

5201 .SBTTL TTY INPUT ROUTINE
5202
5203 ::*****
5204 .ENABL LSB
5205
5206 ::*****
5207 *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
5208 *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
5209 *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
5210 *WHEN OPERATING IN TTY FLAG MODE.
5211 022764 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;; IS THE SOFT-SWR SELECTED?
5212 022772 001074 BNE 15$ ;; BRANCH IF NO
5213 022774 105777 156144 TSTB @STKS ;; CHAR THERE?
5214 023000 100071 BPL 15$ ;; IF NO, DON'T WAIT AROUND
5215 023002 117746 156140 MOVB @STKB,-(SP) ;; SAVE THE CHAR
5216 023006 042716 177600 BIC #1C177,(SP) ;; STRIP-OFF THE ASCII
5217 023012 022726 000007 CMP #7,(SP)+ ;; IS IT A CONTROL G?
5218 023016 001062 BNE 15$ ;; NO, RETURN TO USER
5219 023020 123727 001134 000001 CMPB $AUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?
5220 023026 001456 BEQ 15$ ;; BRANCH IF YES
5221
5222 023030 104401 023511 $GTSWR: TYPE , $CNTLG ;; ECHO THE CONTROL-G (↑G)
5223 023034 104401 023516 TYPE $MSWR ;; TYPE CURRENT CONTENTS
5224 023040 013746 000176 MOV SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
5225 023044 104402 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS,
5226 023046 104401 023527 TYPE , $MNEW ;; PROMPT FOR NEW SWR
5227 023052 005046 19$: CLR -(SP) ;; CLEAR COUNTER
5228 023054 005046 7$: CLR -(SP) ;; THE NEW SWR
5229 023056 105777 156062 TSTB @STKS ;; CHAR THERE?
5230 023062 100375 BPL 7$ ;; IF NOT TRY AGAIN
5231
5232 023064 117746 156056 MOVB @STKB,-(SP) ;; PICK UP CHAR
5233 023070 042716 177600 BIC #1C177,(SP) ;; MAKE IT 7-BIT ASCII
5234
5235
5236
5237 023074 021627 000025 9$: CMP (SP),#25 ;; IS IT A CONTROL-U?
5238 023100 001005 BNE 10$ ;; BRANCH IF NOT
5239 023102 104401 023504 TYPE , $CNTLU ;; YES, ECHO CONTROL-U (↑U)
5240 023106 062706 000006 20$: ADD #6,SP ;; IGNORE PREVIOUS INPUT
5241 023112 000757 BR 19$ ;; LET'S TRY IT AGAIN
5242
5243
5244 023114 021627 000015 10$: CMP (SP),#15 ;; IS IT A <CR>?
5245 023120 001022 BNE 16$ ;; BRANCH IF NO
5246 023122 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?
5247 023126 001403 BEQ 11$ ;; BRANCH IF YES
5248 023130 016677 000002 156002 MOV 2(SP),@SWR ;; SAVE NEW SWR
5249 023136 062706 000006 11$: ADD #6,SP ;; CLEAR UP STACK
5250 023142 104401 001213 14$: TYPE $CRLF ;; ECHO <CR> AND <LF>
5251 023146 123727 001135 000001 CMPB $INTAG,#1 ;; RE-ENABLE TTY KBD INTERRUPTS?
5252 023154 001003 BNE 15$ ;; BRANCH IF NOT
5253 023156 012777 000100 155760 MOV #100,@STKS ;; RE-ENABLE TTY KBD INTERRUPTS
5254 023164 000002 15$: RTI ;; RETURN
5255 023166 004737 024414 16$: JSR PC,$TYPEC ;; ECHO CHAR
5256 023172 021627 000060 CMP (SP),#60 ;; CHAR < 0?
    
```

```

5257 023176 002420          BLT    18$          ;; BRANCH IF YES
5258 023200 021627 000067    CMP    (SP),#67    ;; CHAR > ??
5259 023204 003015          BGT    18$          ;; BRANCH IF YES
5260 023206 042726 000060    BIC    #60,(SP)+   ;; STRIP-OFF ASCII
5261 023212 005766 000002    TST    2(SP)       ;; IS THIS THE FIRST CHAR
5262 023216 001403          BEQ    17$          ;; BRANCH IF YES
5263 023220 006316          ASL    (SP)        ;; NO, SHIFT PRESENT
5264 023222 006316          ASL    (SP)        ;; CHAR OVER TO MAKE
5265 023224 006316          ASL    (SP)        ;; ROOM FOR NEW ONE.
5266 023226 005266 000002    17$: INC    2(SP)    ;; KEEP COUNT OF CHAR
5267 023232 056616 177776    BIS    -2(SP),(SP) ;; SET IN NEW CHAR
5268 023236 000707          BR     7$          ;; GET THE NEXT ONE
5269 023240 1044C1 001212    18$: TYPE $QUES    ;; TYPE ?<CR><LF>
5270 023244 000720          BR     20$        ;; SIMULATE CONTROL-U
5271          .DSABL    LSB
5272
5273
5274          ;; *****
5275          ;; *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
5276          ;; *CALL:
5277          ;; *   RDCHR          ;; INPUT A SINGLE CHARACTER FROM THE TTY
5278          ;; *   RETURN HERE    ;; CHARACTER IS ON THE STACK
5279          ;; *                   ;; WITH PARITY BIT STRIPPED OFF
5280
5281
5282 023246 011646          $RDCHR: MOV    (SP),-(SP) ;; PUSH DOWN THE PC
5283 023250 016666 000004 000002    MOV    4(SP),2(SP) ;; SAVE THE PS
5284 023256 105777 155662    1$: TSTB 2$TKS      ;; WAIT FOR
5285 023262 100375          BPL    1$          ;; A CHARACTER
5286 023264 117766 155656 000004    MOVB  2$TKB,4(SP)  ;; READ THE TTY
5287 023272 042766 177600 000004    BIC    #1C<177>,4(SP) ;; GET RID OF JUNK IF ANY
5288 023300 026627 000004 000023    CMP    4(SP),#23   ;; IS IT A CONTROL-S?
5289 023306 001013          BNE    3$          ;; BRANCH IF NO
5290 023310 105777 155630    2$: TSTB 2$TKS      ;; WAIT FOR A CHARACTER
5291 023314 100375          BPL    2$          ;; LOOP UNTIL ITS THERE
5292 023316 117746 155624    MOVB  2$TKB,-(SP)  ;; GET CHARACTER
5293 023322 042716 177600          BIC    #1C177,(SP) ;; MAKE IT 7-BIT ASCII
5294 023326 022627 000021    CMP    (SP)+,#21   ;; IS IT A CONTROL-Q?
5295 023332 001366          BNE    2$          ;; IF NOT DISCARD IT
5296 023334 000750          BR     1$          ;; YES, RESUME
5297 023336 026627 000004 000140    3$: CMP    4(SP),#140 ;; IS IT UPPER CASE?
5298 023344 002407          BLT    4$          ;; BRANCH IF YES
5299 023346 026627 000004 000175    CMP    4(SP),#175  ;; IS IT A SPECIAL CHAR?
5300 023354 003003          BGT    4$          ;; BRANCH IF YES
5301 023356 042766 000040 000004    BIC    #40,4(SP)   ;; MAKE IT UPPER CASE
5302 023364 000002          4$: RTI           ;; GO BACK TO USER
5303          ;; *****
5304          ;; *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
5305          ;; *CALL:
5306          ;; *   RDLIN          ;; INPUT A STRING FROM THE TTY
5307          ;; *   RETURN HERE    ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
5308          ;; *                   ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
5309
5310 023366 010346          $RDLIN: MOV    R3,-(SP) ;; SAVE R3
5311 023370 012703 023474    1$: MOV    #1TTYIN,R3 ;; GET ADDRESS
5312 023374 022703 023504    2$: CMP    #1TTYIN+8.,R3 ;; BUFFER FULL?
    
```

5313	023400	101405				BLOS	4\$		:: BR IF YES
5314	023402	104410				RDCHR			:: GO READ ONE CHARACTER FROM THE TTY
5315	023404	112613				MOVB	(SP)+, (R3)		:: GET CHARACTER
5316	023406	122713	000177		10\$:	CMPB	#177, (R3)		:: IS IT A RUBOUT
5317	023412	001003				BNE	3\$		:: SKIP IF NOT
5318	023414	104401	001212			TYPE	\$QUES		:: TYPE A '?'
5319	023420	000763				BR	1\$		:: CLEAR THE BUFFER AND LOOP
5320	023422	111337	023472		3\$:	MOVB	(R3), 9\$		:: ECHO THE CHARACTER
5321	023426	104401	023472			TYPE	9\$		
5322	023432	122723	000015			CMPB	#15, (R3)+		:: CHECK FOR RETURN
5323	023436	001356				BNE	2\$		:: LOOP IF NOT RETURN
5324	023440	105063	177777			CLRB	-1(R3)		:: CLEAR RETURN (THE 15)
5325	023444	104401	001214			TYPE	\$LF		:: TYPE A LINE FEED
5326	023450	012603				MOV	{SP)+, R3		:: RESTORE R3
5327	023452	011646				MOV	(SP), -(SP)		:: ADJUST THE STACK AND PUT ADDRESS OF THE
5328	023454	016666	000004	000002		MOV	4(SP), 2(SP)		:: FIRST ASCII CHARACTER ON IT
5329	023462	012766	023474	000004		MOV	#STTYIN, 4(SP)		
5330	023470	000002				RTI			:: RETURN
5331	023472	000			9\$:	.BYTE	0		:: STORAGE FOR ASCII CHAR. TO TYPE
5332	023473	000				.BYTE	0		:: TERMINATOR
5333	023474	000010				\$TTYIN:	.BLKB	8.	:: RESERVE 8 BYTES FOR TTY INPUT
5334	023504	052536	005015	000		\$CNTLU:	.ASCIZ	/↑U/<15><12>	:: CONTROL "U"
5335	023511	136	006507	000012		\$CNTLG:	.ASCIZ	/↑G/<15><12>	:: CONTROL "G"
5336	023516	005015	053523	020122		\$MSWR:	.ASCIZ	<15><12>/SWR = /	
5337	023524	020075	000						
5338	023527	040	047040	053505		\$MNEW:	.ASCIZ	/ NEW = /	
5339	023534	036440	000040						

H10

5340  
5341  
5342  
5343  
5344  
5345  
5346  
5347  
5348  
5349  
5350  
5351  
5352  
5353  
5354  
5355  
5356  
5357  
5358  
5359  
5360  
5361  
5362  
5363  
5364  
5365  
5366  
5367  
5368  
5369  
5370  
5371  
5372  
5373  
5374  
5375  
5376  
5377

023540 011646  
023542 016666 000004 000002  
023550 010046  
023552 010146  
023554 010246  
023556 104411  
023560 012600  
023562 005001  
023564 005002  
023566 112046  
023570 001412  
023572 006301  
023574 006102  
023576 006301  
023600 006102  
023602 006301  
023604 006102  
023606 042716 177770  
023612 062601  
023614 000764  
023616 005726  
023620 010166 000012  
023624 010237 023640  
023630 012602  
023632 012601  
023634 012600  
023636 000002  
023640 000000

```

.SBTTL READ AN OCTAL NUMBER FROM THE TTY
:*****
:THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
:CHANGE IT TO BINARY.
:CALL:
:*      RDOCT          ;; READ AN OCTAL NUMBER
:*      RETURN HERE   ;; LOW ORDER BITS ARE ON TOP OF THE STACK
:*                  ;; HIGH ORDER BITS ARE IN $HIOCT
$RDOCT: MOV      (SP),-(SP) ;; PROVIDE SPACE FOR THE
        MOV      4(SP),2(SP) ;; INPUT NUMBER
        MOV      R0,-(SP)   ;; PUSH R0 ON STACK
        MOV      R1,-(SP)   ;; PUSH R1 ON STACK
        MOV      R2,-(SP)   ;; PUSH R2 ON STACK
1$:     ROLIN      (SP)+,R0  ;; READ AN ASCII LINE
        MOV      (SP)+,R0   ;; GET ADDRESS OF 1ST CHARACTER
        CLR      R1         ;; CLEAR DATA WORD
        CLR      R2
2$:     MOVB      (R0)+,-(SP) ;; PICKUP THIS CHARACTER
        BEQ      3$        ;; IF ZERO GET OUT
        ASL      R1         ;; *2
        ROL      R2
        ASL      R1         ;; *4
        ROL      R2
        ASL      R1         ;; *8
        ROL      R2
3$:     BIC      #1C7,(SP)   ;; STRIP THE ASCII JUNK
        ADD      (SP)+,R1   ;; ADD IN THIS DIGIT
        BR       2$        ;; LOOP
        TST      (SP)+      ;; CLEAN TERMINATOR FROM STACK
        MOV      R1,12(SP)  ;; SAVE THE RESULT
        MOV      R2,$HIOCT
        MOV      (SP)+,R2   ;; POP STACK INTO R2
        MOV      (SP)+,R1   ;; POP STACK INTO R1
        MOV      (SP)+,R0   ;; POP STACK INTO R0
        RTI                ;; RETURN
$HIOCT: .WORD      0        ;; HIGH ORDER BITS GO HERE
    
```

.SBTTL READ A DECIMAL NUMBER FROM THE TTY

5378									
5379									
5380									
5381									
5382									
5383									
5384									
5385									
5386									
5387									
5388									
5389									
5390									
5391									
5392	023642	011646							
5393	023644	016666	000004	000002					
5394	023652	010046							
5395	023654	010146							
5396	023656	010246							
5397	023660	104411							
5398	023662	012600							
5399	023664	010037	024010						
5400	023670	005046							
5401	023672	005002							
5402	023674	122710	000055						
5403	023700	001001							
5404	023702	112002							
5405	023704	112001							
5406	023706	001424							
5407	023710	122701	000060						
5408	023714	003032							
5409	023716	122701	000071						
5410	023722	002427							
5411	023724	032716	170000						
5412	023730	001024							
5413	023732	006316							
5414	023734	011646							
5415	023736	006316							
5416	023740	006316							
5417	023742	062616							
5418	023744	102416							
5419	023746	162701	000060						
5420	023752	060116							
5421	023754	102412							
5422	023756	000752							
5423	023760	005702							
5424	023762	001401							
5425	023764	005416							
5426	023766	012666	000012						
5427	023772	012602							
5428	023774	012601							
5429	023776	012600							
5430	024000	000002							
5431									
5432	024002	005726							
5433	024004	105010							

```

*****
: THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
: CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
: ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
: THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
: USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
: POSITIVE 32767 TO NEGATIVE 32768.
: CALL:
: *      RODEC                ;; READ A DECIMAL NUMBER
: *      RETURN HERE          ;; NUMBER IS ON TOP OF THE STACK
:
SRDEEC: MOV      (SP) -(SP)      ;; PROVIDE SPACE FOR
        MOV      4(SP), 2(SP)   ;; THE INPUT NUMBER
        MOV      R0, -(SP)      ;; PUSH R0 ON STACK
        MOV      R1, -(SP)      ;; PUSH R1 ON STACK
        MOV      R2, -(SP)      ;; PUSH R2 ON STACK
1$:     RDLIN      (SP)+, R0      ;; READ AN ASCII LINE
        MOV      R0, 6$         ;; ADDRESS OF 1ST CHAR.
        CLR      -(SP)          ;; SAVE INCASE OF BAD INPUT
        CLR      R2             ;; CLEAR DATA WORD
        CMPB     #'-, (R0)      ;; SIGN SET POSITIVE
        BNE     2$              ;; SEE IF A MINUS SIGN WAS TYPED
        MOVB    (R0)+, R2       ;; BR IF NO MINUS SIGN
        MOVB    (R0)+, R1       ;; SAVE FOR LATER USE
2$:     BEQ      3$              ;; PICKUP THIS CHARACTER
        CMPB    #'0, R1         ;; GET OUT IF ZERO
        BGT     5$              ;; MAKE SURE THIS CHARACTER
        CMPB    #'9, R1         ;; IS A DIGIT BETWEEN 0 & 9
        BLT     5$
        BIT     #'C7777, (SP)   ;; DON'T LET NUMBER GET TO BIG
        BNE     5$              ;; BR IF NUMBER WOULD OVERFLOW
        ASL     (SP)             ;; *2
        MOV     (SP), -(SP)     ;; SAVE FOR LATER
        ASL     (SP)             ;; *4
        ASL     (SP)             ;; *8
        ADD     (SP)+, (SP)     ;; *10.
        BVS     5$              ;; OVERFLOW ISN'T ALLOWED
        SUB     #'0, R1         ;; STRIP AWAY THE ASCII JUNK
        ADD     R1, (SP)        ;; ADD IN THIS DIGIT
        BVS     5$              ;; OVERFLOW ISN'T ALLOWED
        BR     2$              ;; LOOP
3$:     TST      R2              ;; CHECK IF NUMBER IS NEG
        BEQ     4$              ;; BR IF NO
        NEG     (SP)            ;; YES--NEGATE THE NUMBER
        MOV     (SP)+, 12(SP)   ;; SAVE THE RESULT
        MOV     (SP)+, R2       ;; POP STACK INTO R2
        MOV     (SP)+, R1       ;; POP STACK INTO R1
        MOV     (SP)+, R0       ;; POP STACK INTO R0
        RTI                    ;; RETURN
5$:     TST      (SP)+          ;; CLEAN PARTIAL NUMBER FROM STACK
        CLRB    (R0)           ;; SET A TERMINATOR
    
```

5434 024006 104401  
5435 024010 000000  
5436 024012 104401 001212  
5437 024016 000720

6\$: TYPE WORD 0  
TYPE .SQUES  
BR 1\$

U10  
:: TYPE THE INPUT UP TO BAD CHAR.  
:: POINTER GOES HERE  
:: CR & LF  
:: TRY AGAIN

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

5438  
5439  
5440  
5441  
5442  
5443  
5444  
5445  
5446  
5447  
5448  
5449  
5450  
5451  
5452  
5453  
5454  
5455  
5456  
5457  
5458  
5459  
5460  
5461  
5462  
5463  
5464  
5465  
5466  
5467  
5468  
5469  
5470  
5471  
5472  
5473  
5474  
5475  
5476  
5477  
5478  
5479  
5480  
5481  
5482  
5483  
5484  
5485  
5486  
5487  
5488  
5489  
5490  
5491  
5492  
5493

024020  
024020 010046  
024022 010146  
024024 010246  
024026 010346  
024030 010546  
024032 012746 020200  
024036 016605 000020  
024042 100004  
024044 005405  
024046 112766 000055 000001  
024054 005000 1\$:  
024056 012703 024234  
112723 000040  
024066 005002 2\$:  
024070 016001 024224  
024074 160105 3\$:  
024076 002402  
024100 005202  
024102 000774  
024104 060105 4\$:  
024106 005702  
024110 001002  
024112 105716  
024114 100407  
024116 106316 5\$:  
024120 103003  
024122 116663 000001 177777  
024130 052702 000060 6\$:  
024134 052702 000040 7\$:  
024140 110223  
024142 005720  
024144 020027 000010  
024150 002746  
024152 003002  
024154 010502  
024156 000764  
024160 105726 8\$:  
024162 100003  
024164 116663 177777 177776  
024172 105013 9\$:  
024174 012605  
024176 012603  
024200 012602

```

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

$TYPDS:
MOV      R0,-(SP)      ;; PUSH R0 ON STACK
MOV      R1,-(SP)      ;; PUSH R1 ON STACK
MOV      R2,-(SP)      ;; PUSH R2 ON STACK
MOV      R3,-(SP)      ;; PUSH R3 ON STACK
MOV      R5,-(SP)      ;; PUSH R5 ON STACK
MOV      #20200,-(SP)    ;; SET BLANK SWITCH AND SIGN
MOV      20(SP),R5      ;; GET THE INPUT NUMBER
BPL      1$            ;; BR IF INPUT IS POS.
NEG      R5            ;; MAKE THE BINARY NUMBER POS.
MOVB    #'-,1(SP)      ;; MAKE THE ASCII NUMBER NEG.
1$:      CLR      R0      ;; ZERO THE CONSTANTS INDEX
MOV      #0BLK,R3      ;; SETUP THE OUTPUT POINTER
MOVB    #' ,(R3)+      ;; SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2      ;; CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1  ;; GET THE CONSTANT
3$:      SUB      R1,R5    ;; FORM THIS BCD DIGIT
BLT     4$            ;; BR IF DONE
INC     R2            ;; INCREASE THE BCD DIGIT BY 1
BR      3$
4$:      ADD      R1,R5    ;; ADD BACK THE CONSTANT
TST     R2            ;; CHECK IF BCD DIGIT=0
BNE     5$            ;; FALL THROUGH IF 0
TSTB   (SP)          ;; STILL DOING LEADING 0'S?
BMI     7$            ;; BR IF YES
5$:      ASLB    (SP)      ;; MSD?
BCC     6$            ;; BR IF NO
MOVB   1(SP),-1(R3)   ;; YES--SET THE SIGN
6$:      BIS     #'0,R2    ;; MAKE THE BCD DIGIT ASCII
7$:      BIS     #' ,R2    ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB   R2,(R3)+      ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST    (R0,+         ;; JUST INCREMENTING
CMP    R0,#10        ;; CHECK THE TABLE INDEX
BLT    2$            ;; GO DO THE NEXT DIGIT
BGT    8$            ;; GO TO EXIT
MOV    R5,R2         ;; GET THE LSD
BR     6$            ;; GO CHANGE TO ASCII
8$:      TSTB   (SP)+    ;; WAS THE LSD THE FIRST NON-ZERO?
BR     9$            ;; BR IF NO
9$:      MOVB   -1(SP),-2(R3) ;; YES--SET THE SIGN FOR TYPING
CLRB   (R3)          ;; SET THE TERMINATOR
MOV    (SP)+,R5      ;; POP STACK INTO R5
MOV    (SP)+,R3      ;; POP STACK INTO R3
MOV    (SP)+,R2      ;; POP STACK INTO R2
    
```

5494	024202	012601			MOV	(SP)+,R1	::POP STACK INTO R1
5495	024204	012600			MOV	(SP)+,R0	::POP STACK INTO R0
5496	024206	104401	024234		TYPE	.SDBLK	::NOW TYPE THE NUMBER
5497	024212	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
5498	024220	012616			MOV	(SP)+,(SP)	
5499	024222	000002			RTI		::RETURN TO USER
5500	024224	023420			\$DTBL:	10000.	
5501	024226	001750				1000.	
5502	024230	000144				100.	
5503	024232	000012				10.	
5504	024234	000004			\$DBLK:	.BLKW 4	



M10

```

5505 .SBTTL TYPE ROUTINE
5506
5507 ;*****
5508 ;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
5509 ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
5510 ;NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
5511 ;NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
5512 ;NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
5513 ;
5514 ;CALL:
5515 ;#1) USING A TRAP INSTRUCTION
5516 ;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
5517 ;*OR
5518 ;* TYPE
5519 ;* MESADR
5520 ;*
5521
5522 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
5523 024244 105737 001157 BPL 1$ ;; BR IF YES
5524 024250 100002 HALT ;; HALT HERE IF NO TERMINAL
5525 024252 000000 BR 3$ ;; LEAVE
5526 024254 000407 1$: MOV RD, -(SP) ;; SAVE RD
5527 024256 010046 000002 MOV 22(SP), RD ;; GET ADDRESS OF ASCIZ STRING
5528 024260 017600 2$: MOVB (RD)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
5529 024264 112046 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
5530 024266 001005 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
5531 024270 005726 60$: MOV (SP)+, RD ;; RESTORE RD
5532 024272 012600 3$: ADD #2, (SP) ;; ADJUST RETURN PC
5533 024274 062716 RTI ;; RETURN
5534 024300 000002 4$: CMPB #HT, (SP) ;; BRANCH IF <HT>
5535 024302 122716 BEQ 8$
5536 024306 001430 5$: CMPB #CRLF, (SP) ;; BRANCH IF NOT <CRLF>
5537 024310 122716 BNE 5$
5538 024314 001006 TST (SP)+ ;; POP <CR><LF> EQUIV
5539 024316 005726 TYPE ;; TYPE A CR AND LF
5540 024320 104401 $CRLF
5541 024322 001213 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
5542 024324 105037 024460 BR 2$ ;; GET NEXT CHARACTER
5543 024326 000755 5$: JSR PC, $TYPEPC ;; GO TYPE THIS CHARACTER
5544 024328 004737 024414 6$: CMPB $FILLC, (SP)+ ;; IS IT TIME FOR FILLER CHARS.?
5545 024330 000755 BNE 2$ ;; IF NO GO GET NEXT CHAR.
5546 024332 004737 024414 MOV $NULL, -(SP) ;; GET # OF FILLER CHARS. NEEDED
5547 024334 001350 AND THE NULL CHAR.
5548 024336 013746 001.54 7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
5549 024338 105366 000001 BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
5550 024340 002770 JSR PC, $TYPEPC ;; GO TYPE A NULL
5551 024342 004737 024414 DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
5552 024344 013746 001.54 BR 7$ ;; LOOP
5553
5554 ;HORIZONTAL TAB PROCESSOR
5555
5556 024370 112716 000040 8$: MOVB #' , (SP) ;; REPLACE TAB WITH SPACE
5557 024374 004737 024414 9$: JSR PC, $TYPEPC ;; TYPE A SPACE
5558 024400 132737 000007 024460 BITB #7, $CHARCNT ;; BRANCH IF NOT AT
5559 024406 001372 BNE 9$ ;; TAB STOP
5560 024410 005726 TST (SP)+ ;; POP SPACE OFF STACK

```

5561	024412	000724			BR	25	;; GET NEXT CHARACTER
5562	024414	105777	154530		\$TYPEC: TSTB	\$STPB	;; WAIT UNTIL PRINTER IS READY
5563	024420	100375			BPL	\$TYPEC	
5564	024422	116677	000002	154522	MOVB	2(SP), 2\$STPB	;; LOAD CHAR TO BE TYPED INTO DATA REG.
5565	024430	122766	000015	000002	CMPB	#CR, 2(SP)	;; IS CHARACTER A CARRIAGE RETURN?
5566	024436	001003			BNE	1\$	;; BRANCH IF NO
5567	024440	105037	024460		CLRB	\$CHARCNT	;; YES--CLEAR CHARACTER COUNT
5568	024444	000406			BR	\$TYPEX	;; EXIT
5569	024446	122766	000012	000002	1\$: CMPB	#LF, 2(SP)	;; IS CHARACTER A LINE FEED?
5570	024454	001402			BEQ	\$TYPEX	;; BRANCH IF YES
5571	024456	105227			INCB	(PC)+	;; COUNT THE CHARACTER
5572	024460	000000			\$CHARCNT: WORD	0	;; CHARACTER COUNT STORAGE
5573	024462	000207			\$TYPEX: RTS	PC	
5574							

```

5575 .SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
5576
5577
5578 ::*****
5579 ::THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
5580 ::UNSIGNED OCTAL ASCII NUMBER.
5581 ::CALL
5582 ::      MOV      @PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
5583 ::      JSR      PC, @SDB20      ;; CALL THE ROUTINE
5584 ::      RETURN     ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
5585
5586 SDB20: SAVREG      ;; SAVE ALL REGISTERS
5587      MOV      2(SP), R1      ;; PICKUP THE POINTER TO LOW WORD
5588      MOV      @SDB20+13, R5   ;; POINTER TO DATA TABLE
5589      MOV      #12, R4      ;; DO ELEVEN CHARACTERS
5590      MOV      #17, R3      ;; MASK
5591      MOV      (R1)+, R0     ;; LOWER WORD
5592      MOV      (R1)+, R1     ;; HIGH WORD
5593      CLR      R2          ;; TERMINATOR
5594 1$:      MOVB   R2, -(R5)    ;; PUT CHARACTER IN DATA TABLE
5595      MOV      R0, R2      ;; GET THIS DIGIT
5596      DEC      R4          ;; COUNT THIS CHARACTER
5597      BGT     3$          ;; BR IF NOT THE LAST DIGIT
5598      BEQ     2$          ;; BR IF IT IS THE LAST DIGIT
5599      INC      R5          ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5600      MOV      R5, 2(SP)    ;; ASCII CHAR. & PUT IT ON THE STACK
5601      RESREG     ;; RESTORE ALL REGISTERS
5602      RTS      PC          ;; RETURN TO USER
5603 2$:      ASR      R3          ;; POSITION THE MASK FOR THE LAST DIGIT
5604 3$:      ROR      R1          ;; POSITION THE BINARY NUMBER FOR
5605      ROR      R0          ;; THE NEXT OCTAL DIGIT
5606      ROR      R1
5607      ROR      R0
5608      ROR      R0
5609      ROR      R0
5610      BIC     R3, R2      ;; MASK OUT ALL JUNK
5611      ADD     #'0, R2     ;; MAKE THIS CHAR. ASCII
5612      BR      1$          ;; GO PUT IT IN THE DATA TABLE
5613      $OCTVL: .BLKB 14.  ;; RESERVE DATA TABLE
    
```

```

5614 .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
5615
5616 *****
5617 *THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
5618 *DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
5619 *POSITIVE.
5620 *CALL
5621 *      MOV      #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
5622 *      JSR      PC, @#SDB2D      ;; THE FIRST ADDRESS OF ASCII
5623 *      RETURN     ;; IS ON THE STACK
5624
5625
5626
5627 $DB2D: SAVREG      ;; SAVE REGISTERS
5628 MOV      2(SP), R2      ;; PICKUP THE DATA POINTER
5629 MOV      #SDECVL, R0     ;; GET ADDRESS OF "SDECVL" STRING
5630 MOV      R0, 2(SP)      ;; PUT ADDRESS OF ASCII STRING ON STACK
5631 MOV      (R2)+, R1      ;; PICKUP THE BINARY NUMBER
5632 MOV      (R2)+, R2
5633 MOV      #10, 4$       ;; SET UP TO DO 10 CONVERSIONS
5634 MOV      #STNPNR, R4     ;; ADDRESS OF TEN POWER
5635 MOV      #STNPNR+2, R5
5636 1$: CLR      R3          ;; CLEAR PARTIAL
5637 2$: SUB      (R4), R1     ;; SUBTRACT TEN POWER
5638 SBC      R2
5639 SUB      (R5), R2
5640 BLT      3$            ;; BR IF TEN POWER TO LARGE
5641 INC      R3          ;; ADD 1 TO PARTIAL
5642 BR      2$          ;; LOOP
5643 3$: ADD      (R4)+, R1   ;; RESTORE SUBTRACTED VALUE
5644 ADC      R2
5645 ADD      (R4)+, R2
5646 CMP      (R5)+, (R5)+   ;; MOVE TO NEXT TEN POWER
5647 BIS      #'0, R3       ;; CHANGE PARTIAL TO ASCII
5648 MOV      R3, (R0)+     ;; SAVE IT
5649 DEC      (PC)+        ;; DONE?
5650 4$: .WORD    0
5651 BNE      1$          ;; BR IF NO
5652 CLRB    (R0)+        ;; TERMINATOR
5653 RESREG  ;; RESTORE REGISTERS
5654 RTS      PC         ;; RETURN
5655 $STNPNR: 145000      ;; 1.0E09
5656 35632
5657 160400             ;; 1.0E08
5658 2765
5659 113200             ;; 1.0E07
5660 230
5661 041100             ;; 1.0E06
5662 17
5663 103240             ;; 1.0E05
5664 1
5665 23420              ;; 1.0E04
5666 0
5667 1750               ;; 1.0E03
5668 0
5669 144                 ;; 1.0E02
    
```

5670	024752	000000
5671	024754	000012
5672	024756	000000
5673	024760	000001
5674	024762	000000
5675	024764	000014

0  
12  
0  
1  
0  
\$DECVL: .BLKB 12.

:::1.0E01  
:::1.0E00  
:::RESERVE STORAGE FOR ASCII STRING

```

5676 .SBTTL SUPRS - TYPE NUMERICAL ASCIZ STRING, REPLACE LEADING O'S BY BLANKS
5677 .SBTTL SUPRSL - TYPE NUMERICAL ASCIZ STRING, LEFT JUSTIFY
5678 ;NOT FROM SYSMAC
5679
5680 SUPRSL: MOV RO, -(SP) ;SAVE RO
5681 CLR SUP2
5682 MOV 4(SP), RO
5683 BR SUP1
5684
5685 SUPRS: MOV RO, -(SP) ;SAVE RO
5686 MOV 4(SP), RO ;PICKUP THE POINTER
5687 MOV RO, SUP2 ;SAVE FOR TYPING
5688
5689 SUP1:
5690 1$: TSTB (RO) ;TERMINATOR?
5691 BEQ 2$ ;BR IF YES
5692 CMPB #'0, (RO) ;IS THIS AN ASCII "0"?
5693 BNE 4$ ;NO
5694 MOVB #'40, (RO)+ ;REPLACE IT WITH "BLANK"
5695 BR 1$
5696 2$: DEC RO ;BACKUP BY 1
5697 MOVB #'0, (RO) ;ASCII "0"
5698 TST SUP2 ;LEFT JUSTIFY?
5699 BNE 5$ ;NO
5700 MOV RO, SUP2 ;YES
5701 5$: TYPE ;GO TYPE
5702 SUP2: .WORD 0
5703 MOV (SP)+, RO ;RESTORE RO
5704 MOV (SP)+, (SP) ;RESTORE THE STACK
RTS PC ;RETURN

```

```

5705 .SBTTL INTEGER MULTIPLY ROUTINE
5706
5707 ::*****
5708 :CALL
5709 :*      MOV      MULTIPLIER,-(SP)
5710 :*      MOV      MULTIPLICAND,-(SP)
5711 :*      JSR      PC,@$MULT
5712 :*      RETURN   ;;PRODUCT IS ON THE STACK
5713
5714 :*      STACK   PRODUCT
5715 :*      -----
5716 :*      TOP     LSB'S
5717 :*      +2     MSB'S
5718
5719 $MULT:
5720 025103 MOV      R0,-(SP)      ;;PUSH R0 ON STACK
5721 025100 MOV      R1,-(SP)      ;;PUSH R1 ON STACK
5722 025102 MOV      R2,-(SP)      ;;PUSH R2 ON STACK
5723 025104 CLR      -(SP)      ;;CLEAR THE SIGN KEY
5724 025106 MOV      12(SP),R1    ;;GET THE MULTIPLICAND
5725 025108 BPL      1$        ;;BR IF PLUS
5726 025110 INC      (SP)      ;;SET THE SIGN KEY
5727 025112 NEG      R1        ;;MAKE THE MULTIPLICAND POSITIVE
5728 025114 MOV      14(SP),R2    ;;GET THE MULTIPLIER
5729 025116 BPL      2$        ;;BR IF PLUS
5730 025118 DEC      (SP)      ;;UPDATE THE SIGN KEY
5731 025120 NEG      R2        ;;MAKE THE MULTIPLIER POSITIVE
5732 025122 MOV      #17,-(SP)    ;;SET THE LOOP COUNT
5733 025124 CLR      R0        ;;SETUP FOR THE MULTIPLY LOOP
5734 025126 BCC      4$        ;;DON'T ADD IF MULTIPLICAND = 0
5735 025128 ADD      R2,R0
5736 025130 ROR      R0        ;;POSITION THE PARTIAL PRODUCT AND
5737 025132 ROR      R1        ;;THE MULTIPLICAND
5738 025134 DEC      (SP)      ;;HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
5739 025136 BNE      3$        ;;BR IF NO
5740 025138 CMP      (SP)+,(SP)  ;;SHOULD PRODUCT BE NEGATIVE?
5741 025140 BEQ      5$        ;;GO TO EXIT IF NO
5742 025142 NEG      R0        ;;YES--SO MAKE IT SO
5743 025144 NEG      R1
5744 025146 SBC      R0
5745 025148 TST      (SP)+      ;;CLEAR SIGN INFO. OFF OF STACK
5746 025150 MOV      R0,12(SP)    ;;PUT THE PRODUCT ON THE STACK (MSB'S)
5747 025152 MOV      R1,10(SP)    ;;LSB'S
5748 025154 MOV      (SP)+,R2    ;;POP STACK INTO R2
5749 025156 MOV      (SP)+,R1    ;;POP STACK INTO R1
5750 025158 MOV      (SP)+,R0    ;;POP STACK INTO R0
5751 025160 RTS      PC
  
```

.SBTTL INTEGER DIVIDE ROUTINE

```

5752
5753
5754
5755
5756
5757
5758
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5770
5771 025212 013746 000034
5772 025216 012737 025226 000034
5773 025224 104400
5774 025226 012716 025250
5775 025232 016637 000004 000034
5776 025240 016666 000002 000004
5777 025246 000002
5778
5779 025250 042716 000017
5780 025254 010046
5781 025256 010146
5782 025260 010246
5783 025262 010346
5784 025264 005046
5785 025266 012746 000021
5786 025272 016601 000024
5787 025276 016600 000022
5788 025302 100005
5789 025304 105366 000003
5790 025310 005400
5791 025312 005401
5792 025314 005600
5793 025316 016602 000020
5794 025322 022702 000001
5795 025326 001463
5796 025330 005702
5797 025332 002407
5798 025334 003011
5799 025336 052766 000003 000014
5800 025344 012700 177777
5801 025350 000424
5802 025352 005266 000002
5803 025356 000401
5804 025360 005402
5805 025362 000241
5806 025364 000405
5807 025366 006100

```

```

:CALL:
*   MOV    LOW DIVIDEND,-(SP)      ;THE HIGH DIVIDEND MUST BE 1/2
*   MOV    HIGH DIVIDEND,-(SP)    ;      AS LARGE AS THE DIVISOR
*   MOV    DIVISOR,-(SP)
*   JSR    PC,$DIV
*   RETURN
*   "V"=0  IMPLIES NO ERROR
*   "V"=1  IMPLIES ERROR OCCURRED
*   "C"=0  DIVIDE OVERFLOW OCCURRED
*   "C"=1  ATTEMPTED TO DIVIDE BY ZERO

*   STACK NO ERROR OVERFLOW DIVIDE BY ZERO
*   -----
*   TOP    REMAINDER ALL ZEROS ALL ONES
*   +2     QUOTIENT  ALL ZEROS  ALL ONES

$DIV:  MOV    34,-(SP)      ;SAVE CURRENT TRAP VECTOR
        MOV    #1$,34    ;SET UP TRAP VECTOR
        TRAP
1$:    MOV    #2$(SP)    ;REPLACE NEW PC
        MOV    4(SP),34  ;RESTORE OLD TRAP VECT
        MOV    2(SP),4(SP) ;SAVE PSW
        RTI           ;RESTORE PSW

2$:    BIC    #17,(SP)   ;STRIP AWAY CONDITION CODES
        MOV    R0,-(SP)  ;PUSH R0 ON STACK
        MOV    R1,-(SP)  ;PUSH R1 ON STACK
        MOV    R2,-(SP)  ;PUSH R2 ON STACK
        MOV    R3,-(SP)  ;PUSH R3 ON STACK
        CLR    -(SP)     ;SAVE A PLACE FOR SIGNS
        MOV    #17,-(SP) ;SETUP THE ITERATION COUNTER
        MOV    24(SP),R1 ;PICKUP THE DIVIDEND
        MOV    22(SP),R0
        BPL   3$        ;CHECK THE SIGN
        DECB  3(SP)     ;KEEP TRACK OF THE SIGN
        NEG   R0        ;AND NEGATE THE ORIGINAL
        NEG   R1        ;NUMBER
        SBC   R0
        MOV   20(SP),R2 ;PICKUP THE DIVISOR
        CMP   #1,R2    ;IF THE DIVISOR IS 1 SKIP THE REST
        BEQ   13$     ;YES
        TST  R2
        BLT  4$        ;CHECK THE SIGN
        BGT  5$        ;DIVISOR OF 0 IS A NO-NO
        BIS  #3,14(SP) ;SET "V" & "C"
        MOV  #-1,R0    ;SET REMAINDER TO ALL ONES
        BR   9$        ;EXIT
        INC  2(SP)    ;KEEP TRACK OF DIVISORS SIGN
        BR   6$
5$:    NEG   R2        ;NEGATE THE ORIGINAL NUMBER
6$:    CLC
        BR   8$        ;START FORMING QUOTIENT
7$:    RCL  R0        ;POSITION MSB'S

```



5808	025370	010003			MOV	R0,R3	:COPY
5809	025372	060203			ADD	R2,R3	:COMPARE DIVIDEND & DIVISOR
5810	025374	103001			BCC	8\$	:BR IF DIVIDEND > DIVISOR
5811	025376	010300			MOV	R3,R0	:REMAINDER AFTER THIS LOOP
5812	025400	006101		8\$:	ROL	R1	:QUOTIENT BIT ENTERS HERE
5813	025402	005316			DEC	(SP)	:DONE?
5814	025404	001370			BNE	7\$	:BR IF NO
5815	025406	005701			TST	R1	:OVERFLOW?
5816	025410	100005			BPL	10\$	:BR IF NO
5817	025412	052766	000002	000014	BIS	#2,14(SP)	:SET "V" IN RETURN STATUS WORD
5818	025420	005000			CLR	R0	:SET REMAINDER TO ALL ZEROS
5819	025422	010001		9\$:	MOV	R0,R1	:COPY REMAINDER INTO QUOTIENT
5820	025424	005726		10\$:	TST	(SP)+	:CLEAR COUNTER FROM STACK
5821	025426	005716			TST	(SP)	:REMAINDER SIGN CORRECTION NEEDED?
5822	025430	002004			BGE	11\$	:BR IF NO
5823	025432	005400			NEG	R0	:NEGATE REMAINDER
5824	025434	105066	000001		CLRB	1(SP)	:CLEAR SIGN
5825	025440	005316			DEC	(SP)	:BUT DON'T FORGET QUOTIENT
5826	025442	005726		11\$:	TST	(SP)+	:QUOTIENT SIGN CORRECTION NEEDED?
5827	025444	001401			BEQ	12\$	:BR IF NO
5828	025446	005401			NEG	R1	:NEGATE QUOTIENT
5829	025450	010166	000020		MOV	R1,20(SP)	:RETURN QUOTIENT AND
5830	025454	010066	000016		MOV	R0,16(SP)	:REMAINDER TO USER
5831	025460	012603			MOV	(SP)+,R3	:POP STACK INTO R3
5832	025462	012602			MOV	(SP)+,R2	:POP STACK INTO R2
5833	025464	012601			MOV	(SP)+,R1	:POP STACK INTO R1
5834	025466	012600			MOV	(SP)+,R0	:POP STACK INTO R0
5835	025470	012666	000002		MOV	(SP)+,2(SP)	:SETUP TO RETURN CONDITION CODES
5836	025474	000002			RTI		:RETURN
5837	025476	022626		13\$:	CMP	(SP)+,(SP)+	:POP THE STACK
5838	025500	0007E3			BR	12\$	

5839  
 5840  
 5841  
 5842  
 5843  
 5844  
 5845  
 5846  
 5847  
 5848  
 5849  
 5850  
 5851  
 5852  
 5853  
 5854  
 5855  
 5856  
 5857  
 5858  
 5859  
 5860  
 5861  
 5862  
 5863  
 5864  
 5865  
 5866  
 5867  
 5868  
 5869  
 5870  
 5871  
 5872  
 5873  
 5874  
 5875  
 5876  
 5877  
 5878  
 5879  
 5880  
 5881  
 5882  
 5883

.SBTTL SAVE AND RESTORE RO-R5 ROUTINES

::\*\*\*\*\*

```

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

\$\$SAVREG:

```

MOV R0,-(SP)      ::PUSH R0 ON STACK
MOV R1,-(SP)      ::PUSH R1 ON STACK
MOV R2,-(SP)      ::PUSH R2 ON STACK
MOV R3,-(SP)      ::PUSH R3 ON STACK
MOV R4,-(SP)      ::PUSH R4 ON STACK
MOV R5,-(SP)      ::PUSH R5 ON STACK
MOV 22(SP),-(SP)  ::SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP)  ::SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP)  ::SAVE PS OF CALL
MOV 22(SP),-(SP)  ::SAVE PC OF CALL
RTI
    
```

::\*RESTORE RO-R5

::\*CALL:

::\* RESREG

\$\$RESREG:

```

MOV (SP)+,22(SP)  ::RESTORE PC OF CALL
MOV (SP)+,22(SP)  ::RESTORE PS OF CALL
MOV (SP)+,22(SP)  ::RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP)  ::RESTORE PS OF MAIN FLOW
MOV (SP)+,R5      ::POP STACK INTO R5
MOV (SP)+,R4      ::POP STACK INTO R4
MOV (SP)+,R3      ::POP STACK INTO R3
MOV (SP)+,R2      ::POP STACK INTO R2
MOV (SP)+,R1      ::POP STACK INTO R1
MOV (SP)+,R0      ::POP STACK INTO R0
RTI
    
```

```

025502
025502 010046
025504 010146
025506 010246
025510 010346
025512 010446
025514 010546
025516 016646 000022
025522 016646 000022
025526 016646 000022
025532 016646 000022
025536 000002
025540
025544 012666 000022
025544 012666 000022
025550 012666 000022
025554 012666 000022
025560 012605
025562 012604
025564 012603
025566 012602
025570 012601
025572 012600
025574 000002
    
```

```

5884
5885
5886
5887
5888
5889
5890 025576
5891 025576 010046
5892 025600 010146
5893 025602 010246
5894 025604 010346
5895 025606 010446
5896 025610 017604 000012
5897
5898 025614 011400
5899 025616 016401 000002
5900 025622 012703 177771
5901 025626 005002
5902 025630 006300
5903 025632 006101
5904 025634 006102
5905 025636 005203
5906 025640 001373
5907 025642 061400
5908 025644 005501
5909 025646 066401 000002
5910 025652 005502
5911 025654 062700 001057
5912 025660 005501
5913 025662 005502
5914 025664 062701 047401
5915 025670 005502
5916 025672 062702 000006
5917 025676 060200
5918 025700 005501
5919 025702 010014
5920 025704 010164 000002
5921 025710 012604
5922 025712 012603
5923 025714 012602
5924 025716 012601
5925 025720 012600
5926 025722 062716 000002
5927 025726 000207
5928 025730 123456
5929 025732 176543
5930 025734 001201
5931 025736 062465
5932 025740 176105
5933 025742 174532
5934 025744 157650
5935 025746 030757
5936 025750 131547
5937 025752 032070
5938 025754 123456
5939 025756 176543

```

```

.SBTTL RANDOM NUMBER GENERATOR ROUTINE
:CALL:
:JSR PC,$RAND ;CALL THE ROUTINE
:RETURN ;RETURN HERE THE RANDOM
: ;NUMBER WILL BE IN
: ;$SHNUM,$LONUM
$RAND:
MOV R0,-(SP) ;PUSH R0 ON STACK
MOV R1,-(SP) ;PUSH R1 ON STACK
MOV R2,-(SP) ;PUSH R2 ON STACK
MOV R3,-(SP) ;PUSH R3 ON STACK
MOV R4,-(SP) ;PUSH R4 ON STACK
MOV @12(SP),R4 ;GET POINTER TO THE SAVED SEEDS
;FOR GENERATING THIS RANDOM NUMBER
MOV (R4),R0 ;GET LO NUMBER SEED
MOV 2(R4),R1 ;GET HIGH NUMBER SEED
MOV #-7,R3 ;SET SHIFT COUNT
CLR R2 ;ZERO R2
1$:ASL R0 ;SHIFT R0 LEFT AND
ROL R1 ;ROTATE CARRY INTO R1 AND
ROL R2 ;ROTATE CARRY INTO R2
INC R3 ;CHECK FOR DONE
BNE 1$ ;CONTINUE SHIFT LOOP
ADD (R4),R0 ;ADD NUMBER TO MAKE X 129
ADC R1 ;PROPOGATE CARRY
ADD 2(R4),R1 ;ADD NUMBER TO MAKE X 129
ADC R2 ;PROPOGATE CARRY
ADD #1057,R0 ;ADD LOW CONSTANT
ADC R1 ;PROPOGATE CARRY
ADC R2 ;PROPOGATE CARRY
ADD #47401,R1 ;ADD HIGH CONSTANT
ADC R2 ;PROPOGATE CARRY
ADD #6,R2 ;ADD HIGHEST CONSTART
ADD R2,R0 ;REPRIME R0 WITH HIGHEST DIGIT
ADC R1 ;PROPOGATE CARRY
MOV R0,(R4) ;SAVE R0-$LONUM (FOR USE NXT TIME)
MOV R1,2(R4) ;SAVE R1-$SHNUM (FOR USE NXT TIME)
MOV (SP)+,R4
MOV (SP)+,R3 ;POP STACK INTO R3
MOV (SP)+,R2 ;POP STACK INTO R2
MOV (SP)+,R1 ;POP STACK INTO R1
MOV (SP)+,R0 ;POP STACK INTO R0
ADD #2,(SP) ;ADJUST SP FOR CORRECT RETURN
RTS ;RETURN
PC
RSDRVL: 123456 ;RANDOM SEED FOR DRIVE SELECTION (LO)
RSDRVH: 176543 ;" (HI)
RSFUNL: 1201 ;RANDOM SEED FOR FUNCTION
RSFUNH: 62465 ;" (HI)
RSCYLL: 176105 ;RANDOM SEED FOR CYLINDER (LO)
RSCYLH: 174532 ;" (HI)
RSBAL: 157650 ;RANDOM SEED FOR BUS ADDRESS (LO)
RSBAH: 30753 ;" (HI)
RSWCL: 131547 ;RANDOM SEED FOR WORD COUNT (LO)
RSWCH: 32070 ;" (HI)
RSDTL: 123456 ;RANDOM SEED FOR DATA (LO)
RSDTH: 176543 ;" (HI)

```

5940  
5941  
5942  
5943  
5944  
5945  
5946  
5947  
5948  
5949  
5950  
5951  
5952  
5953  
5954  
5955  
5956  
5957  
5958  
5959  
5960  
5961  
5962  
5963  
5964  
5965  
5966  
5967  
5968  
5969  
5970  
5971  
5972  
5973  
5974  
5975  
5976  
5977  
5978  
5979  
5980  
5981  
5982  
5983  
5984  
5985  
5986  
5987  
5988  
5989  
5990  
5991  
5992  
5993  
5994  
5995

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBE OF DIGITS TO TYPE
*CALL:
*   MOV   NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPOS                ;; CALL FOR TYPEOUT
*   .BYTE N              ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE M              ;; M=1 OR 0
*                               ;; 1=TYPE LEADING ZEROS
*                               ;; 0=SUPPRESS LEADING ZEROS
*
*$STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV   NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPON                ;; CALL FOR TYPEOUT
*
*$STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV   NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPOC                ;; CALL FOR TYPEOUT
*
5965 025760 017646 000000 $TYPOS: MOV   2(SP),-(SP)      ;; PICKUP THE MODE
5966 025764 116637 000001 026203 MOVVB 1(SP), $OFILL    ;; LOAD ZERO FILL SWITCH
5967 025772 112637 026205 MOVVB (SP)+, $OMODE+1  ;; NUMBER OF DIGITS TO TYPE
5968 025776 062716 000002 ADD   #2, (SP)      ;; ADJUST RETURN ADDRESS
5969 026002 000406 BR     $TYPON
5970 026004 112737 000001 026203 $TYPOC: MOVVB #1, $OFILL    ;; SET THE ZERO FILL SWITCH
5971 026012 112737 000006 026205 MOVVB #6, $OMODE+1  ;; SET FOR SIX(6) DIGITS
5972 026020 112737 000005 026202 $TYPON: MOVVB #5, $OCNT  ;; SET THE ITERATION COUNT
5973 026026 010346 MOV   R3,-(SP)      ;; SAVE R3
5974 026030 010446 MOV   R4,-(SP)      ;; SAVE R4
5975 026032 010546 MOV   R5,-(SP)      ;; SAVE R5
5976 026034 113704 026205 MOVVB $OMODE+1, R4  ;; GET THE NUMBER OF DIGITS TO TYPE
5977 026040 005404 NEG   R4
5978 026042 062704 000006 ADD   #6, R4      ;; SUBTRACT IT FOR MAX. ALLOWED
5979 026046 110437 026204 MOVVB R4, $OMODE  ;; SAVE IT FOR USE
5980 026052 113704 026203 MOVVB $OFILL, R4  ;; GET THE ZERO FILL SWITCH
5981 026056 016605 000012 MOV   12(SP), R5  ;; PICKUP THE INPUT NUMBER
5982 026062 005003 CLR   R3          ;; CLEAR THE OUTPUT WORD
5983 026064 006105 1$: ROL  R5          ;; ROTATE MSB INTO "C"
5984 026066 000404 BR     3$
5985 026070 006105 2$: ROL  R5          ;; GO DO MSB
5986 026072 006105 ROL  R5          ;; FORM THIS DIGIT
5987 026074 006105 ROL  R5
5988 026076 010503 MCV  R5, R3
5989 026100 006103 3$: ROL  R3          ;; GET LSB OF THIS DIGIT
5990 026102 105337 026204 DECB  $OMODE      ;; TYPE THIS DIGIT?
5991 026106 100016 BPL  7$          ;; BR IF NO
5992 026110 042703 177770 BIC  #177770, R3 ;; GET RID OF JUNK
5993 026114 001002 BNE  4$          ;; TEST FOR 0
5994 026116 005704 TST  R4          ;; SUPPRESS THIS 0?
5995 026120 001403 BEQ  5$          ;; BR IF YES
    
```

5996	026122	005204		4\$:	INC	R4		:: DON'T SUPPRESS ANYMORE 0'S
5997	026124	052703	000060		BIS	#'0,R3		:: MAKE THIS DIGIT ASCII
5998	026130	052703	000040	5\$:	BIS	#' R3		:: MAKE ASCII IF NOT ALREADY
5999	026134	110337	026200		MOVB	R3 9\$		:: SAVE FOR TYPING
6000	026140	104401	026200		TYPE	8\$		:: GO TYPE THIS DIGIT
6001	026144	105337	026202	7\$:	DECB	\$OCNT		:: COUNT BY 1
6002	026150	003347			BGT	2\$		:: BR IF MORE TO DO
6003	026152	002402			BLT	6\$		:: BR IF DONE
6004	026154	005204			INC	R4		:: INSURE LAST DIGIT ISN'T A BLANK
6005	026156	000744			BR	2\$		:: GO DO THE LAST DIGIT
6006	026160	012605		6\$:	MOV	(SP)+,R5		:: RESTORE R5
6007	026162	012604			MOV	(SP)+,R4		:: RESTORE R4
6008	026164	012603			MOV	(SP)+,R3		:: RESTORE R3
6009	026166	016666	000002 000004		MOV	2(SP),4(SP)		:: SET THE STACK FOR RETURNING
6010	026174	012616			MOV	(SP)+,(SP)		
6011	026176	000002			RTI			:: RETURN
6012	026200	000		8\$:	.BYTE	0		:: STORAGE FOR ASCII DIGIT
6013	026201	000			.BYTE	0		:: TERMINATOR FOR TYPE ROUTINE
6014	026202	000		\$OCNT:	.BYTE	0		:: OCTAL DIGIT COUNTER
6015	026203	000		\$OFILL:	.BYTE	0		:: ZERO FILL SWITCH
6016	026204	000000		\$OMODE:	.WORD	0		:: NUMBER OF DIGITS TO TYPE
5017								

```

6018 .SBTTL ERROR HANDLER ROUTINE
6019
6020 ;*SW15=1 HALT ON ERROR
6021 ;*SW13=1 INHIBIT ERROR TYPEOUTS
6022 ;*SW12=1 TYPE OUT THE ERROR HISTORY, THE FUNCTION THAT
6023 ; WAS BEING PERFORMED ON RK AT THE TIME OF ERROR AND
6024 ; THE FUNCTION PERFORMED PRIOR TO THAT.
6025 ;*NOTE THIS SWITCH OPTION (12) IS MEANINGFUL ONLY FOR ERRORS OCCURING IN THE
6026 ;*EXERCISER PART OF THE PROGRAM.
6027 ;*SW11=1 DUMP OUT ALL RK REGISTERS
6028 ;*SW10=1 BELL ON ERROR
6029 ;*SW09=1 LOOP ON ERROR
6030 ;*SW03=1 TYPE OUT TIME AT WHICH ERROR OCCURED
6031 ;*SW02=1 DROP THE DRIVE AFTER MAXM ERORS ON THIS DRIVE
6032 ;*SW01=1 TYPE OUT THE SERIAL NUMBER OF THE ERRORING DRIVE
6033 ;*GO TO $ERRTYP ON ERROR
6034

```

```

6035
6036 026206 104407 $ERROR: CKSWR ;LOOK FOR A 'CONTROL G'
6037 026210 105237 001103 INCB $ERFLG ;SET THE ERROR FLAG
6038 026214 001774 BEQ $ERROR ;DON'T LET THE FLAG GO TO ZERO
6039 026216 013777 001102 152716 MOV $STNM,$DISP ;DISPLAY TEST NUMBER AND ERROR FLAG
6040 026224 032777 002000 152706 BIT #SW10,$SWR ;BELL ON ERROR?
6041 026232 001402 BEQ 1$ ;NO - SKIP
6042 026234 104401 001206 TYPE $BELL ;RING BELL
6043 026240 005237 001112 1$: INC $ERTTL ;COUNT THE NUMBER OF ERRORS
6044
6045 026244 032777 000004 152666 BIT #SW2,$SWR ;COUNT # OF ERORS & DROP DRIVE?
6046 026252 001415 BEQ 3$ ;NO
6047 ;YES
6048 026254 010146 MOV R1,-(SP) ;SAVE R1
6049 026256 013701 001250 MOV SRDRV,R1 ;GET ERRORING DRIVE #
6050 026262 100410 BMI 2$ ;IF (SRDRV)=-1, SKIP(BECAUSE THE
6051 ;ERROR WAS NOT ATTRIBUTABLE TO ANY
6052 ;SPECIFIC DRIVE)
6053 026264 105261 001542 INCB ERDRV(R1) ;COUNT # OF ERORS ON THIS DRIVE
6054 026270 126127 001542 000003 CMPB ERDRV(R1),#3 ;# OF ERORS GREATER THAN ALLOWABLE?
6055 026276 101402 BLOS 2$ ;NO
6056 026300 000137 016312 JMP DSELCT ;DROP THE DRIVE
6057
6058 026304 012601 2$: MOV (SP)+,R1 ;RESTORE R1
6059
6060 026306 011637 001116 3$: MOV (SP),$ERRPC ;GET ADDRESS OF ERROR INSTRUCTION
6061 026312 162737 000002 001116 SUB #2,$ERRPC
6062 026320 005046 CLR -(SP)
6063 026322 117716 152570 MOVB $ERRPC,(SP) ;STRIP AND SAVE THE ERROR ITEM CODE
6064 026326 121627 000100 CMPB (SP),#100 ;FORM THE COMSECT ITEM# IF THIS IS AN
6065 026332 002402 BLT 4$ ;EROR MESSAGE EQUAL OR ABOVE 100.
6066 026334 162716 000040 SUB #40,(SP) ;NOTE THERE R 2 CLASSES OF ERRORS:
6067 ;1) $ITEMB'S BELOW 100
6068 ;2) $ITEMB'S ABOVE 100
6069 ;SUBTRACTION FACTOR HAS TOBE SUCH THAT
6070 ;THE COMSECT OFFSET IS SELECTED. THIS
6071 ;FACTOR WILL CHANGE IF THE TOTAL # OF
6072 ;ERROR MESSAGES IN CLASS 1 CHANGES.
6073 ;#=100 -LAST ITEM IN # CLASS 1 - 1

```

```

6074 026340 012637 001114 4$: MOV (SP)+,$ITEMB
6075 026344 032777 020000 152566 BIT #SW13,$SWR ;SKIP TYPEOUT IF SET
6076 026352 001012 BNE 5$ ;SKIP TYPEOUTS
6077 026354 004737 026472 JSR PC,$SERRTYP ;GO TO USER ERROR ROUTINE
6078 026360 104401 001213 TYPE ,SCLF
6079
6080 026364 032777 010000 152546 BIT #SW12,$SWR ;TYPE ERROR HISTORY?
6081 026372 001402 BEQ 5$ ;NO
6082 026374 004737 020426 JSR PC,HISTRY ;YES
6083
6084 026400 023737 000042 000046 5$: CMP @#42,@#46 ;ACT11 AUTOMATIC MODE?
6085 026406 001403 BEQ .+10 ;YES HALT ON ERROR
6086 026410 005777 152524 TST @SWR ;HALT ON ERROR
6087 026414 100002 BPL 6$ ;SKIP IF CONTINUE
6088 026416 000000 HALT ;HALT ON ERROR!
6089 026420 104407 CKSWR ;LOOK FOR A 'CONTROL G'
6090 026422 032777 001000 152510 6$: BIT #SW09,$SWR ;LOOP ON ERROR SWITCH SET?
6091 026430 001411 BEQ 7$ ;BR IF NO
6092 026432 123727 001114 000040 CMPB $ITEMB,#40 ;THERE R 37 ERROR MESSAGES IN CLASS 1
6093 026440 103011 BHIS B$
6094 026442 013746 001244 MOV PPRLVL,-(SP) ;LOCK OUT ALL INTERRUPTS ON RETURN
6095 ;FROM THIS EROR HANDLER, IF THE EROR
6096 ;IS IN EXERCISER & LOOPING IS TO
6097 ;BE DONE
6098 026446 005726 TST (SP)+
6099 026450 012716 015726 MOV #EXCLUP,(SP) ;IF THIS ERROR CALL WAS FROM EXERCISER
6100 ;PART OF THE PROGRAM, GO TO 'EXCLUP'
6101 ;OTHERWISE RETURN THRU '$LUPERR'
6102
6103 026454 012737 177777 001250 7$: MOV #-1,$RDRV ;RESET SERIAL NO FLAG
6104 ;IF ($RDRV)=-1, THEN THE SERIAL
6105 ;NO OF THE DRIVE WILL NOT BE
6106 ;TYPED OUT.
6107 ;OTHERWISE, SERIAL NO FOR THE DRIVE
6108 ;# IN '$RDRV' WILL BE TYPED.
6109 026462 000002 RTI
6110 026464 013716 001110 9$: MOV $LPERR,(SP) ;FUDGE RETURN FOR LOOPING
6111 026470 000771 BR 7$ ;RETURN

```

SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```
6112
6113
6114
6115
6116
6117
6118
6119 026472 104401 001213      SERRTYP:TYPE      , $CRLF      ;"CARRIAGE RETURN" & "LINE FEED"
6120 026476 010046      MOV      RO,-(SP) ;SAVE RO
6121 026500 005000      CLR      RO      ;PICKUP THE ITEM INDEX
6122 026502 153700 001114      BLSB     @($ITEMB,RO
6123 026506 001004      BNE      1$      ;IF ITEM NUMBER IS ZERO, JUST
6124 026510 013746 001116      MOV      $ERRPC,-(SP) ;TYPE THE PC OF THE ERROR
6125 026514 104402      TYPOC
6126 026516 000425      BR       5$      ;GET OUT
6127 026520 005300      1$: DEC     RO      ;ADJUST THE INDEX SO THAT IT WILL
6128 026522 006300      ASL     RO      ;WORK FOR THE ERROR TABLE
6129 026524 006300      ASL     RO
6130 026526 006300      ASL     RO
6131 026530 062700 002666      ADD     @SERRTB,RO ;FORM TABLE POINTER
6132 026534 012037 026544      MOV     (RO)+,2$  ;PICKUP "ERROR MESSAGE" POINTER
6133 026540 001404      BEQ     4$      ;SKIP TYPEOUT IF NO POINTER
6134 026542 104401      TYPE
6135 026544 000000      2$: .WORD 0      ;TYPE THE "ERROR MESSAGE"
6136 026546 104401 001213      3$: .CRLF      ;"ERROR MESSAGE" POINTER GOES HERE
6137 026552 032777 004000 152360 4$: .SWI1,@SWR ;"CARRIAGE RETURN" & "LINE FEED"
6138 026560 001404      BEQ     5$      ;DUMP OUT RK REGISTERS?
6139 026562 004737 027032      JSR     PC,DMPREG ;GO TYPE OUT RK REGISTERS
6140 026566 104401 001213      TYPE      , $CRLF
6141
6142 026572 012037 026602      5$: MOV     (RO)+,6$  ;PICKUP "DATA HEADER" POINTER
6143 026576 001404      BEQ     7$      ;SKIP TYPEOUT IF 0
6144 026600 104401      TYPE
6145 026602 000000      6$: .WORD 0      ;TYPE THE "DATA HEADER"
6146 026604 104401 001213      7$: .CRLF      ;"DATA HEADER" POINTER GOES HERE
6147 026610 011000      MOV     (RO),RO  ;"CARRIAGE RETURN" & "LINE FEED"
6148 026612 001406      BEQ     9$      ;PICKUP "DATA TABLE" POINTER
6149
6150
6151 026614 013046      8$: MOV     @($RO)+,-(SP) ;TYPE AN OCTAL NUMBER
6152 026616 104402      TYPOC      ;SAVE @($RO)+ FOR TYPEOUT
6153 026620 104401 002663      TYPE      ;GO TYPE--OCTAL ASCII(ALL DIGITS.
6154 026624 005710      TST     (RO)     ;TYPE TWO(2) SPACES
6155 026626 001372      BNE     8$      ;IS THERE ANOTHER NUMBER?
6156
6157 026630 012600      9$: MOV     (SP)+,RO  ;BR IF YES
6158 026632 104401 001213      TYPE      ;RESTORE RO
6159 026636 123727 001114 000040      CMPB    $ITEMB,#40 ;"CARRIAGE RETURN" & "LINE FEED"
6160
6161 026644 103004      BHIS    10$     ;SKIP TIME & SERIAL # FOR
6162 026646 004737 026660      JSR     PC,TIMYP ;NON-EXERCISER ERRORS
6163 026652 004737 026766      JSR     PC,SNOTYP
6164
6:65 026656 000207      10$: RTS    PC   ;TYPE OUT TIME IF SW 3 IS SET
                                ;GO TYPE OUT THE SERIAL # OF THE
                                ;ERRING DRIVE, IF SW 1 IS SET.
                                ;RE*JRN
```



```

6166          :TIMTYP
6167          :THIS ROUTINE TYPES THE TIME IN HOURS:MIN:SECS IF SW 3 IS SET
6168          :SW 3 SHOULD NOT BE SET IF KWILL IS NOT PRESENT.
6169
6170 026660 032777 000010 152252 TIMTYP: BIT      #SW2,@SWR      ;IS SW 3 SET?
6171 026666 001434          BEQ      4$              ;IF NOT SKIP TYPING TIME
6172 026670 104401 002652          TYPE     ,MSG29        ;'TIME'
6173 026674 104401 002662          TYPE     ,BLNK53
6174 026700 104414          SAVREG
6175 026702 012700 001552          MOV      #KWHR,RO    ;SAVE RO-R4
6176 026706 012001          MOV      (RO)+,R1    ;INITIALIZE POINTER
6177 026710 000404          BR       2$
6178 026712 012001          1$: MOV     (RO)+,R1    ;TYPE OUT
6179 026714 001402          BEQ      2$
6180 026716 062701 000074          ADD     #60,R1
6181 026722 010137 026762          2$: MOV     R1,5$
6182 026726 012746 026762          MOV     #5$,-(SP)   ;HOURS:MINS:SECS
6183 026732 004737 024604          JSR     PC,@#SDB2D  ;CONVERT TO ASCIZ STRING
6184 026736 004737 025000          JSR     PC,@#SUPRSL ;GO TYPE
6185 026742 020027 001560          CMP     RO,#KWSEC+2 ;ALL DONE?
6186 026746 001403          BEQ     3$
6187 026750 104401 002334          TYPE     ,MSG18
6188 026754 000756          BR      1$
6189 026756 104415          3$: RESREG   ;RESTORE RO-R4
6:90 026760 000207          4$: RTS     PC      ;RETURN
6:91 026762 000000 000000          5$: .WORD  0,0
    
```

```

6192 ;SNOTYP
6193 ;THIS ROUTINE TYPES OUT THE SERIAL NUMBER OF THE ERRORING DRIVE, IF SW 1
6194 ;IS SET. NOTE THAT THE SERIAL NUMBER IS TYPED OUT ONLY WHEN THE DRIVE
6195 ;CAN BE IDENTIFIED POSITIVELY, AS THE ONE WHICH GAVE THE ERROR. IF THE
6196 ;ERROR CANNOT BE ATTRIBUTED TO ANY SPECIFIC DRIVE (SRDRV)=-1 THEN
6197 ;THE SERIAL NUMBER IS NOT TYPED OUT.
6198
6199 026766 032777 000002 152144 SNOTYP: BIT #SW1,@SWR ;TYPE OUT SERIAL #?
6200 026774 001415 BEQ 2$ ;NO
6201 026776 010146 MOV R1,-(SP) ;SAVE R1
6202 027000 013701 001250 MOV SRDRV,R1 ;GET ERRORING DRIVE #
6203 027004 006301 ASL R1 ;IF (SRDRV)=-1, SKIP (BECAUSE
6204 027006 100407 BMI 1$ ;THE ERROR WAS NOT ATTRIBUTABLE
6205 ;TO A SPECIFIC DRIVE)
6206 027010 104401 001213 TYPE ,SCLF ;TYPE "SR. NO:"
6207 027014 104401 002326 TYPE MSG17 ;GET THE SERIAL #
6208 027020 016146 001266 MOV SRNO(R1),-(SP) ;TYPE IT OUT (DECIMAL)
6209 027024 104405 TYPDS
6210
6211 027026 012601 1$: MOV (SP)+,R1 ;RESTORE R1
6212 027030 000207 2$: RTS PC ;RETURN
6213
6214
6215
6216 ;DMPREG
6217 ;THIS ROUTINE DUMPS OUT ALL RK11 REGISTERS WHEN SW 11 IS SET AND AN ERROR OCCURS.
6218
6219 027032 DMPREG:
6220 027032 104401 027040 TYPE ,65$ ;;TYPE ASCIZ STRING
6221 027036 000441 BR 64$ ;;GET OVER THE ASCIZ
6222 ;;65$: .ASCIZ <15><12>/ PC PKDS RKER RKCS RKWC RKBA RKCA RKDB
6223 64$:
6224 027142 MOV $ERRPC,-(SP)
6225 027146 TYPOC
6226 027150 104401 002663 TYPE ,BLNKS2
6227 027154 010046 MOV RO,-(SP)
6228 027156 012700 001216 MOV #RKDS,RO
6229 027162 013046 1$: MOV @RO+,-(SP)
6230 027164 104402 TYPOC
6231 027166 104401 002663 TYPE ,BLNKS2
6232 027172 020027 001232 CMP RO,#RKDB
6233 027176 003771 BLE 1$
6234 027200 012600 MOV (SP)+,RO
6235 027202 000207 RTS PC
  
```

.SBTTL SCOPE HANDLER ROUTINE

6236  
6237  
6238  
6239  
6240  
6241  
6242  
6243  
6244  
6245  
6246  
6247  
6248  
6249  
6250  
6251  
6252  
6253  
6254  
6255  
6256  
6257  
6258  
6259  
6260  
6261  
6262  
6263  
6264  
6265  
6266  
6267  
6268  
6269  
6270  
6271  
6272  
6273  
6274  
6275  
6276  
6277  
6278

```

027204
027204 104407
027206 032777 040000 151724
027214 001047
027216 000416
027220 013746 000004
027224 012737 027244 000004
027232 005737 177060
027236 012637 000004
027242 000421
027244 022626
027246 012637 000004
027252 000407
027254 105737 001102
027260 001412
027262 032777 001000 151650
027270 001404
027272 013737 001110 001106
027300 000415
027302 105037 001103
027306 105237 001102
027312 011637 001106
027316 011637 001110
027322 005037 001204
027326 112737 000001 001115
027334 013777 001102 151600
027342 013716 001106
027346 000002
    
```

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY'7:0).
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY'15:08'
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1 LOOP ON TEST
*SW09=1 LOOP ON ERROR
*CALL
* SCOPE ;;SCOPE=IOT

$SCOPE:
CKSWR ;; TEST FOR CHANGE IN SOFT-SWR
1$: BIT #BIT14,$SWR ;; LOOP ON PRESENT TEST?
BNE $OVER ;; YES IF SW14=1
*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 6$
MOV @#ERRVEC, -(SP) ;; IF RUNNING ON THE "XOR" TESTER CHANGE
MOV #5,$#ERRVEC ;; THIS INSTRUCTION TO A "NOP" (NOP=24C,
TST @#177060 ;; SAVE THE CONTENTS OF THE ERROR VECTOR
MOV (SP)+,@#ERRVEC ;; SET FOR TIMEOUT
BR $SVLAD ;; TIME OUT ON XOR?
5$: CMP (SP)+,(SP)+ ;; RESTORE THE ERROR VECTOR
MOV (SP)+,@#ERRVEC ;; GO TO THE NEXT TEST
BR 7$ ;; CLEAR THE STACK AFTER A TIME OUT
6$: *****END OF CODE FOR THE XOR TESTER*****
2$: TSTB $ERFLG ;; HAS AN ERROR OCCURRED?
BEQ $SVLAD ;; BR IF NO
BIT #BIT09,$SWR ;; LOOP ON ERROR?
BEQ 4$ ;; BR IF NO
7$: MOV $LPERR,$LPADR ;; SET LOOP ADDRESS TO LAST SCOPE
BR $OVER
4$: CLRB $ERFLG ;; ZERO THE ERROR FLAG
$SVLAD: INCB $STNM ;; COUNT TEST NUMBERS
MOV (SP),$LPADR ;; SAVE SCOPE LOOP ADDRESS
MOV (SP),$LPERR ;; SAVE ERROR LOOP ADDRESS
CLR $ESCAPE ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
MOVB #1,$EMAX ;; ONLY ALLOW ONE(1) ERROR ON ME : TEST
$OVER: MOV $STNM,@DISPLAY ;; DISPLAY TEST NUMBER
MOV $LPADR,(SP) ;; FUDGE RETURN ADDRESS
RTI ;; FIXES PS
    
```

6279  
6280  
6281  
6282  
6283  
6284  
6285  
6286  
6287  
6288  
6289  
6290  
6291  
6292  
6293  
6294  
6295  
6296  
6297  
6298  
6299  
6300  
6301  
6302  
6303  
6304  
6305  
6306  
6307  
6308  
6309  
6310  
6311  
6312  
6313  
6314  
6315  
6316  
6317  
6318  
6319  
6320  
6321  
6322  
6323  
6324  
6325  
6326  
6327  
6328  
6329

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

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

```

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

.SBTTL TRAP TABLE

```

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

	ROUTINE		
\$TRPAD:	.WORD	\$TRAP2	
	\$TYPE	;;CALL=TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
	\$TYPOC	;;CALL=TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS.)
	\$TYPOS	;;CALL=TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
	\$TYPON	;;CALL=TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
	\$TYPDS	;;CALL=TYPDS	TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
	\$GTSWR	;;CALL=GTSWR	TRAP+6(104406) GET SOFT-SWR SETTING
	\$CKSWR	;;CALL=CKSWR	TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
	\$RDCHR	;;CALL=RDCHR	TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
	\$RDLIN	;;CALL=RDLIN	TRAP+11(104411) TTY TYPEIN STRING ROUTINE
	\$RDOCT	;;CALL=RDOCT	TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
	\$RDEEC	;;CALL=RDEEC	TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY
	\$SAVREG	;;CALL=SAVREG	TRAP+14(104414) SAVE RO-R5 ROUTINE
	\$RESREG	;;CALL=RESREG	TRAP+15(104415) RESTORE RO-R5 ROUTINE
	CN.RST	;;CALL=CN.RESET	TRAP+16(104416) CONTROL RESET ROUTINE
	CN.RDY	;;CALL=CN.RDY	TRAP+17(104417) WAIT FOR CONTROL READY
	DR.RST	;;CALL=DRV.RESET	TRAP+20(104420) DRIVE RESET ROUTINE
	TY.MSG	;;CALL=TYPMSG	TRAP+21(104421) TYPE MESSAGE ROUTINE. Sw13

GIE

.SBTTL POWER DOWN AND UP ROUTINES

```

6330
6331
6332
6333
6334 027450 012737 027614 000024 $PWRDN: MOV $SILLUP,@PWRVEC ;;SET FOR FAST UP
6335 027456 012737 000340 000026 MOV @340,@PWRVEC+2 ;;PRIO:7
6336 027464 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
6337 027466 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
6338 027470 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
6339 027472 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
6340 027474 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
6341 027476 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
6342 027500 017746 151434 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
6343 027504 010637 027620 MOV SP,$SAVR6 ;;SAVE SP
6344 027510 012737 027522 000024 MOV $PWRUP,@PWRVEC ;;SET UP VECTOR
6345 027516 000000 HALT
6346 027520 000776 BR .-2 ;;HANG UP
6347
6348
6349
6350 027522 012737 027614 000024 $PWRUP: MOV $SILLUP,@PWRVEC ;;SET FOR FAST DOWN
6351 027530 013706 027620 MOV $SAVR6,SP ;;GET SP
6352 027534 005037 027620 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
6353 027540 005237 027620 IS: INC $SAVR6 ;;WAIT FOR THE INC
6354 027544 001375 BNE IS ;;OF WORD
6355 027546 012677 151366 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
6356 027552 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
6357 027554 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
6358 027556 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
6359 027560 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
6360 027562 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
6361 027564 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
6362 027566 012737 027450 000024 MOV $PWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
6363 027574 012737 000340 000026 MOV @340,@PWRVEC+2 ;;PRIO:7
6364 027602 104401 TYPE ;;REPORT THE POWER FAILURE
6365 027604 027622 $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
6366 027606 012716 MOV (PC)+,(SP) ;;RESTART AT PFSTR
6367 027610 003366 $PWRAD: .WORD PFSTR ;;RESTART ADDRESS
6368 027612 000002 RTI
6369 027614 000000 $SILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
6370 027616 000776 BR .-2 ;;BEFORE THE POWER DOWN WAS COMPLETE
6371 027620 000000 $SAVR6: 0 ;;PUT THE SP HERE
6372 027622 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
6373 027630 000122 .EVEN
6374
    
```

```

        6375                                     :ERROR MESSAGES
        6376
        6377 027632 051105 051117 047440 EM1: .ASCIZ /EROR ON WRITE/
        6378 027640 020116 051127 052111
        6379 027646 000105
        6380 027650 052101 046505 052120 EM2: .ASCIZ /ATEMPT TO INITIATE FUNCTION ON 'BUSY' DRIVE/
        6381 027656 052040 020117 047111
        6382 027664 052111 040511 042524
        6383 027672 043040 047125 052103
        6384 027700 047511 020116 047117
        6385 027706 023440 052502 054523
        6386 027714 020047 051104 042526
        6387 027722 000
        6388 027723 103 052116 047522 EM3: .ASCIZ /CNTROL RDY NOT SET/
        6389 027730 020114 042122 020131
        6390 027736 047516 020124 042523
        6391 027744 000124
        6392 027746 051057 053457 051457 EM4: .ASCIZ "/R/W/S RDY NOT SET"
        6393 027754 051040 054504 047040
        6394 027762 052117 051440 052105
        6395 027770 000
        6396 027771 103 052116 047522 EM5: .ASCIZ /CNTROL RDY NOT SET AFTER 1ST INTRUPT ON ISSUING SEEK/
        6397 027776 020114 042122 020131
        6398 030004 047516 020124 042523
        6399 030012 020124 043101 042524
        6400 030020 020122 051461 020124
        6401 030026 047111 051124 050125
        6402 030034 020124 047117 044440
        6403 030042 051523 044525 043516
        6404 030050 051440 042505 000113
        6405 030056 051127 047117 020107 EM6: .ASCIZ /WRONG BITS IN RKCS. EXPCT SEEK/
        6406 030064 044502 051524 044440
        6407 030072 020116 045522 051503
        6408 030100 020054 054105 041520
        6409 030106 020124 042523 045505
        6410 030114 000
        6411 030115 047 052502 054523 EM7: .ASCIZ /'BUSY' FLAG CLEAR ON INTRUPTING DRIVE/
        6412 030122 020047 046106 043501
        6413 030130 041440 042514 051101
        6414 030136 047440 020116 047111
        6415 030144 051124 050125 044524
        6416 030152 043516 042040 053122
        6417 030160 000105
        6418 030162 050047 051517 052111 EM10: .ASCIZ /'POSITIONING' FLAG FOR INTRUPTING DRIVE CLEAR/
        6419 030170 047511 044516 043516
        6420 030176 020047 046106 043501
        6421 030204 043040 051117 044440
        6422 030212 052116 052522 052120
        6423 030220 047111 020107 051104
        6424 030226 042526 041440 042514
        6425 030234 051101 000
        6426 030237 047 051105 023522 EM11: .ASCIZ /'ERR'OR SET AFTER 1ST INTERRUPT ON ISSUING SEEK
        6427 030244 051117 051440 052105
        6428 030252 040440 052106 051105
        6429 030260 030440 052123 044440
        6430 030266 052116 051105 052522
    
```

6431	030274	052120	047440	020116		
6432	030302	051511	052523	047111		
6433	030310	020107	042523	045505		
6434	030316	000				
6435	030317	123	050103	051440	EM12:	.ASCIZ /SCP SET AFTER 1ST INTRUPT ON ISSUING SEEK/
6436	030324	052105	040440	052106		
6437	030332	051105	030440	052123		
6438	030340	044440	052116	052522		
6439	030346	052120	047440	020116		
6440	030354	051511	052523	047111		
6441	030362	020107	042523	045505		
6442	030370	000				
6443	030371	103	052116	047522	EM13:	.ASCIZ /CNTROL RDY NOT SET AFTER SEEK DONE INTRUPT/
6444	030376	020114	042122	020131		
6445	030404	047516	020124	042523		
6446	030412	020124	043101	042524		
6447	030420	020122	042523	045505		
6448	030426	042040	047117	020105		
6449	030434	047111	051124	050125		
6450	030442	000124				
6451	030444	047111	051124	050125	EM14:	.ASCIZ /INTRUPTING DRVE (SEEK DONE) WAS NOT 'BUSY'/
6452	030452	044524	043516	042040		
6453	030460	053122	020105	051450		
6454	030466	042505	020113	047504		
6455	030474	042516	020051	040527		
6456	030502	020123	047516	020124		
6457	030510	041047	051525	023531		
6458	030516	000				
6459	030517	122	053457	051457	EM15:	.ASCIZ "R/W/S READY NOT SET FOR INTRUPTING DRVE (SEEK DONE "
6460	030524	051040	040505	054504		
6461	030532	047040	052117	051440		
6462	030540	052105	043040	051117		
6463	030546	044440	052116	052522		
6464	030554	052120	047111	020107		
6465	030562	051104	042526	024040		
6466	030570	042523	045505	042040		
6467	030576	047117	024505	000		
6468	030603	123	047111	042440	EM16:	.ASCIZ /SIN EROR/
6469	030610	047522	000122			
6470	030614	042447	051122	047447	EM17:	.ASCIZ /'ERR'OR ON DOING SEEK/
6471	030622	020122	047117	042040		
6472	030630	044517	043516	051440		
6473	030636	042505	000113			
6474	030642	041523	020120	044504	EM20:	.ASCIZ /SCP DID NOT SET AFTER SEEK WAS DONE/
6475	030650	020104	047516	020124		
6476	030656	042523	020124	043101		
6477	030664	042524	020122	042523		
6478	030672	045505	053440	051501		
6479	030700	042040	047117	000105		
6480	030706	047523	052106	042440	EM21:	.ASCIZ /SOFT EROR/
6481	030714	047522	000122			
6482	030720	040504	040524	024040	EM23:	.ASCIZ /DATA (COMPARISON) EROR/
6483	030726	047503	050115	051101		
6484	030734	051511	047117	020051		
6485	030742	051105	051117	000		
6486	030747	103	052116	047522	EM24:	.ASCIZ /CNTROL RDY CLR ON INTRUPT AFTER RK FUNCTION

6487	030754	020114	042122	020131		
6488	030762	046103	020122	047117		
6489	030770	044440	052116	052522		
6490	030776	052120	040440	052106		
6491	031004	051105	051040	020113		
6492	031012	052506	041516	044524		
6493	031020	047117	000			
6494	031023	123	052524	045503	EM26:	.ASCIZ /STUCK IN LOOP,8 COMANDS SHLD BE DONE BY NOW/
6495	031030	044440	020116	047514		
6496	031036	050117	034054	041440		
6497	031044	046517	047101	051504		
6498	031052	051440	046110	041104		
6499	031060	020105	047504	042516		
6500	031066	041040	020131	047516		
6501	031074	000127				
6502	031076	052101	050115	020124	EM27:	.ASCIZ /ATMPT TO DO WRITE BEFORE WRT CHK/
6503	031104	047524	042040	020117		
6504	031112	051127	052111	020105		
6505	031120	042502	047506	042522		
6506	031126	053440	052122	041440		
6507	031134	045510	000			
6508	031137	101	046524	052120	EM30:	.ASCIZ /ATMPT TO REEXECUTE COMMAND-IN PROGRESS OR ALREADY FINISHED/
6509	031144	052040	020117	042522		
6510	031152	054105	041505	052125		
6511	031160	020105	047503	046515		
6512	031166	047101	026504	047111		
6513	031174	050040	047522	051107		
6514	031202	051505	020123	051117		
6515	031210	040440	051114	040505		
6516	031216	054504	043040	047111		
6517	031224	051511	042510	000104		
6518	031232	043047	047125	052103	EM31:	.ASCIZ /'FUNCTION IN PROGRES' FLG FOR INTRUPTING DRIVE ISN'T SET/
6519	031240	047511	020116	047111		
6520	031246	050040	047522	051107		
6521	031254	051505	020047	046106		
6522	031262	020107	047506	020122		
6523	031270	047111	051124	050125		
6524	031276	044524	043516	042040		
6525	031304	044522	042526	044440		
6526	031312	047123	052047	051440		
6527	031320	052105	000			
6528	031323	125	042516	050130	EM32:	.ASCIZ /UNEXPCTED DRIVE INTRUPTED/
6529	031330	052103	042105	042040		
6530	031336	044522	042526	044440		
6531	031344	052116	052522	052120		
6532	031352	042105	000			
6533	031355	125	042516	050130	EM33:	.ASCIZ /UNEXPCTD FUNCTION CODE IN RKCS AFTER INTRUPT/
6534	031362	052103	020104	052506		
6535	031370	041516	044524	047117		
6536	031376	041440	042117	020105		
6537	031404	047111	051040	041513		
6538	031412	020123	043101	042524		
6539	031420	020122	047111	051124		
6540	031426	050125	000124			
6541	031432	051104	042526	051040	EM34:	.ASCIZ /DRVE RDY CLEAR/
6542	031440	054504	041440	042514		



6543	031446	051101	000			
6544	031451	104	053122	020105	EM35:	.ASCIZ /DRVE POWER LO/
6545	031456	047520	042527	020122		
6546	031464	047514	000			
6547	031467	104	053122	020105	EM36:	.ASCIZ /DRVE UNSAFE/
6548	031474	047125	040523	042506		
6549	031502	000				
6550	031503	127	051520	051440	EM37:	.ASCIZ /WPS SET/
6551	031510	052105	000			
6552	031513	111	052116	051105	EM101:	.ASCIZ /INTERUPT DIDN'T OCUR AFTER WRTE/
6553	031520	050125	020124	044504		
6554	031526	047104	052047	047440		
6555	031534	052503	020122	043101		
6556	031542	042524	020122	051127		
6557	031550	042524	000			
6558	031553	047	051105	023522	EM102:	.ASCIZ /'ERR'OR SET/
6559	031560	051117	051440	052105		
6560	031566	000				
6561	031567	122	042113	020101	EM103:	.ASCIZ /RKDA INCRMENTED WRONG/
6562	031574	047111	051103	042515		
6563	031602	052116	042105	053440		
6564	031610	047522	043516	000		
6565	031615	122	041113	020101	EM104:	.ASCIZ /RKBA INCRMENTED WRONG/
6566	031622	047111	051103	042515		
6567	031630	052116	042105	053440		
6568	031636	047522	043516	000		
6569	031643	122	053513	020103	EM105:	.ASCIZ /RKWC DIDN'T OVRFLO TO 0/
6570	031650	044504	047104	052047		
6571	031656	047440	051126	046106		
6572	031664	020117	047524	030040		
6573	031672	000				
6574	031673	115	054105	041040	EM106:	.ASCIZ /MEX BITS WRONG/
6575	031700	052111	020123	051127		
6576	031706	047117	000107			
6577	031712	051127	042524	041440	EM110:	.ASCIZ /WRTE CHK EROR/
6578	031720	045510	042440	047522		
6579	031726	000122				



```

6636 032342 040504 020040 051040
6637 032350 053513 000103
6638 032354 020040 041520 020040 DH110: .ASCIZ / PC RKCS RKER RKBA RKDA/
6639 032362 020040 051040 041513
6640 032370 020123 020040 051040
6641 032376 042513 020122 020040
6642 032404 051040 041113 020101
6643 032412 020040 051040 042113
6644 032420 000101
6645
6646
6647 .EVEN
6648
6649 032422 001116 001162 001164 DT1: .WORD $ERRPC,$REG0,$REG1,$REG2,$REG3,0
6650 032430 001166 001170 000000
6651
6652 032436 001116 001162 000000 DT2: .WORD $ERRPC,$REG0,0
6653
6654 032444 001116 001162 001164 DT21: .WORD $ERRPC,$REG0,$REG1,$REG2,$REG3,$REG4,$REG5,$REG6,0
6655 032452 001166 001170 001172
6656 032460 001174 001176 000000
6657 032466 001116 001162 001164 DT25: .WORD $ERRPC,$REG0,$REG1,$REG2,$REG3,$REG4,0
6658 032474 001166 001170 001172
6659 032502 000000
6660 032504 001116 001162 001164 DT103: .WORD $ERRPC,$REG0,$REG1,0
6661 032512 000000
6662
6663 ;THIS IS THE DATA BUFFER USED TO WRITE THE RANDOM PATTERNS ON THE
6664 ;DISK AT THE BEGINING. 400 (OCTAL) WORDS ARE WRITTEN AT A TIME, THUS
6665 ;THIS BUFFER IS 400/8 WORDS LONG.
6666
6667 032514 DBUF:
6668 032514 000240 PGEND: NOP
6669 000001 .END

```

ABORT 001672  
ABRT 012046  
ABRT1 012060  
BAMAP 001530  
BASEBA 002052  
BCTST 005352  
BEGEX1 010612  
BEGNEX 010630  
BIT0 = 000001  
BIT00 = 000001  
BIT01 = 000002  
BIT02 = 000004  
BIT03 = 000010  
BIT04 = 000020  
BIT05 = 000040  
BIT06 = 000100  
BIT07 = 000200  
BIT08 = 000400  
BIT09 = 001000  
BIT1 = 000002  
BIT10 = 002000  
BIT11 = 004000  
BIT12 = 010000  
BIT13 = 020000  
BIT14 = 040000  
BIT15 = 100000  
BIT2 = 000004  
BIT3 = 000010  
BIT4 = 000020  
BIT5 = 000040  
BIT6 = 000100  
BIT7 = 000200  
BIT8 = 000400  
BIT9 = 001000  
BLNKS1 002664  
BLNKS2 002663  
BLNKS3 002662  
BPTVEC= 000014  
BUSY 001426  
CBSY 021474  
CHDPRS 022656  
CHFAFN 011154  
CHKBA 020150  
CHKCRD 020272  
CHKCS 020112  
CHKDA 020126  
CHKDRV 016540  
CHKMEX 020204  
CHKRWS 020254  
CHKWC 020230  
CH1 011210  
CH2 011310  
CICNT 001466

CICNT1 001470  
CRSDR = 104407  
CLEANB 006434  
CLRERR 016006  
CLRFLG 015770  
CLRSIN 016152  
CMNABT 014216  
CMND 001326  
CMNERR 014160  
CNOBSY 021712  
CN.RDY 022364  
CN.RST 022356  
COMRET 020306  
CON.RD= 104417  
CON.RE= 104416  
CR = 000015  
CRCMND 020340  
CRLF = 000200  
CROTLF 022170  
CSE = 000002  
CSECN 001652  
CYLMAP 001522  
DATCHK 017110  
DATER 001712  
DBUF 032514  
DDISP = 177570  
DH1 031730  
DH103 032265  
DH105 032327  
DH110 032354  
DH2 031776  
DH21 032013  
DH23 032110  
DH25 032155  
DH27 032232  
DH30 032313  
DISPLA 001142  
DISPRE 000174  
DMPREG 027032  
DOREAD 006424  
DOWRIT 006304  
DOXFER 006312  
DPL = 010000  
DRCMND 020314  
DRMAP 001520  
DRU = 002000  
DRVABT 014426  
DRVCNT 001500  
DRVPRS 001264  
DRVPTR 001476  
DRV.RE= 104420  
DRY = 000200  
DR.RST 022244

DSELECT 016312  
DSWR = 177570  
DT1 032422  
DT103 032504  
DT2 032436  
DT21 032444  
DT25 032466  
ECOUNT 001540  
EMTVEC= 000030  
EM1 027632  
EM10 030162  
EM101 031513  
EM102 031553  
EM103 031567  
EM104 031615  
EM105 031643  
EM106 031673  
EM11 030237  
EM110 031712  
EM12 030317  
EM13 030371  
EM14 030444  
EM15 030517  
EM16 030603  
EM17 030614  
EM2 027650  
EM20 030642  
EM21 030706  
EM23 030720  
EM24 030747  
EM26 031023  
EM27 031076  
EM3 027723  
EM30 031137  
EM31 031232  
EM32 031323  
EM33 031355  
EM34 031432  
EM35 031451  
EM36 031467  
EM37 031503  
EM4 027746  
EM5 027771  
EM6 030056  
EM7 030115  
ERCODE 001474  
ERDRV 001542  
ERINF1 005720  
ERR = 100000  
ERRVEC= 000004  
EXCRLU 015726  
EXCUT 011612  
EXNSK 011470

EXRCSR 007722  
FNCRND 020372  
FNMAP 001526  
FRSTRT 001253  
FTITLE 001252  
GENBUF 016704  
GEN1 014724  
GENBRQ 014500  
GETINF 022064  
GTSDRV 022226  
GTSWR = 104406  
GT3RG 022040  
GT4RG 022032  
HE = 040000  
HECN 001562  
HISTRY 020426  
HRDERR 014016  
HT = 000011  
INTFLG 001534  
INTHND 013256  
INTIFL 001535  
INTISK 012376  
IOTVEC= 000020  
KDPAR0= 172360  
KDPAR1= 172362  
KDPAR2= 172364  
KDPAR3= 172366  
KDPAR4= 172370  
KDPAR5= 172372  
KDPAR6= 172374  
KDPAR7= 172376  
KDPDR0= 172320  
KDPDR1= 172322  
KDPDR2= 172324  
KDPDR3= 172326  
KDPDR4= 172330  
KDPDR5= 172332  
KDPDR6= 172334  
KDPDR7= 172336  
KEY 001306  
KIPAR0= 172340  
KIPAR1= 172342  
KIPAR2= 172344  
KIPAR3= 172346  
KIPAR4= 172350  
KIPAR5= 172352  
KIPAR6= 172354  
KIPAR7= 172356  
KIPDR0= 172300  
KIPDR1= 172302  
KIPDR2= 172304  
KIPDR3= 172306  
KIPDR4= 172310

KIPDR5= 172312  
KIPDR6= 172314  
KIPDR7= 172316  
KWCOUM 001560  
KWHR 001552  
KWLS 001234  
KWLVEC= 000100  
KWMIN 001554  
KWPLVL 001246  
KWSEC 001556  
KWSRVE 022552  
LF = 000012  
MAXBA 002054  
MH1 020746  
MH2 020762  
MH3 020772  
MH4 020775  
MMVEC = 000250  
MSG1 002062  
MSG10 002165  
MSG11 002201  
MSG12 002206  
MSG13 002214  
MSG14 002225  
MSG15 002245  
MSG16 002305  
MSG17 002326  
MSG18 002334  
MSG19 002336  
MSG2 002070  
MSG20 002362  
MSG24 002372  
MSG25 002375  
MSG26 002377  
MSG26A 002474  
MSG27 002532  
MSG28 002604  
MSG29 002652  
MSG3 002076  
MSG4 002104  
MSG5 002120  
MSG6 002133  
MSG7 002141  
MSG8 002146  
MSG9 002156  
NIEROR 021516  
NOEROR 014300  
NRDH 001774  
NRDL 001772  
NWRFNC 011766  
NWRTH 001734  
NWRTL 001732  
NXTDRV 005366

00DEVN	015724	RG4SDR	022222	ST3	005322	TYPMSG=	104421	\$GDDAT	001124
PCMND	002032	RKBA	001226	ST4	004752	TYPOC =	104402	\$GET42	022740
PCNTR	001236	RKCS	001222	SUPRS	025014	TYPON =	104404	\$GTSWR	023034
PDR	001254	RKDA	001230	SUPRSL	025000	TYPOS =	104403	\$HO =	000000
PFSTRT	003366	RKDB	001232	SUP1	025026	TY.MSG	022522	\$HIOCT	023640
PFTERR	013214	RKDS	001216	SUP2	025070	WATIME	022630	\$ICNT	001104
PGEND	032514	RKER	001220	SWR	001140	WCE =	000001	\$ILLUP	027614
PIRQ =	177772	RKSTAT	001242	SWREG	000176	WCECN	001632	\$INTAG	001135
PIRQVE =	000240	RKVEC	001240	SWD =	030001	WCFLG	001456	\$ITEMB	001114
PORKER	013222	RKWC	001224	SW00 =	000001	WCMAP	001532	\$KTNEX	017666
POS	001436	RNUM	017106	SW01 =	000002	WPS =	000040	\$KTOUT	017656
POSCMN	020350	RSBAH	025746	SW02 =	000004	WRDSK	010266	\$KTI1	017524
POSK	012164	RSBAL	025744	SW03 =	000010	WRFNC	012064	\$LF	001214
POSTIO	011414	RSCYLH	025742	SW04 =	000020	XFR	007514	\$LPAOR	001106
PPRLVL	001244	RSCYLL	025740	SW05 =	000040	XFR16K	007504	\$LPEPR	001110
PRICMN	011410	RSORVH	025732	SW06 =	000100	XXDPMO	002060	\$LSTAD	017770
PRSFNC	001462	RSDRVL	025730	SW07 =	000200	\$AUTOB	001134	\$LSTBK	017772
PRD =	000000	RSDTH	025756	SW08 =	000400	\$BDAOR	001122	\$MNEW	023527
PR1 =	000040	RSDTL	025754	SW09 =	001000	\$BDDAT	001126	\$MSWR	023516
PR2 =	003100	RSFUNH	025736	SW10 =	000002	\$BELL	001206	\$MUL =	000003
PR3 =	000140	RSFUNL	025734	SW11 =	002000	\$CHARC	024460	\$MULT	025100
PR4 =	000200	RSWCH	025752	SW12 =	004000	\$CKSWR	022764	\$NULL	001154
PRS =	000240	RSWCL	025750	SW13 =	010000	\$CMTAG	001100	\$NWTST =	000001
PR6 =	000300	RTIPC7	021732	SW14 =	020000	\$CM1 =	000011	\$OCNT	026202
PR7 =	000340	RWS =	000100	SW15 =	040000	\$CM2 =	000022	\$OCTVL	024566
PS =	177776	R6 =	%000006	SW2 =	100060	\$CM3 =	000011	\$OMODE	026204
PSINER	013070	R7 =	%000007	SW3 =	000004	\$CNTLG	023511	\$OVER	027234
PSRTRY	013130	SAVKEY	001536	SW4 =	000010	\$CNTLU	023504	\$PASS	001100
PSTFNC	001464	SAVREG=	104414	SW5 =	000020	\$SCORE	017674	\$POWER	027622
PSW =	177776	SCP =	020000	SW6 =	000040	\$SCRLF	001213	\$PWAD	027610
PWRVEC=	000024	SECMAP	001524	SW7 =	000100	\$SCROUT	017724	\$PWADN	027450
P1	020376	SEXIT	021720	SW8 =	000200	\$SOBLK	024234	\$PWARG	027604
P2	020410	SFTERR	013636	SW9 =	000400	\$SDB20	024604	\$PWRLP	027522
QBUSAD	001514	SIN =	001000	TBITVE =	001000	\$SDB20	024464	\$QUES	001212
QCYL	001504	SINCN	001622	TIMER	001472	\$SDECVL	024764	\$RAND	025576
QDRV	001502	SINCNT	016264	TIMOUT	022354	\$SDIV	025212	\$RDCHR	023246
QEROR	021610	SKCMP	012704	TIMTYP	026660	\$SDOAGN	022760	\$RDECC	023642
QFNC	001512	SKE =	010000	TKVEC =	000060	\$SDTBL	024224	\$RDLIN	023366
QINGER	010650	SKECN	001602	TPVEC =	000064	\$SENDAD	022750	\$RDOCT	023540
QSCNT	001460	SNOTYP	026766	TRAPVE=	000034	\$SENDCT	022734	\$RDSZ =	000010
QSEC	001510	SRODRV	001250	TRTVEC=	000014	\$SEOP	022704	\$REGAD	001160
QSUR	001506	SRNO	001266	TST1	005414	\$SEOPCT	022726	\$REGO	001162
QWRCNT	001516	SRO =	177572	TST2	005526	\$SERFLG	001103	\$REG1	001164
RCNT	017104	SR1 =	177574	TST3	005772	\$SERMAX	001115	\$REG10	001202
ROCHR =	104410	SR2 =	177576	TST4	006106	\$SERROR	026206	\$REG2	001166
RODEC =	104413	SR3 =	172516	TST5	006456	\$SERRPC	001116	\$REG3	001170
ROLIN =	104411	STACK =	001100	TST6	007040	\$SERRTB	002666	\$REG4	001172
ROCT =	104412	START	003376	TST7	007374	\$SERTY	026472	\$REG5	001174
REPCNT	002056	STARTA	003444	TYPOBO	017774	\$SERTTL	001112	\$REG6	001176
REPSTA	021116	STATST	021006	TYPOS =	104405	\$ESCAP	001204	\$REG7	001200
RESREG=	104415	STATUS	021436	TYPE =	104401	\$FILLC	001156	\$RESRE	025540
RESVEC=	000010	STKLMT=	177774	TYPFN	021736	\$FILLS	001155	\$RTNAD	022762
RE*PY	001446	ST2	004400			\$GDADR	001120	\$SAVRE	025502

\$SAVR6	027620	\$SVPC =	000220	\$TPB	001152	\$TSTNM	001102	\$TYPON	026020
\$SCOPE	027204	\$SWR =	143000	\$TPFLG	001157	\$TTYIN	023474	\$TYPOS	025760
\$SETUP=	000115	\$SWRMK=	000000	\$TPS	001150	\$TYPOS	024020	\$XTSTR	027216
\$SIZE	017466	\$TKB	001146	\$TRAP	027350	\$TYPE	024244	\$SGETH=	000000
\$SIZEX	017730	\$TKS	001144	\$TRAP2	027372	\$TYPEC	024414	\$OFILL	026203
\$STUP =	177777	\$TN =	000010	\$TRP =	000022	\$TYPEX	024462	.	= 032516
\$SVLAD	027306	\$TNPWR	024714	\$TRPAD	027404	\$TYPOC	026004		

. ABS. 032516 000

ERRORS DETECTED: 0

DSKZ:DZRKHG/SOL=DSKZ:SYSMAC.SML,DSKM:DZRKHG.P11

RUN-TIME: 15 22 .6 SECONDS

RUN-TIME RATIO: 609/38=15.6

CORE USED: 36k (71 PAGES)