

.REM &

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZRKH-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 ON 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 (9). 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 = NNNNN 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 RETSAPT
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 DIFFERENCE 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 (HASEBA) 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 'GENSRQ' 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 (TST1-TST7) 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<03>

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 KW11 LINE CLOCK IS NOT AVAILABLE ON THE SYSTEM.

6.11 SW<02>

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<01>

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<00>

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 (GEN8RQ) AND PUT IN A QUEUE TO BE PROCESSED. THE ALGORITHM WORKS AS FOLLOWS. COMMANDS IN THE QUEUE ARE PREPOSITIONED (HEADS) BY PERFORMING 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 INTERFERE 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 SIMILIAR 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 PECREATE A SET OF EVENTS THAT LED TO THE EPROR.

9.0 TPANSFER 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 EPROR 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
SFEK 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 SHOULD 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 'DSELECT' 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 (\$TRAP) WILL PICK UP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION AND USE IT TO INDEX THROUGH THE TRAP TABLE (\$TRAPAD) 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 ERROP 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 PEFER TO THEM TROUGH 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-DZRKHG, RK11-RK05 PERFORMANCE EXERCISER
; *COPYRIGHT (C) 1973,1977
; *DIGITAL EQUIPMENT CORP.
; *MAYNARD, MASS. 01784
; *
; *PROGRAM BY JIM KAPADIA
; *
; *THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
; *PACKAGE (MAINDEC=11-DEGAC-C3), JAN 19, 1977.
; *
; *REVISED BY GEORGE GALLANT, TOM SAWYER = MARCH 1976
; *REVISED BY CHUCK HESS = AUGUST 1976
.SBTL OPERATIONAL SWITCH SETTINGS
; *
; * SWITCH USE
; * *****
; * 18 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(GENSRQ)
; * 4 DO NOT REWRITE THE DISKS ON RESTART AT 210
; * 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 SW=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

```

```

.SBTL 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
STKLMT= 177774 ; *STACK LIMIT REGISTER
PIRQ= 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= 8 ; *STACK POINTER
PC= 9 ; *PROGRAM COUNTER
; *PRIORITY LEVEL DEFINITIONS
PR0= 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      $W03= 10
678      000004      $W02= 4
679      000002      $W01= 2
680      000001      $W00= 1
681      ,EQUIV $W09,$W9
682      ,EQUIV $W08,$W8
683      ,EQUIV $W07,$W7
684      ,EQUIV $W06,$W6
685      ,EQUIV $W05,$W5
686      ,EQUIV $W04,$W4
687      ,EQUIV $W03,$W3
688      ,EQUIV $W02,$W2
689      ,EQUIV $W01,$W1
690      ,EQUIV $W00,$W0
691
692      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
693      BIT15= 100000
694      BIT14= 40000
695      BIT13= 20000
696      BIT12= 10000
697      BIT11= 4000
698      BIT10= 2000
699      BIT09= 1000
700      BIT08= 400
701      BIT07= 200
702      BIT06= 100
703      BIT05= 40
704      BIT04= 20
705      BIT03= 10
706      BIT02= 4
707      BIT01= 2
708      BIT00= 1
709      ,EQUIV BIT09,BIT9
710      ,EQUIV BIT08,BIT8
711      ,EQUIV BIT07,BIT7
712      ,EQUIV BIT06,BIT6
713      ,EQUIV BIT05,BIT5
714      ,EQUIV BIT04,BIT4
715      ,EQUIV BIT03,BIT3
716      ,EQUIV BIT02,BIT2
717      ,EQUIV BIT01,BIT1
718      ,EQUIV BIT00,BIT0
719
720      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
721      ERRVEC= 4 ;TIME OUT AND OTHER ERRORS
722      RESVEC= 10 ;RESERVED AND ILLEGAL INSTRUCTIONS
723      TBITVEC=14 ;"T" BIT
724      TRIVEC= 14 ;TRACE TRAP
725      BPTVEC= 14 ;BREAKPOINT TRAP (BPT)
726      IOTVEC= 20 ;INPUT/OUTPUT TRAP (IOT) **SCOPE**
727      PWRVEC= 24 ;POWER FAIL
728      EMIVEC= 30 ;EMULATOR TRAP (EMT) **ERROR**
729      TRAPVEC=34 ;"TRAP" TRAP
730      TKVEC= 60 ;TTY KEYBOARD VECTOR
731      TPVEC= 64 ;TTY PRINTER VECTOR
732      PIRGVEC=240 ;PROGRAM INTERRUPT REQUEST VECTOR
    
```

```

733      000100      KMLVEC=100 ;KNILL CLOCK VECTOR
734
735      ,EQUIV BIT15,ERR
736      ,EQUIV BIT14,ME
737      ,EQUIV BIT13,SCP
738
739
740      ,EQUIV BIT12,DPL
741      ,EQUIV BIT10,DRU
742      ,EQUIV BIT09,$IN
743      ,EQUIV BIT07,DRY
744      ,EQUIV BIT06,$WS
745      ,EQUIV BIT05,$WS
746
747
748
749      ,EQUIV BIT12,$KE
750      ,EQUIV BIT01,CSE
751      ,EQUIV BIT00,WCE
752
753      ;SBTTL TRAP CATCHER
754
755      000000      ,=0
756      ;*ALL UNUSED LOCATIONS FROM 4 = 776 CONTAIN A ",+2,HALT"
757      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
758      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
759      000174      ,=174
760      000174      000000      DISPREG; ,WORD 0 ;SOFTWARE DISPLAY REGISTER
761      000176      000000      $NREG; ,WORD 0 ;SOFTWARE SWITCH REGISTER
762      ;SBTTL STARTING ADDRESS(ES)
763      000900      000137      003376      JMP @START ;JUMP TO STARTING ADDRESS OF PROGRAM
764
765      000210      ,=210 ;RESTART ADDRESS, IF RESTART IS
766      000210      105237      001253      INCB FRSTRT ;DONE AT 210, THE BOUNDARY CONDITION
767      000214      000137      003376      JMP @START ;TESTS (TST1-7) ARE SKIPPED, IF SW 4
768      ;IS SET THEN THE DISKS ARE NOT REWRITTEN
769      ;(WRD$K) WITH RANDOM PATTERNS, NORMALLY
770      ;ALL THE DISKS PRESENT ARE COMPLETELY
771      ;WRITTEN WITH RANDOM PATTERNS, AT THE
772      ;BEGINNING OF THE
773      ;EXERCISER PART OF THE PROGRAM.
774
775      ;SBTTL ACT11 HOOKS
776
777      ;*****
778      ;HOOKS REQUIRED BY ACT11
779
780      000220      $SVPC=, ;SAVE PC
781      000446      ,=46
782      000446      022750      SENDAD ;1)SET LOC,46 TO ADDRESS OF SENDAD IN ,SEOP
783      00052 000000      ,=52
784      00052 000220      ,WORD 0 ;2)SET LOC,52 TO ZERO
785      ,=6SVPC ;RESTORE PC
786
787      ;KT11 REGISTER DEFINITIONS
788      ;SBTTL MEMORY MANAGEMENT DEFINITIONS
    
```

```

789 ;*K11 VECTOR ADDRESSES
790
791 000280 MMVEC= 280
792
793 ;*K11 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 ;SBTTL COMMON TAGS
846
847 ;*****
848 ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
849 ;*USED IN THE PROGRAM.
850
851 001100 .=1100
852 001100 %CNTAG; .WORD 0 ;START OF COMMON TAGS
853 001100 %PASS; .WORD 0 ;CONTAINS PASS COUNT
854 001102 .000 %STNM; .BYTE 0 ;CONTAINS THE TEST NUMBER
855 001103 .000 %ERFLG; .BYTE 0 ;CONTAINS ERROR FLAG
856 001104 .000000 %ICNT; .WORD 0 ;CONTAINS SUBTEST ITERATION COUNT
857 001106 .000000 %LPADR; .WORD 0 ;CONTAINS SCOPE LOOP ADDRESS
858 001110 .000000 %LPERR; .WORD 0 ;CONTAINS SCOPE RETURN FOR ERRORS
859 001112 .000000 %ERTTL; .WORD 0 ;CONTAINS TOTAL ERRORS DETECTED
860 001114 .000 %ITEMB; .BYTE 0 ;CONTAINS ITEM CONTROL BYTE
861 001115 .001 %ERMAX; .BYTE 1 ;CONTAINS MAX. ERRORS PER TEST
862 001116 .000000 %ERRPC; .WORD 0 ;CONTAINS PC OF LAST ERROR INSTRUCTION
863 001120 .000000 %GDADR; .WORD 0 ;CONTAINS ADDRESS OF 'GOOD' DATA
864 001122 .000000 %BDADR; .WORD 0 ;CONTAINS ADDRESS OF 'BAD' DATA
865 001124 .000000 %GDDAT; .WORD 0 ;CONTAINS 'GOOD' DATA
866 001126 .000000 %BDDAT; .WORD 0 ;CONTAINS 'BAD' DATA
867 001130 .000000 .WORD 0 ;RESERVED--NOT TO BE USED
868 001132 .000000 .WORD 0
869 001134 .000 %AUTOB; .BYTE 0 ;AUTOMATIC MODE INDICATOR
870 001135 .000 %INTAG; .BYTE 0 ;INTERRUPT MODE INDICATOR
871 001136 .000000 .WORD 0
872 001140 177570 %SWR; .WORD DSWR ;ADDRESS OF SWITCH REGISTER
873 001142 177570 %DISPLAY; .WORD DDISP ;ADDRESS OF DISPLAY REGISTER
874 001144 177560 %TKB; 177560 ;TTY KBD STATUS
875 001146 177562 %STPB; 177562 ;TTY KBD BUFFER
876 001150 177564 %TPB; 177564 ;TTY PRINTER STATUS REG. ADDRESS
877 001152 177566 %TPB; 177566 ;TTY PRINTER BUFFER REG. ADDRESS
878 001154 .000 %NULL; .BYTE 0 ;CONTAINS NULL CHARACTER FOR FILLS
879 001155 .002 %FILLS; .BYTE 2 ;CONTAINS # OF FILLER CHARACTERS REQUIRED
880 001156 .012 %FILLC; .BYTE 12 ;INSERT FILL CHARS. AFTER A "LINE FEED"
881 001157 .000 %TPFLG; .BYTE 0 ;"TERMINAL AVAILABLE" FLAG (%IT<0>=0=YES)
882 001160 .000000 %REGAD; .WORD 0 ;CONTAINS THE ADDRESS FROM
883 ;WHICH (%REGO) WAS OBTAINED
884 001162 .000000 %REG0; .WORD 0 ;CONTAINS ((%REGAD)+0)
885 001164 .000000 %REG1; .WORD 0 ;CONTAINS ((%REGAD)+2)
886 001166 .000000 %REG2; .WORD 0 ;CONTAINS ((%REGAD)+4)
887 001170 .000000 %REG3; .WORD 0 ;CONTAINS ((%REGAD)+6)
888 001172 .000000 %REG4; .WORD 0 ;CONTAINS ((%REGAD)+10)
889 001174 .000000 %REG5; .WORD 0 ;CONTAINS ((%REGAD)+12)
890 001176 .000000 %REG6; .WORD 0 ;CONTAINS ((%REGAD)+14)
891 001200 .000000 %REG7; .WORD 0 ;CONTAINS ((%REGAD)+16)
892 001202 .000000 %REG10; .WORD 0 ;CONTAINS ((%REGAD)+20)
893 001204 .000000 %ESCAPE; .WORD 0 ;ESCAPE ON ERROR ADDRESS
894 001206 177507 000377 %BELL; .ASCIZ <207><377><377> ;CODE FOR BELL
895 001212 .077 %QUES; .ASCII ?? ;QUESTION MARK
896 001213 .015 %CRLF; .ASCII <15> ;CARRIAGE RETURN
897 001214 .000012 %LF; .ASCIZ <12> ;LINE FEED
898
899 001216 177400 RKDS; .WORD 177400
900 001220 177402 RKER; .WORD 177402
    
```

```

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      KWL8:  .WORD 177546 ;STATUS REGISTER FOR KWILL
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 ;P&N TO BE USED ON INTERRUPT
912
913 001244 000240      PPRVL: .WORD 240 ;PROGRAM PRIORITY LEVELS, 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 KWILL CLOCK SERVICE
917 ;ROUTINE.
918
919 001280 177777      BRDRV: .WORD 177777 ;'BRDRV' CONTAINS THE DRIVE NO WHOOSE SERIAL
920 ;NO IS TO BE TYPED OUT WHEN AN ERROR OCCURS,
921 ;IF SW 1 IS SET, WHEN (BRDRV)=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 ;THIS TABLE CONTAINS (IN ASCENDING ORDER) THE DRIVE NUMBERS THAT ARE
929 ;PRESENT, THUS IF 3 DRIVES 0,1,2 ARE PRESENT; PDR WILL CONTAIN PDR1 WILL
930 ;CONTAIN 1 AND PDR2 WILL CONTAIN 2. THE UPPER BIT OF EACH 'PDR' BYTE IS SET IF THE
931 ;CORRESPONDING DRIVE IS AN 'F' DRIVE.
932
933
934 001254 000010      PDR:  .BLKS 10
935
936 001264 000000      DRVPRS: .WORD 0 ;CONTAINS TOTAL NUMBER OF DRIVES PRESENT
937
938 ;THE FOLLOWING LOCATIONS CONTAIN SERIAL NUMBERS CORRESPONDING TO EACH
939 ;DRIVE. THE SERIAL NUMBERS ARE KEYED IN BY THE USER, WHEN THE PROGRAM
940 ;IS STARTED WITH SWITCH 1 SET TO 1. THIS FEATURE IS NORMALLY USED IN
941 ;PRODUCTION ENVIRONMENT.
942
943 001266 000010      BRNO:  .BLKW 10 ;SERIAL NO'S FOR DRIVES 0-7
944
945
946 ;THE FOLLOWING 8 KEYS ARE FOR THE 8 COMMANDS IN THE QUEUE, TO BE
947 ;EXECUTED ON DIFFERENT DRIVES. EACH KEY IS ASSOCIATED WITH AN EXECUTABLE
948 ;COMMAND ON THE RK11. VARIOUS BITS OF THE KEY DESCRIBE A COMMAND
949 ;AS INDICATED BELOW
950
951 ;<0-2> DRIVE NUMBER ON WHICH THE COMMAND IS TO BE EXECUTED
952 ;<4> INDICATES THAT THE HEADS ARE BEING/OR HAVE BEEN
953 ;POSITIONED ON THE DRIVE
954 ;<5> INDICATES A 'WRT CHK' SHOULD BE DONE FOLLOWING THE 'WRITE'
955 ;<6> INDICATES A WRITE CHECK FUNCTION HAS BEEN INITIATED
956 ;<7> INDICATES THAT A FUNCTION IS IN PROGRESS (IT IS NOT SET
957 ;WHEN POSITIONING IS BEING DONE ON A DRIVE)
958 ;<8-10> INDICATES THE POSITION OF THIS KEY IN THE 8-KEY TABLE
959 ;(POSITIONS BEING 0,1,2,3,4,5,6,7)
960 ;<11> INDICATES THAT FUNCTION CORRESPONDING TO THIS KEY HAS
961 ;BEEN ABORTED
962 ;<12> INDICATES HIGH PRIORITY FOR THE COMMAND (NORMALLY
963 ;SET AFTER AN ERROR OCCURED ON THE COMMAND)
964 ;<14> INDICATES THAT THE COMMAND CORRESPONDING TO THIS KEY HAS BEEN
965 ;ABORTED BECAUSE THE DRIVE WAS Deselected (DSELECT)
966 ;<15> INDICATES THAT THE COMMAND HAS BEEN COMPLETED
967 ;(ALSO SET WHEN COMMAND IS ABORTED AFTER RETRIEVE)
968
969
970 001266 000010      KEY:  .BLKW 10 ;KEY FOR THE COMMANDS IN QUEUE
971
972 ;THE PARAMETERS TO BE USED FOR EACH COMMAND IN THE QUEUE
973
974 ;ARE STORED IN A TABLE STARTING AT 'CMND'. BITS <8-10>
975 ;OF THE COMMAND KEYS (KEY, KEY2, ---KEY8) ARE USED TO POINT
976 ;TO THE RIGHT SET OF PARAMETERS.
977
978
979 ; WORD 1 CONTAINS RKDA TO BE USED
980 ; WORD 2 CONTAINS RKCS (FUNCTION BITS ONLY)
981 ; WORD 3 CONTAINS RKWC (WORD COUNT 2'S COMP)
982 ; WORD 4 CONTAINS RKBA
983
  
```

```

984 00142b 000040 CMND: .BLKB 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 "BUSY#" WILL CONTAIN 310, 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 001436 000010 POS: .BLKB 10 ;DRIVE 0 POSITIONED
999
1000 ;RETRY COUNTS FOR A PARTICULAR FUNCTION ON A DRIVE THE FUNCTION IS ABORTED
1001 ;ON A DRIVE WHEN THE RETRY COUNT REACHES 3.
1002
1003 RETRY: .BLKB 10 ;DRIVES 0-7 RERTY COUNTS
1004
1005
1006 001446 000010 WCFG: .WORD 0 ;IF BIT 15 IS SET WRITE CHK IS TO BE DONE
1007 ;FOLLOWING THE WRITE, BITS 0-3 CONTAIN THE
1008 ;OFFSET TO KEY# (FROM BASE=KEY)
1009
1010
1011
1012 001460 000000 QSCNT: .WORD 0 ;THIS IS A COUNT FOR KEEPING TRACK OF THE TIME
1013 ;TAKEN BY ALL THE 8 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 CICT: .WORD 0 ;THIS IS A COUNT-TIMER USED FOR KEEPING TRACK
1024 001470 000000 CICNT: .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 "GENSRQ"
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 OFNC: .WORD 0
1038 001514 000000 QBUSAD: .WORD 0
1039 001516 000000 QWRCNT: .WORD 0
  
```

```

1040 ;THIS TABLE CONTAINS VARIOUS MAPPING FACTORS TO BE USED
1041 ;FOR GENERATING RANDOM PARAMETERS FROM RANDOM NUMBERS
1042
1043
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 INTPLG: .BYTE 0 ;FOR "INTHND", CLEARED ON ENTERING HANDLER
1058 001535 000 INT1FL: .BYTE 0 ;FOR "INT1SK", 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 KMIN: .WORD 0 ;COUNTS MINUTES (2'S COMPLEMENT)
1073 001556 000000 KSEC: .WORD 0 ;COUNTS SECONDS (2'S COMPLEMENT)
1074 001560 000000 KNCOUNT: .WORD 0 ;COUNTS CPS FROM KWHIL (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: .BLKB 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: .BLKB 10 ;DRIVE 0-7 SEEK ERROR COUNTS
1086
1087 ;THIS TABLE CONTAINS COUNTS FOR SIN ERRORS ON A
1088 ;PARTICULAR DRIVE
1089
1090 001622 000010 SINCN: .BLKB 10 ;DRIVE 0-7 SIN COUNTS
1091
1092 ;THIS TABLE CONTAINS COUNTS FOR WRITE CHECK ERRORS
1093 ;THAT OCCURED ON A PARTICULAR DRIVE
1094
1095
  
```

1096 001A32 000010 WCCCN: ,BLKW 10 ;WCE COUNT FOR DRIVES 0-7
 1097
 1098 ;THIS TABLE CONTAINS COUNTS FOR CHECK SUM ERROR THAT
 1099 ;OCCURED ON A PARTICULAR DRIVE
 1100
 1101
 1102 001A32 000010 CSERN: ,BLKW 10 ;CSE COUNT FOR DRIVES 0-7
 1103
 1104 ;THIS TABLE CONTAINS COUNT OF NUMBER OF FUNCTIONS
 1105 ;THAT WERE ABORTED ON A PARTICULAR DRIVE, A
 1106 ;FUNCTION IS ABORTED ONLY AFTER DOING RETRIES
 1107
 1108
 1109 001A72 000010 ABORT: ,BLKW 10 ;ABORT COUNT FOR DRIVES 0-7
 1110
 1111 ;COUNTS FOR NUMBER OF DATA ERRORS THAT OCCURED ON INDIVIDUAL DRIVES.
 1112
 1113 001712 000010 DATER: ,BLKW 10 ;DRIVES 0-7

1114 001732 000000 NWRTL: ,WORD 0 ;LO WORD: OF THE 2 WORD COUNT-GIVING TOTAL
 1115 001734 000000 NNRT: ,WORD 0 ;HI WORD: OF WORDS WRITTEN ON DRIVE 0
 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 0
 1121 001776 000016 ,BLKW 14 ;FOR DRIVES 1-7
 1122
 1123 002A32 001326 PCMND: ,WORD CMND ;POINTERS TO PARAMETERS FOR COMMANDS IN QUEUE
 1124 002A34 001336 ,WORD CMND+10 ;POINTER TO SECOND COMMAND
 1125 002A36 001346 ,WORD CMND+20 ;POINTER TO THIRD COMMAND
 1126 002A40 001356 ,WORD CMND+30 ;POINTER TO FOURTH COMMAND
 1127 002A42 001366 ,WORD CMND+40 ;POINTER TO FIFTH COMMAND
 1128 002A44 001376 ,WORD CMND+50 ;POINTER TO SIXTH COMMAND
 1129 002A46 001406 ,WORD CMND+60 ;POINTER TO SEVENTH COMMAND
 1130 002A50 001416 ,WORD CMND+70 ;POINTER TO EIGHTH COMMAND
 1131
 1132
 1133 002A52 000000 BASEBA: ,WORD 0 ;CONTAINS THE LOWEST BUS ADDRESS STARTING WHICH DATA TRANSFERS
 1134 ;CAN BE DONE
 1135 002A54 000000 MAXBA: ,WORD 0 ;CONTAINS THE HIGHEST BUS ADDRESS TO WHICH DATA TRANSFERS
 1136 ;CAN BE DONE,
 1137 002A56 000000 REPCNT: ,WORD 0 ;CONTAINS THE REPETITION COUNT- THE NUMBER
 1138 ;OF TIMES Q 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 002A60 000000 XDPND: ,WORD 0 ;LOW BYTE CONTAINS ADDRESS OF RK08 DRIVE
 1145 ;WHICH PROGRAM WAS LOADED FROM; HIGH BYTE
 1146 ;CONTAINS THE RK08 'XXDP' CODE.
 1147
 1148 ;ASCII MESSAGES
 1149
 1150 002A62 005015 045523 000105 MSG1: ,ASCII <15><12>/SKE/
 1151 002A70 005015 041527 000105 MSG2: ,ASCII<15><12>/NCE/
 1152 002A76 005015 051503 000105 MSG3: ,ASCII<15><12>/CSE/
 1153 002104 005015 040510 042122 MSG4: ,ASCII <15><12>/HARD EROR/
 1154 002112 042440 047522 000122
 1155 002120 047440 020116 047504 MSG5: ,ASCII / ON DOING /
 1156 002126 047111 020107 000
 1157 002133 127 044522 042524 MSG6: ,ASCII /WRITE/
 1158 002140 000
 1159 002141 122 040505 000104 MSG7: ,ASCII /READ/
 1160
 1161 002146 051127 020124 044103 MSG8: ,ASCII /WRT CHK/
 1162 002156 042122 041440 045510 MSG9: ,ASCII /RD CHK/
 1163 002164 000
 1164 002165 015 040412 047502 MSG10: ,ASCII <15><12>/ABORTED/<15><12>
 1165 002172 052122 042103 005015
 1166 002200 000
 1167 002201 123 042505 000113 MSG11: ,ASCII /SEEK/
 1168 002206 005015 041520 000075 MSG12: ,ASCII <15><12>/PC=/
 1169 002214 044120 051531 041040 MSG13: ,ASCII /PHYS RAM/

1170	002922	036501	000																	
1171	002925	015	047012	020117	MSG14:	,ASCIZ	<15><12>/NO DRVS PRSNT/													
1172	002932	051104	081526	090040																
1173	002940	051523	082116	000																
1174	002945	015	042012	083122	MSG15:	,ASCIZ	<15><12>/DRVE # DIDN'T INTERRUPT AFTER /													
1175	002952	020105	020043	044804																
1176	002960	047104	082047	044440																
1177	002966	082116	081109	080128																
1178	002974	020124	043101	042824																
1179	002902	020122	000																	
1180	002905	015	045412	054805	MSG16:	,ASCIZ	<15><12>/KEY=0 BUSY=7/													
1181	002912	034055	020040	041040																
1182	002920	051525	026531	000067																
1183	002926	051440	020122	047516	MSG17:	,ASCII	/ BR NO/													
1184	002934	000072			MSG18:	,ASCIZ	/ /													
1185	002936	005015	042040	047522	MSG19:	,ASCIZ	<15><12>/ DROPPED DRIVE # /													
1186	002944	050120	042105	042040																
1187	002952	044522	042526	021440																
1188	002960	000040																		
1189	002962	005015	051104	093111	MSG20:	,ASCIZ	<15><12>/DRIVE/													
1190	002970	000105																		
1191	002972	020054	000		MSG24:	,ASCIZ	/ /													
1192	002975	106	000		MSG25:	,ASCIZ	/ /													
1193	002977	015	042012	044522	MSG26:	,ASCIZ	<15><12>/DRIVE WRD8 WRITN WRD8 READ C8E WCE DATERR HE /													
1194	002404	042526	020040	051127																
1195	002412	051504	053440	044522																
1196	002420	047124	020040	051127																
1197	002426	051504	051040	040505																
1198	002434	020104	020040	041440																
1199	002442	042523	020040	020040																
1200	002450	041527	020105	040504																
1201	002456	042524	051122	020040																
1202	002464	020040	044040	020105																
1203	002472	000040																		
1204	002474	005015	051104	053111	MSG26A:	,ASCIZ	<15><12>/DRIVE SKE ABORT SIN/													
1205	002502	020105	020040	020040																
1206	002510	045523	020105	040440																
1207	002516	047502	052122	020040																
1208	002524	020040	044523	000116																
1209	002532	005015	047125	041101	MSG27:	,ASCIZ	<15><12>/UNABLE TO CLEAR ERROR AFTER THREE TRIES/													
1210	002540	042514	052040	020117																
1211	002546	046103	040505	020122																
1212	002554	051105	047522	020122																
1213	002562	043101	042524	020122																
1214	002570	044124	042522	020105																
1215	002576	051124	042511	000123																
1216	002604	005015	051105	047522	MSG28:	,ASCIZ	<15><12>/ERROR CONDITION CLEARED ON RETRY # /													
1217	002612	020122	047503	042116																
1218	002620	052111	047511	020116																
1219	002626	046103	040505	042522																
1220	002634	020104	047117	051040																
1221	002642	042105	054522	021440																
1222	002650	000040																		
1223	002652	005015	044524	042515	MSG29:	,ASCIZ	<15><12>/TIME /													
1224	002660	000040																		
1225																				

1226	002662	040	BLNK83:	,ASCII	/ /
1227	002663	040	BLNK82:	,ASCII	/ /
1228	002664	000040	BLNK81:	,ASCIZ	/ /
1229				,EVEN	

```

1230          ,SBTTL  ERROR POINTER TABLE
1231
1232          ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR,
1233          ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
1234          ;*LOCATION ITEM#, THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT,
1235          ;*NOTE1: IF ITEM# IS 0 THE ONLY PERTINENT DATA IS ($ERRPC),
1236          ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
1237
1238          ;*      EM          ;;POINTS TO THE ERROR MESSAGE
1239          ;*      DH          ;;POINTS TO THE DATA HEADER
1240          ;*      DT          ;;POINTS TO THE DATA
1241          ;*      DF          ;;POINTS TO THE DATA FORMAT
1242
1243
1244          002666          $ERRTS:
1245          ;*THERE ARE TWO CLASSES OF ERRORS:
1246          ;*1. ERRORS IN EXERCISER PART OF THE PROGRAM = ERROR NUMBERS BELOW 100
1247          ;*2. ERRORS IN THE NON-EXERCISER PART OF THE PROGRAM = ERROR NUMBERS EQUAL
1248          ;*TO AND GREATER THAN 100.
1249          ;*THE DOCUMENT CONTAINS MORE INFORMATION ON THESE.
1250          ;*THE FOLLOWING ERRORS OCCUR IN THE EXERCISER PART OF THE PROGRAM.
1251
1252          ;ITEM  1
1253
1254          EM1          ;ERROR ON WRITE
1255          DH1          ;PC  RKCS  RKER  RKDS  RKDA
1256          DT1          ;ERRPC $REG0 $REG1 $REG2 $REG3
1257          0
1258
1259          ;ITEM  2
1260
1261          EM2          ;ATTEMPT TO INITIATE FUNCTION ON 'BUSY' DRIVE
1262          DH2          ;PC  DRIVE
1263          DT2          ;ERRPC $REG0
1264          0
1265
1266          ;ITEM  3
1267
1268          EM3          ;CONTROL READY NOT SET
1269          DH1          ;PC  RKCS  RKER  RKDS  RKDA
1270          DT1          ;ERRPC $REG0 $REG1 $REG2 $REG3
1271          0
1272
1273          ;ITEM  4
1274
1275          EM4          ;R/W/S READY NOT SET
1276          DH1          ;PC  RKCS  RKER  RKDS  RKDA
1277          DT1          ;ERRPC $REG0 $REG1 $REG2 $REG3
1278          0
1279
1280          ;ITEM  5
1281
1282          EM5          ;CONTROL READY NOT SET AFTER FIRST INTERRUPT ON ISSUING SEEK
1283          DH1          ;PC  RKCS  RKER  RKDS  RKDA
1284          DT1          ;ERRPC $REG0 $REG1 $REG2 $REG3
1285          0
  
```

```

1286          002734          000000          0
1287
1288
1289          ;ITEM  6
1290
1291          EM6          ;WRONG BITS IN RKCS, EXPECT SEEK
1292          DH1          ;PC  RKCS  RKER  RKDS  RKDA
1293          DT1          ;ERRPC $REG0 $REG1 $REG2 $REG3
1294          0
1295
1296          ;ITEM  7
1297
1298          EM7          ;'BUSY' FLAG CLEAR ON INTERRUPTING DRIVE
1299          DH2          ;PC  DRIVE
1300          DT2          ;ERRPC $REG0
1301          0
1302
1303          ;ITEM  10
1304
1305          EM10         ;'POSITIONING' FLAG FOR INTERRUPTING DRIVE CLEAR
1306          DH2          ;PC  DRIVE
1307          DT2          ;ERRPC $REG0
1308          0
1309
1310          ;ITEM  11
1311
1312          EM11         ;'ERROR SET AFTER FIRST INTERRUPT ON ISSUING SEEK
1313          DH1          ;PC  RKCS  RKER  RKDS  RKDA
1314          DT1          ;ERRPC $REG0 $REG1 $REG2 $REG3
1315          0
1316
1317          ;ITEM  12
1318
1319          EM12         ;SCP SET AFTER FIRST INTERRUPT ON ISSUING SEEK
1320          DH1          ;PC  RKCS  RKER  RKDS  RKDA
1321          DT1          ;ERRPC $REG0 $REG1 $REG2 $REG3
1322          0
1323
1324          ;ITEM  13
1325
1326          EM13         ;CONTROL READY NOT SET AFTER SEEK DONE INTERRUPT
1327          DH1          ;PC  RKCS  RKER  RKDS  RKDA
1328          DT1          ;ERRPC $REG0 $REG1 $REG2 $REG3
1329          0
1330
1331          ;ITEM  14
1332
1333          EM14         ;INTERRUPTING DRIVE (SEEK DONE) WAS NOT 'BUSY'
1334          DH1          ;PC  RKCS  RKER  RKDS  RKDA
1335          DT1          ;ERRPC $REG0 $REG1 $REG2 $REG3
1336          0
1337
1338          ;ITEM  15
1339
1340          EM15         ;R/W/S READY NOT SET FOR INTERRUPTING DRIVE (SEEK DONE)
1341          DH1          ;PC  RKCS  RKER  RKDS  RKDA
  
```



```

1454 ;ITEM 35
1455
1456 003926 031451 EM35 ;DRIVE POWER LOW
1457 003930 031730 DH1 ;PC RKCS RKER RKDS RKDA
1458 003932 032422 DT1 ;ERRPC $REG0 $REG1 $REG2 $REG3
1459 003934 000000 0
1460
1461 ;ITEM 36
1462
1463 003936 031467 EM36 ;DRIVE UNSAFE
1464 003940 031730 DH1 ;PC RKCS RKER RKDS RKDA
1465 003942 032422 DT1 ;ERRPC $REG0 $REG1 $REG2 $REG3
1466 003944 000000 0
1467
1468 ;ITEM 37
1469
1470 003946 031503 EM37 ;MPS SET
1471 003950 031730 DH1 ;PC RKCS RKER RKDS RKDA
1472 003952 032422 DT1 ;ERRPC $REG0 $REG1 $REG2 $REG3
1473 003954 000000 0
1474
1475 ;*
1476 ;*THE FOLLOWING ERRORS OCCUR IN THE NON-EXERCISER PART OF THE PROGRAM.
1477 ;*
1478
1479 ;ITEM 100
1480
1481 003956 030720 EM23 ;DATA (COMPARISON) ERROR
1482 003960 032110 DH23 ;PC RKBA EXPCT RECYD RKDA
1483 003962 032422 DT1 ;ERRPC $REG0 $REG1 $REG2 $REG3
1484 003964 000000 0
1485
1486 ;ITEM 101
1487
1488 003966 031513 EM101 ;INTERRUPT DID NOT OCCUR AFTER WRITE
1489 003970 031730 DH1 ;PC RKCS RKER RKDS RKDA
1490 003972 032422 DT1 ;ERRPC $REG0 $REG1 $REG2 $REG3
1491 003974 000000 0
1492
1493 ;ITEM 102
1494
1495 003976 031553 EM102 ;'ERR'OR SET
1496 003980 031730 DH1 ;PC RKCS RKER RKDS RKDA
1497 003982 032422 DT1 ;ERRPC $REG0 $REG1 $REG2 $REG3
1498 003984 000000 0
1499
1500 ;ITEM 103
1501
1502 003986 031567 EM103 ;RKDA INCREMENTED WRONGLY
1503 003990 032265 DH103 ;PC EXPCT RECYD
1504 003992 032504 DT103 ;ERRPC $REG0 $REG1
1505 003994 000000 0
1506
1507 ;ITEM 104
1508
1509 003996 031619 EM104 ;RKBA INCREMENTED WRONGLY
  
```

```

1510 003920 032265 DH103 ;PC EXPCT RECYD
1511 003922 032504 DT103 ;ERRPC $REG0 $REG1
1512 003924 000000 0
1513
1514 ;ITEM 105
1515
1516 003926 031643 EM105 ;RKC DID NOT OVERFLOW TO 0
1517 003930 032327 DH105 ;PC RKDA RKC
1518 003932 032504 DT103 ;ERRPC $REG0 $REG1
1519 003934 000000 0
1520
1521 ;ITEM 106
1522
1523 003936 031673 EM106 ;MEX BITS INCORRECT
1524 003940 031730 DH1 ;PC RKCS RKER RKDS RKDA
1525 003942 032422 DT1 ;ERRPC $REG0 $REG1 $REG2 $REG3
1526 003944 000000 0
1527
1528 ;ITEM 107
1529
1530 003946 030720 EM23 ;DATA (COMPARISON) ERROR ON READ
1531 003950 032265 DH103 ;PC EXPCT RECYD
1532 003952 032504 DT103 ;ERRPC $REG0 $REG1
1533 003954 000000 0
1534
1535 ;ITEM 110
1536
1537 003956 031712 EM110 ;WRITE CHECK ERROR
1538 003960 032384 DH110 ;PC RKCS RKER RKBA RKDA
1539 003962 032422 DT1 ;ERRPC $REG0 $REG1 $REG2 $REG3
1540 003964 000000 0
  
```

```

1541                                     ;IF POWER FAILED, ON RETURN OF POWER ENTER HERE.
1542
1543 003166 004737 022630                PFSTRT: JSR      PC,WATIME                ;WAIT SOME TIME
1544 003172 103237 001253                INCB      FRSTRT                        ;INDICATE THAT THE STATISTICS HAVE
1545                                     ;TO BE SAVED, ON RETRN FROM PWR FAIL.
1546
1547
1548
1549
1550 003176 000005                START:  RESET                       ;CLEAR THE BUS
1551                                     ;GIVE DRIVES TIME TO LOAD IF PROGRAM WAS STARTED BY APT
1552 003400 023737 000042 000046          CMP      #42,#46
1553 003406 001016                BNE      STARTA
1554 003410 005077 175614                CLR      @RKDA                          ;SELECT UNIT 0
1555 003414 012700 000250                MOV      #250,R0                        ;WAIT FOR..
1556 003420 032777 000200 175570 208:  BIT      #200,@RKDS                    ;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      208
1563 003442 000000                HALT
1564 003444                STARTA:                               ;RKDS BIT 7 (DRIVE READY) NEVER SET.
1565                                     ;SBTTL INITIALIZE THE COMMON TAGS
1566                                     ;CLEAR THE COMMON TAGS (@CMTAG) AREA
1567 003444 012706 001100                MOV      @CMTAG,R6                      ;FIRST LOCATION TO BE CLEARED
1568 003450 005026                CLR      (R6)+                          ;CLEAR MEMORY LOCATION
1569 003452 022706 001140                CMP      @SWR,R6 ;DONE?
1570 003456 001374                BNE      =6                             ;LOOP BACK IF NO
1571 003460 012706 001100                MOV      @STACK,SP                     ;SETUP THE STACK POINTER
1572                                     ;INITIALIZE A FEW VECTORS
1573 003464 012737 027204 000020          MOV      @SCOPE,@IOTVEC ;IOT VECTOR FOR SCOPE ROUTINE
1574 003472 012737 000340 000022          MOV      #340,@IOTVEC+2 ;LEVEL 7
1575 003500 012737 027350 000034          MOV      @TRAP,@TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
1576 003506 012737 000340 000036          MOV      #340,@TRAPVEC+2 ;LEVEL 7
1577 003514 012737 027450 000024          MOV      @PNRDN,@PWRVEC ;POWER FAILURE VECTOR
1578 003522 012737 000340 000026          MOV      #340,@PWRVEC+2 ;LEVEL 7
1579 003530 012737 000330 001106          MOV      @,@LPADR                      ;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1580 003536 012737 000336 001110          MOV      @, @LPERR                     ;SETUP THE ERROR LOOP ADDRESS
1581                                     ;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
1582                                     ;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1583 003544 013746 000004                MOV      @ERRVEC,-(SP) ;SAVE ERROR VECTOR
1584 003550 012737 003604 000004          MOV      #648,@ERRVEC ;SET UP ERROR VECTOR
1585 003556 012737 177570 001140          MOV      @DWR,SWR ;SETUP FOR A HARDWARE SWICH REGISTER
1586 003564 012737 177570 001142          MOV      @DISP,DISPLAY ;HARD A HARDWARE DISPLAY REGISTER
1587 003572 022777 177777 175340          CMP      #1,@SWR ;TRY TO REFERENCE HARDWARE SWR
1588 003580 001012                BNE      668 ;BRANCH IF NO TIMEOUT TRAP OCCURRED
1589                                     ;AND THE HARDWARE SWR IS NOT = -1
1590 003602 000403                BR      668 ;BRANCH IF NO TIMEOUT
1591 003604 012716 003612 648:  MOV      @SWR,(SP) ;SET UP FOR TRAP RETURN
1592 003610 000002                RTI
1593 003612 012737 000176 001140 658:  MOV      @SWREG,SWR ;POINT TO SOFTWARE SWR
1594 003620 012737 000174 001142          MOV      @DISPREG,DISPLAY ;DISPREG,DISPLAY
1595 003626 012637 000004 668:  MOV      (SP)+,@ERRVEC ;RESTORE ERROR VECTOR
1596

```

```

1597 003632 023737 000042 000046          CMP      #42,#46 ;ARE WE IN ACT11 AUTOMATIC MODE?
1598 003640 001416                BEQ      698 ;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      678 ;BRANCH IF NO
1603 003650 104401 003706                TYPE     ,698 ;TYPE ASCIZ STRING
1604                                     ;SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
1605 003654 005737 000042                TST     @42 ;ARE WE RUNNING UNDER XDP/ACT?
1606 003660 001006                BNE      698 ;BRANCH IF YES
1607 003662 023727 001140 000176          CMP      SWR,@SWREG ;SOFTWARE SWITCH REG SELECTED?
1608 003670 001005                BNE      708 ;BRANCH IF NO
1609 003672 104406                GTSWR  ;GET SOFT-SWR SETTINGS
1610 003674 000403                BR      708
1611 003676 112737 000001 001134 698:  MOV     @1,@AUTOB ;SET AUTO-MODE INDICATOR
1612 003704 708:
1613 003706 000433                BR      678 ;GET OVER THE ASCIZ
1614                                     ;688: .ASCIZ <CRLF>*RK11/RK05 PERFORMANCE EXERCISER* <15><12>*MAINDEC-11-DZRKH-G* <CRLF>
1615                                     ;678:
1616 003774 012737 026206 000030          MOV     @ERROR,@EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
1617 004002 013737 001244 000032          MOV     PPLVL,@EMTVEC+2 ;LEVEL 8
1618 004010 012737 022552 000100          MOV     @KNSRVE,@KNLVEC ;KN111 CLOCK SERVICE
1619 004016 013737 001246 000102          MOV     KNPLVL,@KNLVEC+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     XDPMD ;CLEAR 'XDP' 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,XDPMD ;GET DEVICE INDICATOR AND DRIVE ADDRESS OF
1629                                     ;LOADING RK05
1630 004046 122737 000010 002060          CMPB   #10,XDPMD ;VALID DRIVE ADDRESS ?
1631 004054 101002                BHI     28 ;BR IF YES
1632 004056 105037 002060                CLR    XDPMD ;CHANGE TO DRIVE ZERO
1633 004062 005737 000042 28:  TST    42 ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
1634 004066 001424                BEQ    38 ;BR IF NEITHER
1635 004070 104401 004076                TYPE   ,728 ;TYPE ASCIZ STRING
1636 004074 000413                BR     718 ;GET OVER THE ASCIZ
1637                                     ;728: .ASCIZ <15><12>/NOT TESTING DRIVE /
1638                                     ;718:
1639 004124 005046                CLR     =(SP) ;CLEAR WORD ON STACK
1640 004126 113716 002060          MOV    XDPMD,(SP) ;GET DRIVE ADDRESS
1641 004132 104403                TYPOS  ;TYPE THE ADDRESS
1642 004134 001 ;BYTE 1 ;ONLY 1 CHARACTER
1643                                     ;BYTE 0 ;SUPPRESS LEADING ZEROS
1644 004136 000520                BR     ST2 ;GET NUMBER OF DRIVES
1645 004140 005227 177777 38:  INC    #1 ;FIRST TIME THROUGH HERE ?
1646 004144 001115                BNE     ST2 ;BR IF NOT
1647 004146 104401 004154                TYPE   ,748 ;TYPE ASCIZ STRING
1648 004152 000411                BR     738 ;GET OVER THE ASCIZ
1649                                     ;738: .ASCIZ <15><12>/TO TEST DRIVE /
1650 004176
1651 004176 005046                CLR     =(SP) ;CLEAR WORD ON THE STACK
1652 004200 113716 002060          MOV    XDPMD,(SP) ;GET DRIVE ADDRESS

```

1653	004704	104403		TYPOS			;TYPE THE DRIVE ADDRESS
1654	004706	001		,BYTE	1		;ONLY 1 CHARACTER
1655	004707	000		,BYTE	0		;SUPPRESS LEADING ZEROS
1656	004710	104401	004216	TYPE	,768		;TYPE ASCIZ STRING
1657	004714	000431		BR	758		;GET OVER THE ASCIZ
1658				,ASCIZ	/HALT PROGRAM, REMOVE RKDP PACK AND REPLACE IT/<15><12>		
1659	004700			758:			
1660	004700	104401	004306	TYPE	,768		;TYPE ASCIZ STRING
1661	004704	000435		BR	778		;GET OVER THE ASCIZ
1662				,ASCIZ	/WITH A WORK PACK, CLEAR LOCATION 40, AND RESTART PROGRAM/		
1663	004400			778:			
1664							
1665							
1666							
1667	004400	104416		BT2:	CON,RESET		
1668	004402	005037	001264	CLR	DRVPR8		;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	SPDR,R1		
1676	004436	012703	001306	MOV	SKY,R3		
1677	004442	010277	174562	18:	MOV	R2,SRKDA	;SELECT A DRIVE
1678	004446	032777	000200	174562	BIT	SRKDA	;IS IT IN SYSTEM?
1679	004454	001415		BEQ	SR		;NO
1680	004456	010237	001802	MOV	R2,GDRV		;LOAD DRIVE ADDRESS INTO GDRV
1681	004462	104420		DRV,RESET			;RESET THE DRIVE
1682	004464	005737	002060	TST	XXDPM		;PROGRAM LOADED FROM AN RK05 ?
1683	004470	001403		BEQ	SR		;BR IF NOT
1684	004472	120037	002060	CMPB	R0,XXDPM		;LOADED FROM THIS RK05 ?
1685	004476	001404		BEQ	SR		;BR IF YES
1686	004500	110021		28:	MOVB	R0,(R1)+	;STORE THE DRIVE NUMBER
1687	004502	010223		MOV	R2,(R2)+		;STORE ADDRESS IN KEY TABLE
1688	004504	005237	001264	INC	DRVPR8		;BUMP THE NUMBER OF DRIVES COUNTER
1689	004510	062702	020000	38:	ADD	SRKDA,R2	;NEXT DRIVE ADDRESS
1690	004514	005200		INC	R0		;NEXT DRIVE NUMBER
1691	004516	022700	000010	CMP	SR,R0		;DONE ALL DRIVES???
1692	004522	001347		BNE	SR		;LOOP TILL DONE
1693	004524	013703	001264	MOV	DRVPR8,R3		;FIND WHICH DRIVES ARE TYPE F
1694	004530	001510		BEQ	BT4		;BR IF NOT DRIVES PRESENT
1695	004532	012701	001254	MOV	SPDR,R1		
1696	004536	005000		CLR	R0		
1697	004540	005002		CLR	R2		
1698	004542	104401	002362	TYPE	,MSQ20		
1699	004546	111102		48:	MOVB	(R1),R2	;GET DRIVE NUMBER
1700	004550	010200		MOV	R2,R0		
1701	004552	002472		BLT	SR		
1702							
1703	004554	104401	002372	TYPE	,MSQ24		
1704							
1705	004560	010246		MOV	R2,=(SP)		;TYPE THE DRIVE NUMBER
1706	004562	104403		TYPOS			
1707	004564	001		,BYTE	1		
1708	004566	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 BIT16	
1712	004574	006002		ROR	R2		;BIT0 TO BIT14	
1713	004576	006002		ROR	R2		;BIT0 TO BIT13	
1714	004600	042702	017777	BIC	SR,SR		;CLEAR ANY EXTRANEOUS BITS	
1715								
1716	004604	010237	001502	MOV	R2,GDRV			
1717	004610	104420		DRV,RESET			;RESET THE DRIVE VIA GDRV	
1718								
1719	004612	032702	020000	RIT	SRKDA,R2		;EVEN DRIVE NUMBER???	
1720	004616	001003		BNE	SR		;NO - CLEAR BIT	
1721								
1722	004620	052702	020000	BIS	SRKDA,R2		;MAKE IT AN ODD DRIVE	
1723	004624	000402		BR	SR			
1724								
1725	004626	042702	020000	58:	BIC	SRKDA,R2	;MAKE IT AN EVEN DRIVE	
1726								
1727	004632	010277	174372	68:	MOV	R2,SRKDA	;SELECT THE NEW DRIVE	
1728	004636	032777	000200	174372	BIT	SRKDA	;MAKE SURE DRIVE IS IN SYSTEM	
1729	004644	001420		BEQ	SR		;IF NOT, SKIP THIS TEST	
1730								
1731	004646	012777	000011	174346	MOV	SR,SRKDA	;START A SEEK TO CYL 0	
1732	004654	104417		CON,RDY			;WAIT FOR CONTROLLER	
1733	004656	032777	000100	174332	BIT	SR,SRKDA	;IS IT IN MOTION???	
1734	004664	001010		BNE	SR		;NO - J TYPE DRIVE	
1735								
1736	004666	152711	000200	BISB	SRKDA,(R1)		;YES - SET THE F TYPE BIT	
1737	004672	104401	002375	TYPE	,MSQ25			
1738								
1739	004676	032777	000100	174312	78:	BIT	SRKDA	;WAIT FOR HEADS TO STOP
1740	004704	001774		BEQ	SR			
1741								
1742	004706	105737	001253	88:	TSTB	FRSTRT		
1743	004712	001012		BNE	SR			
1744								
1745	004714	032777	000002	174216	BIT	SR,SRNR	;SERIAL NO. SW SET?	
1746	004722	001406		BEQ	SR		;NO	
1747								
1748	004724	104401	002326	TYPE	,MSQ17		;TYPE "SR NO"	
1749	004730	104413		RDDEC			;READ FROM TTY INPUT	
1750	004732	006300		ASL	R0		;SAVE SERIAL NO FOR THE DRIVE	
1751	004734	012660	001266	MOV	(SP)+,SRNR(R0)			
1752								
1753	004740	005201		98:	INC	R1		
1754	004742	005303		DEC	R3			
1755								
1756	004744	003300		BOT	SR			
1757	004746	104401	001213	TYPE	,SCRLF			

```

1757 ;'MAXBA' IS THE HIGHEST BUS ADDRESS (MEMORY) TO WHICH DATA TRANSFER CAN
1758 ;BE DONE BY THE PROGRAM.
1759 ;'MAXBA' IS FIGURED USING THE FOLLOWING ALGORITHM;
1760
1761 ;1. IF KT11 IS NOT PRESENT,
1762 ;A. AND THE PROGRAM IS RUN UNDER XXDP, THEN THE TOP 1,5 K IS RESERVED
1763 ;AND THE 'MAXBA' IS COMPUTED ($LSTAD-6000).
1764 ;B. AND THE PROGRAM IS NOT RUNNING UNDER XXDP, THEN THE TOP 320 WORDS
1765 ;ARE RESERVED FOR 'MOM',LOADER,ETC. AND THE 'MAXBA' IS COMPUTED ($LSTAD-800).
1766
1767 ;2. IF KT11 IS PRESENT,
1768 ;A. AND MORE THAN 28K MEMORY IS PRESENT, THEN THE MAXIMUM BUS ADDRESS
1769 ;IS 147776 (OCTAL).
1770 ;B. AND LESS THAN 28K IS PRESENT, THEN THE TOP 2K IS RESERVED FOR RKDP
1771 ;MONITOR AND 'MAXBA' IS COMPUTED.
1772 ;FIGURE OUT THE AVAILABLE MEMORY AND 'MAXBA'
1773
1774 004752 004737 017466 ST4: JSR PC,$SIZE ;GO SIZE THE MEMORY
1775 004756 012702 002052 MOV $BASEBA,R2 ;INITIALIZE POINTERS
1776 004762 012703 002054 MOV $MAXBA,R3
1777 004766 005737 017524 TST $KT11 ;KT11 AVAILABLE?
1778 004772 100022 BPL 48 ;NO
1779 004774 013700 017772 MOV $LSTBK,R0 ;GET THE LAST BANK OF MEMORY
1780 005A00 020027 001540 CMP RO,$1540 ;28K OR MORE?
1781 005A04 002012 BGE 38 ;YES
1782
1783 005A06 162700 000040 SUB $40,R0 ;BACK UP 2 K'S (RKDP MONITOR, ETC.)
1784 005A12 012701 177772 MOV $6,R1 ;AND FORM THE MAXIMUM BUS ADDRESS
1785 ;FOR DATA TRANSFER
1786
1787 005A16 006300 18: ASL R0
1788 005A20 005201 INC R1
1789 005A22 001375 BNE 18
1790 005A24 162700 000002 SUB $2,R0
1791 005A30 000415 28: BR 68
1792
1793 005A32 012713 147776 38: MOV $147776,(R3) ;FOR 28K OR MORE, THIS IS THE 'MAXBA'
1794 005A36 000413 BR 78
1795
1796 005A40 013700 017770 48: MOV $LSTAD,R0 ;KT11 NOT PRESENT, GET THE LAST
1797 ;AVAILABLE ADDRESS
1798
1799 005A44 005737 000040 58: TST $840 ;'XXDP' LOADED PROGRAM ?
1800 005A50 001003 BNE 88 ;YES
1801 005A52 162700 000800 SUB $800,R0 ;NO, SAVE THE LAST 320 WORDS
1802 005A54 000402 BR 68
1803 005A56 162700 006000 68: SUB $6000,R0 ;SAVE THE LAST 1,5K OF MEMORY (RKDP
1804 005A64 010013 88: MOV RO,(R3) ;MONITOR, ETC.)
1805 ;SAVE THE MAXIMUM BUS ADDRESS(MAXBA) TO
1806 ;WHICH DATA TRANSFER CAN BE DONE SAFELY
1807 005A66 012712 032514 78: MOV $PGEND,(R2) ;'BASEBA'
1808 005A77 032777 000100 174040 BIT $806,$8WR
1809 005100 001510 BEQ 87
1810 005102 104401 005110 TYPE ,658 ;TYPE ASCII STRING
1811 005106 000432 BR 648 ;GET OVER THE ASCII
1812 005174 ;;658: ,ASCII <15><12>/TYPE OCTAL BUS ADDRESSES FOR DATA XFER, BETWEEN /
648:

```

```

1813 005174 011246 MOV (R2),-(SP) ;'BASEBA'
1814 005176 104402 TYPOC
1815 005200 104401 005206 TYPE ,678 ;TYPE ASCII STRING
1816 005204 000402 BR 668 ;GET OVER THE ASCII
1817 ;;678: ,ASCII / & /
688:
1818 005212 MOV (R3),-(SP) ;'MAXBA'
1819 005212 TYPOC
1820 005214 104402 98:
1821 005216 TYPE ,698 ;TYPE ASCII STRING
1822 005216 104401 005224 BR 688 ;GET OVER THE ASCII
1823 005222 000407 ;;698: ,ASCII <15><12>/LO LIMIT? /
688:
1824
1825 005242 RDOCT
1826 005242 104412 MOV (SP)+,R0
1827 005244 012600 CMP RO,(R2) ;CORRECT LO LIMIT?
1828 005246 020012 BLO 98
1829 005250 103762 CMP RO,(R3) ;CORRECT LO LIMIT?
1830 005252 020013 BHS 98
1831 005254 103360 MOV RO,(R2) ;'BASEBA'
1832 005256 010012 108:
1833 005260 TYPE ,718 ;TYPE ASCII STRING
1834 005260 104401 005266 BR 708 ;GET OVER THE ASCII
1835 005264 000407 ;;718: ,ASCII <15><12>/HI LIMIT? /
708:
1836
1837 005304 RDOCT
1838 005304 104412 MOV (SP)+,R0
1839 005306 012600 CMP RO,(R3) ;CORRECT HI LIMIT?
1840 005310 020013 BHI 108
1841 005312 101362 CMP RO,(R2) ;CORRECT LO LIMIT?
1842 005314 020012 BLOS 108
1843 005316 101760 MOV RO,(R3) ;'MAXBA'
1844 005320 010013
1845
1846 005322 023727 002054 037476 ST3: CMP MAXBA,$37476 ;8K MEMORY = CLOBBER XXDPT
1847 005330 002003 BGE 18
1848 005332 012737 037476 002054 MOV $37476,MAXBA ;BUT SAVE LOADER
1849 005340 105737 001253 18: TST FRSTRT ;PROGRAM RESTARTED AT 210?
1850 005344 001402 BEQ BCTST ;NO
1851 005346 000137 007772 JMP EXRCBR ;YES, SKIP TEST 1 TO 7

```

```

1852 ;THIS IS THE BEGINING OF THE CONSTRAINED TESTS AIMED AT CHECKING THE
1853 ;DIFFERENT BOUNDARY CONDITIONS OF RK11/RK05
1854
1855 ;FIND OUT THE DRIVE NUMBER TO BE TESTED,
1856 BCTST: MOV #KEY,DRVPTR ;INITIALIZE PTR TO DRV#
1857 MOV #DRVPR,DRVCNT ;NUMBER OF DRIVES PRESENT
1858 NXTDRV: MOV #DRVPTR,QDRV ;SAVE DRIVE #(BITS 15-13)
1859 ADD #2,DRVPTR ;INCREMENT PTR TO NXT DRV#
1860 DEC DRVCNT ;DONE ALL DRIVES?
1861 BPL TST1 ;NO, GO TEST THIS DRIVE
1862 JMP EXRCER ;ALL DONE, GO TO EXERCISER PART
    
```

```

1863 ;*****
1864 ;TEST 1 PERFORM WRITE OF 401 WORDS (1 SECTOR + 1 WORDS)
1865 ;THIS TEST PERFORMS A WRITE OF 401 WORDS (1 SECTOR + 1WORD) AND
1866 ;CHECKS IF RKDA,RKBA,RKWC INCREMENTED CORRECTLY,WRITING IS DONE
1867 ;ON CYLINDER 0, SURFACE 0, SECTORS 0,1 AND 10,11, IT SHOULD BE
1868 ;NOTED THAT THIS IS A BOUNDARY CONDITION TRANSFER, THE VALIDITY
1869 ;OF THE TRANSFER IS CHECKED IN THE NEXT TEST,
1870 ;DATA PATTERN WRITTEN IS 111111.
1871 ;*****
1872 TST1: SCOPE
1873
1874 MOV QDRV,R1 ;GET RKDA
1875 MOV R1,R2 ;SAVE RKDA
1876 ADD #2,R2 ;EXPCTD RKDA AFTER WRITE IS DONE
1877
1878 MOV #18,SLPERR ;RETURN ADDRESS FOR LUPING
1879
1880 CON,RESET
1881 DRV,RESET
1882 CON,RESET
1883 MOV #111111,DBUF ;CLEAR MASK BITS IN POLLING LOGIC
1884 MOV #401,R3 ;PATTERN TO BE WRITTEN
1885 JBR PC,DOWRITE ;WORD COUNT FOR WRITE
1886 ;GO DO WRITE
1887 ERROR 101 ;INTERRUPT DID NOT OCCUR AFTER WRITE
1888 JBR PC,CHKCB ;CHECK ERROR BIT IN RKCB
1889 ERROR 102 ;ERROR BIT IN RKCB SET ON DOING WRITE
1890 JBR PC,CHRDA ;CHECK IF RKDA INCREMENTED RIGHT
1891 ERROR 103 ;RKDA DID NOT INCREMENT RIGHT AFTER
1892 ;A WRITE OF 401 WORDS,
1893 JBR PC,CHKNC ;CHECK IF RKWC OVERFLOWED TO 0
1894 ERROR 105 ;RKWC DID NOT OVERFLOW TO 0 AFTER
1895 ;A WRITE OF 401 WORDS,
1896 BIT #10,R1 ;SECTORS 10,11 WRITTEN?
1897 BNE TST2 ;YES
1898 ADD #12,R1 ;RKDA TO BE USED NEXT (SEC 10)
1899 ADD #16,R2 ;EXPCTD RKDA AFTER WRITE IS DONE
1900 BR 28 ;GO WRITE SECS 10,11
    
```



```

1901 ;*****
1902 ;TEST ? READ & CHECK THAT 401 WORD WRITE WAS DONE CORRECTLY
1903 ;THIS TEST PERFORMS A READ OF THE 401 WORDS WRITTEN IN THE
1904 ;PREVIOUS TEST AND CHECKS THAT THEY WERE CORRECTLY READ, MOREOVER
1905 ;IT CHECKS THAT ONLY ONE NON-ZERO WORD (401TH) WAS WRITTEN IN THE
1906 ;SECOND SECTOR AND THE REST OF THE WORDS ARE ALL ZEROS,
1907 ;*****
1908 TST2: BCUPE
1909
1910 005430 000004
1911 005430 013701 001502 001110 MOV QDRV,R1 ;GET DRIVE #
1912 005434 012737 005542 101 MOV #10,SLPERR ;ADDRESS FOR LUPING ON ERROR
1913 005444 104416 CON,RESET
1914 005446 104416 DRV,RESET
1915 CON,RESET
1916
1917 005450 004737 006434 JSR PC,CLEANBUF ;CLEAN UP THE DATA BUFFER
1918 ;INTO WHICH READ WILL
1919 ;BE DONE
1920 MOV #1000,R3 ;WORD COUNT
1921
1922 005460 004737 006424 JSR PC,DOREAD ;GO DO A READ OF 2 SECTORS
1923 ;FROM DISK ADDRESS GIVEN IN R1
1924 ;INTERRUPT DID NOT OCCUR AFTER
1925 ;READ OF 401 WORDS WAS DONE.
1926 005466 104101 ERROR 101 ;CHECK IF ERROR BIT IN RKCS SET?
1927 ;ERROR BIT IN RKCS SET ON DOING A
1928 ;READ OF 401 WORDS.
1929 005474 012704 032514 MOV #DBUF,R4 ;STARTING BUS ADDRESS, INTO WHICH READ
1930 005600 010402 MOV R4,R2 ;WAS DONE
1931 005602 004737 020150 JSR PC,CHKBA ;CHECK IF RKBA INCREMENTED RIGHT
1932 005606 104104 ERROR 104 ;RKBA DID NOT INCREMENT RIGHT AFTER READ
1933 ;OF 401 WORDS.
1934 005610 012705 177764 MOV #-14,R5 ;ALLOW 12 ERRORS, AT THE MOST
1935 005614 022712 111111 20: CMP #111111,(R2) ;CORRECT DATA READ?
1936 005620 001410 BEQ 38 ;YES
1937 005622 012737 111111 001164 MOV #111111,#REG1 ;GET EXPCTD DATA WORD
1938 005630 004737 005720 JSR PC,ERINF1 ;GET ERROR INFORMATION
1939 ;DATA ERROR OCCURRED WHEN A
1940 ;READ OF 401 WORDS WAS DONE
1941 ;THE DISK ADDRESS FROM WHERE
1942 ;THE DATA WAS READ INCORRECTLY
1943 ;IS GIVEN IN THE ERROR MESSAGE
1944 005636 005205 INC R5 ;REPORT 12 ERRORS AT MOST
1945 005640 001421 BEQ 68
1946 005642 005722 30: TST (R2)+ ;INCREMENT POINTER
1947 005644 020227 033516 CMP R2,#DBUF+1002 ;CHECKED ALL 401 WORDS?
1948 005650 001361 BNE 28
1949
1950 005652 005712 40: TST (R2) ;CHECK THAT REST OF 377 WORDS
1951 005654 001407 BEQ 58 ;ARE ALL 0'S
1952 005656 005037 001164 CLR #REG1 ;GET EXPCTD DATA WORD (0)
1953 005662 004737 005720 JSR PC,ERINF1 ;GET ERROR INFO
1954 005666 104100 ERROR 100 ;DATA ERROR, IN A PREVIOUS
1955 ;TEST A WRITE OF 401 WORDS
1956 ;(1 SECTOR + 1 WORD) WAS DONE

```

```

1957 ;NOW THESE 2 SECTORS WERE
1958 ;READ IN THE SECOND SECTOR
1959 ;THE FIRST WORD IS A NON-ZERO
1960 ;WORD (WHICH WAS WRITTEN BEFORE)
1961 ;THE REST OF 377 WORD
1962 ;SHOULD BE ALL ZEROS, IF
1963 ;THE WRITE WAS DONE CORRECTLY
1964 ;(& READ IS DONE CORRECTLY)
1965
1966 005670 005205 INC R5 ;REPORT 12 ERRORS AT MOST
1967 005672 001404 BEQ 68
1968 005674 005722 50: TST (R2)+ ;ALL WORDS CHECKED?
1969 005676 020227 034514 CMP R2,#DBUF+2000
1970 005702 001363 BNE 40 ;IF NOT GO BAK
1971
1972 005704 032701 000010 60: BIT #10,R1 ;WERE SECTORS 10,11 READ
1973 005710 001030 BNE TST3 ;YES
1974 005712 062701 000012 ADD #12,R1 ;FROM NEW RKDA, SEC 10
1975 005716 000711 BR 18 ;GO BACK AND READ FROM SECS 10,11
1976
1977 ;ERINF1
1978 ;AT THE TIME OF ENTRY:
1979 ;R2 CONTAINS ERRORING BUS ADDRESS (WHERE DATA ERROR OCCURRED),
1980 ;(R2) CONTAINS BAD DATA THAT WAS READ BACK FROM DISK,
1981 ;R1 CONTAINS DISK ADDRESS WHERE READ BEGAN.
1982
1983 005720 010237 001162 ERINF1: MOV R2,#REG0 ;GET BUS ADDRESS OF DATA ERROR
1984 005724 011237 001166 MOV (R2),#REG2 ;GET BAD DATA WORD (READ)
1985 005730 001046 MOV R1,#(SP)
1986 005732 020227 033512 CMP R2,#DBUF+776 ;FIGURE OUT THE DISK ADDRESS
1987 005736 003001 BGT 18 ;WHERE DATA ERROR OCCURRED
1988 005740 005316 DEC (SP)
1989 005742 005216 INC (SP)
1990 005744 032716 10: BIT #10,(SP)
1991 005750 001405 BEQ 28
1992 005752 032716 BIT #4,(SP)
1993 005756 001402 BEQ 28
1994 005760 062716 ADD #4,(SP)
1995 005764 012637 001170 20: MOV (SP)+,#REG3
1996 005770 000207 RTS PC

```

```

1997 ;*****
1998 ;*TEST 3 PERFORM WRITE OF 12 SECTORS + 1 WORD
1999 ;THIS TEST CHECKS FOR ANOTHER BOUNDARY CONDITION, IT
2000 ;PERFORMS A WRITE OF 12 SECTORS + 1 WORD, RKDA,RKBA,
2001 ;RKWC ARE CHECKED TO SEE IF THEY ARE INCREMENTED CORRECTLY,
2002 ;VALIDITY OF THE DATA WRITTEN IS CHECKED IN THE NEXT
2003 ;TEST, DATA IS WRITTEN ON SECTORS 0=11, SURFACE 0
2004 ;CYLINDER 0 (6001TH WORD ON SECTOR 0, SURFACE 1, CYL 0),
2005 ;ALSO ON SECTORS 0=11, SURFACE 1 (6001TH WORD ON SECTOR
2006 ;0, CYL 1)
2007 ;*****
2008 TST3: SCOPE
2009 MOV QDRV,R1 ;GET DRIVE #
2010 MOV #18,SLPERR ;LUP ON ERROR TO '18'
2011 MOV R1,R2
2012 ADD #21,R2
2013 MOV #6001,R3
2014 CON,RESET
2015 DRV,RESET
2016 CON,RESET
2017
2018
2019 MOV #44444,DBUF ;PATTERN TO BE WRITTEN
2020
2021 JSR PC,DMWRITE ;GO DO WRITE
2022
2023 ERROR 101 ;INTERUPT DID NOT OCCUR ON
2024 ;COMPLETION OF WRITE
2025 JSR PC,CHKCS ;CHECK IF EROR BIT IN RKCS SET
2026 ERROR 102 ;EROR BIT IN RKCS SET ON DOING WRITE
2027 JSR PC,CHKDA ;CHECK IF RKDA INCREMENTED RIGHT
2028 ERROR 103 ;RKDA DID NOT INCREMENT CORRECTLY
2029 ;AFTER A WRITE OF 6001 (OCTAL) WORDS,
2030 ;(12 SECTORS + 1)
2031 JSR PC,CHKWC ;CHECK IF RKWC OVERFLOWED TO 0
2032 ERROR 105 ;RKWC DID NOT OVERFLOW TO 0
2033 BIT #20,R1 ;RKWC DID NOT OVERFLOW TO 0
2034 BNE TST4 ;WRITTEN ON SURFACE 1?
2035 MOV R2,R1 ;YES
2036 ADD #20,R2 ;SURFACE 1
2037 MOV #8401,R3 ;WORD COUNT
2038 BR 18 ;GO WRITE SURFACE 1
  
```

```

2039 ;*****
2040 ;*TEST 4 READ & CHECK THAT 6001 WORD WRITE WAS DONE CORRECTLY
2041 ;THIS TEST CHECKS THAT THE 6001-WORD WRITE THAT WAS DONE IN THE
2042 ;PREVIOUS TEST WAS CORRECT, ESPECIALLY THE LAST 401 WORDS, THE
2043 ;FIRST WORD OF THE SECTOR( IN WHICH THE 6001TH WORD) IS WRITTEN
2044 ;IS THE ONLY NON-ZERO WORD IN THAT SECTOR, THE REST 377 WORDS ARE
2045 ;ALL ZEROS, IF THE WRITING WAS DONE CORRECTLY.
2046 ;*****
2047 TST4: SCOPE
2048 MOV QDRV,R1 ;GET DRIVE #
2049 ADD #13,R1 ;DISK ADDRESS FROM WHERE READ IS DONE
2050 MOV #18,SLPERR
2051 CON,RESET
2052 DRV,RESET
2053 CON,RESET
2054
2055 JSR PC,CLEANBUF ;CLEAN UP THE BUFFER INTO WHICH
2056 ;READ WILL BE DONE
2057 ;SET UP RKDA
2058 ;SECTOR 11, SURFACE 0
2059 MOV #1000,R3 ;WORD COUNT
2060
2061 JSR PC,DONREAD ;GO READ 1000 WORDS (2 SECS)
2062 ERROR 101 ;INTERUPT DID NOT OCCUR AFTER
2063 ;COMPLETION OF READ
2064 JSR PC,CHKCS ;CHECK IF EROR BIT IN RKCS SET
2065 ERROR 102 ;EROR (RKCS) SET ON DOING READ
2066 MOV #DBUF,R4 ;STARTING BA OF DATA BUFR
2067 MOV R4,R2
2068 JSR PC,CHKBA ;RKBA INCREMENTED CORRECTLY?
2069 ERROR 104 ;RKBA DID NOT INCREMENT CORRECTLY
2070 MOV #-14,R8
2071 CMP #44444,(R3) ;DATA WORD OK?
2072 BEQ 38
2073 MOV #44444,REG0 ;NO, GET EXPECTD DATA WORD
2074 JSR PC,ERINP1 ;GET, OTHER ERROR INFO
2075 ERROR 100 ;DATA ERROR, A WRITE OF 6001
2076 ;WORDS (12 SECS + 1 WORD) WAS DONE
2077 ;IN A PREVIOUS TEST, THE LAST TWO
2078 ;SECTORS (LAST 401 WORDS) WERE READ
2079 ;BACK, THIS ERROR INDICATES THAT
2080 ;SEC #11 (LAST BUT ONE SECTOR) GAVE
2081 ;BAD DATA WORDS
2082 ;REPORT 12 ERRORS AT MOST
2083 INC R8
2084 BEQ 68
2085 TST (R2)+ ;INCREMENT POINTER TO BA
2086
2087 CMP R2,#DBUF+1002 ;CHECKED 401 WORDS?
2088 BNE 28
2089 TST (R2) ;CHECK THAT THE REMAINING 377
2090 BEQ 58 ;WORDS OF THE LAST SECTOR (SEC #0)
2091 CLR #REG0 ;WERE READ BACK AS 0'S
2092 JSR PC,ERINP1
2093 ERROR 100 ;DATA ERROR, IF WRITE WAS DONE CORRECTLY
2094 ;IN THE PREVIOUS TEST, THE LAST SECTOR
2095 ;OF THE DATA BLOCK (12 SECS + 1 WORD)
2096 ;SHOULD CONTAIN ONLY 1 (FIRST) WORD
  
```

```

2095 ;AS NON-ZERO, THE REST 377 SHOULD BE
2096 ;ALL 0'S. THIS ERROR INDICATES THAT
2097 ;THE SOME OF 377 WORDS
2098 ;WERE NOT CORRECT
2099 ;REPORT 12 ERRORS AT MOST
2100 006954 008209 INC R6
2101 006958 001404 BEQ 68
2102 006960 008722 581 TST (R2)+ ;INCREMENT POINTER
2103 006962 020227 034514 CMP R2,DBUF+2000 ;CHECKED ALL WORDS?
2104 006964 001363 BNE 48 ;NO
2105 006970 032701 000020 581 BIT #20,R1 ;DONE CHECKING FOR SURFACE 1?
2106 006974 001070 BNE T8T5 ;YES
2107 006976 062701 000020 ADD #20,R1 ;SO SET UP FOR SURFACE 1
2108 006978 000711 BR 18 ;GO BACK & READ SURFACE 1
    
```

```

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 #38, @RKVEC ;SET UP INTERRUPT VECTOR
2125 006342 005046 CLR -(SP) ;NEW PSW
2126 006344 012746 006352 MOV #18, -(SP) ;SET NEW PC TO STACK *****
2127 006350 000002 RTI
2128 006352 052777 000101 172642 18; 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 28; INC CICNT ;WAIT FOR INTERRUPT
2132 006376 001375 BNE 4
2133
2134 006400 005237 001470 INC CICNT1
2135 006404 001372 BNE 28
2136
2137 006406 004737 022032 JSR PC, @T4RG ;TIMED OUT, INTERRUPT DID NOT OCCUR
2138 006412 000207 RTS PC ;RETURN TO THE ERROR MESSAGE
2139
2140 006414 022626 38; CMP (SP)+, (SP)+ ;RESTORE STACK POINTER
2141 006416 062716 000002 ADD #2, (SP) ;ADJUST RETURN ADDRESS TO SKIP OVER
2142 006422 000207 RTS PC ;ERROR MESSAGE ON RETURN
    
```

```

2143 ;THIS ROUTINE PERFORMS A READ ON THE DISK, AT THE TIME OF ENTRY R1 CONTAINS
2144 ;THE DISK ADDRESS FROM WHERE THE READ IS TO BE DONE, R3 CONTAINS THE WORD
2145 ;COUNT (RKWC). READ WILL BE DONE INTO DATA BUFFER AT 'DBUF'.
2146
2147 ;READ IS DONE IN INTERRUPT MODE. IF THE INTERRUPT DOES NOT OCCUR WITHIN
2148 ;A CERTAIN TIME, RETURN IS MADE TO THE ERROR MESSAGE FOLLOWING THE 'JBR'
2149 ;CALL. IF THE INTERRUPT OCCURS AS EXPECTED, RETURN ADDRESS IS ADJUSTED
2150 ;TO SKIP OVER THE ERROR MESSAGE.
2151
2152 006424 012777 000004 172570 DOREAD: MOV #4, @RKCS ;READ
2153 006432 000727 BR DOXFER
2154
2155 ;CLEANBUF
2156 ;CLEANS OUT THE DATA BUFFER (ALL WORDS WRITTEN TO 177777) INTO WHICH THE
2157 ;READ FROM THE DISK WILL BE DONE. DATA BUFFER STARTS AT 'DBUF' AND IS
2158 ;1600 (OCTAL) WORDS LONG.
2159
2160
2161 006434 012702 177000 CLEANBUF: MOV #1000, R2 ;SET COUNT
2162 006440 012708 032514 MOV @DBUF, R5 ;INITIALIZE BA
2163 006444 012725 022222 10: MOV #22222, (R5)+
2164 006480 005202 INC R2 ;DONE ALL WORDS?
2165 ;BUFFER STARTING AT (PHYSICAL) BUS
2166 ;ADDRESS 177000 (177000-200776)
2167 006452 001374 BNE 10
2168 006454 000207 RTS PC ;YES RETURN
    
```

```

2169 ;*****
2170 ;TEST 5 CHECK DATA TRANSFER AROUND 32K BOUNDARY
2171 ;*THIS TEST PERFORMS A WRITE OF 2 SECTORS ON THE DISK FROM MEMORY
2172 ;*LOCATIONS AROUND THE 32K BOUNDARY. SECTORS 0,1, CYL 0, SURFACE
2173 ;*0 ARE WRITTEN. PHYSICAL BUS ADDRESSES FOR THE DATA BUFFER:
2174 ;* 177000 TO 200776 I.E. (32K-256) TO (32K+256)
2175
2176 ;*CHECKING IS DONE TO SEE IF MEX BITS, RKBA, RKDA, RKWC INCREMENTED
2177 ;*CORRECTLY. THEN DATA BUFFER IS CLEARED OUT AND A READ IS DONE
2178 ;*INTO IT. A CHECK IS MADE TO SEE IF THE CORRECT DATA IS RECEIVED.
2179 ;*ONLY 12 DATA ERRORS ARE REPORTED.
2180 ;*****
2181 006456 000004 TSTS: SCOPE
2182 006460 023727 017772 002000 CMP #LSTBK, @2000 ;32K OR MORE OF MEMORY?
2183 006466 103002 BHS 10 ;YES
2184 006470 000137 007040 JMP TST6 ;IF NOT, DONT DO THIS TEST
2185 006474 012737 001600 172352 10: MOV #1600, @RKIPAR5 ;MAP 28-32K THRU PAR 5
2186 006502 012737 002000 172354 MOV #2000, @RKIPAR6 ;MAP 32-36K THRU PAR 6
2187 006510 012737 000001 172572 MOV #1, @SRO ;TURN ON MEMORY MANAGEMENT
2188
2189 ;SET UP DATA BUFFER (1000 OCTAL WORDS LONG) FOR WRITING TWO SECTORS
2190 ;ON THE DISK. THE TRANSFER IS DONE AROUND THE 32K BOUNDARY FROM
2191 ;BUS ADDRESS (PHYSICAL) 177000 TO 200776, (32K-256) TO (32+256)
2192
2193 ;DATA IN THE BUFFER IS A COUNT PATTERN STARTING FROM 1 FOR THE FIRST
2194 ;WORD TO 000 FOR THE LAST WORD.
2195 ;PHYSICAL BUFFER ADDRESS: 177000 TO 200776 (32K-256 TO 32K+256)
2196 006516 012737 006524 001110 20: MOV #20, @LVERR ;LUP TO 20 ON ERROR (SW 9)
2197 006524 104416 CON, RESET
2198 006526 104420 DRV, RESET
2199
2200 006530 012700 000001 MOV #1, R0 ;INITIALISE DATA PATTERN TO BE
2201 ;WRITTEN
2202 006534 012701 137000 30: MOV #137000, R1 ;BA TO START PHYSICAL ADDRESS=177000
2203 006540 010021 MOV R0, (R1)+ ;WRITE COUNT PATTERN (1-1000)
2204 006542 005200 INC R0 ;INTO DATA BUFFER (PHYS ADDRESS
2205 006544 020027 001001 CMP R0, #1001 ;177000 TO 200776)
2206 006550 001373 BNE 30
2207 006552 013777 001502 172450 MOV @DRV, @RKDA ;SUR 0, SEC 0, CYL 0
2208 006560 012777 177000 172438 MOV #1000, @RKWC ;WORD COUNT #2 SECS
2209 006568 012777 177000 172432 MOV #177000, @RKBA ;BUS ADDRESS.
2210 006574 012777 000003 172420 MOV #3, @RKCS ;WRITE, GO
2211 006602 104417 CON, RDY ;WAIT FOR CONTROL RDY
2212
2213 006604 004737 020112 JSR PC, CHKCS ;ANY ERROR IN RKCS?
2214 006610 104102 ERROR 102 ;'ERR' SET IN RKCS, ON DOING A
2215 ;WRITE OF SECTORS (0,1) FROM
2216 ;(PHYSICAL) BUS ADDRESSES 177000
2217 ;(177000 TO 200776)
2218 006612 004737 020204 JSR PC, CHRMEX
2219 ;CHECK THAT RKBA OVERFLOWED INTO
2220 ;EXTENDED MEM, BIT (0) OF RKCS (BIT 4)
2221 ;EX MEM BIT 0 SET?
2222 ;GET RKCS, ER, DS, DA
2223 006616 104106 ERROR 106 ;MEX BITS INCORRECT, BIT 4 OF RKCS
2224 ;(MEX BIT 0) SHOULD HAVE SET
    
```

```

2225
2226
2227
2228 006420 012703 001000      MOV      #1000,R3
2229 006424 012704 177000      MOV      #177000,R4
2230 006430 004737 020150      JSR      PC,CHKBA
2231 006434 104104      ERROR   104
2232
2233
2234
2235 006436 013702 001502      MOV      QDRV,R2
2236 006442 082702 000002      ADD      #2,R2
2237 006446 004737 020126      JSR      PC,CHKDA
2238 006452 104103      ERROR   103
2239
2240
2241
2242
2243 006454 004737 020230      JSR      PC,CHKWC
2244 006460 104103      ERROR   103
2245
2246
2247
2248
2249
2250
2251
2252 006662 012737 006670 001110      MOV      #48,$LPERR
2253 006670 104416      48:    CON,RESET
2254 006672 104420      DRV,RESET
2255
2256 006674 012701 137000      MOV      #137000,R1
2257 006700 005021      58:    CLR      (R1)+
2258 006702 020127 141000      CMP      R1,#141000
2259 006706 001374      BNE     58
2260
2261
2262
2263
2264
2265 006710 013777 001502 172312      MOV      QDRV,0RKDA
2266 006716 012777 177000 172300      MOV      #=1000,0RKWC
2267 006724 012777 177000 172274      MOV      #177000,0RKBA
2268 006732 012777 000005 172262      MOV      #5,0RKCS
2269
2270 006740 104417      CON,RDY
2271
2272 006742 004737 020112      JSR      PC,CHKCS
2273 006746 104102      ERROR   102
2274
2275
2276
2277
2278
2279
2280

```

```

;AFTER RKBA OVERFLOWED ON DOING
;A TRANSFER OF 2 SECTORS FROM BUS
;ADDRESS (PHYSICAL) STARTING AT 177000
;WORD COUNT
;STARTING BUS ADDRESS
;CHECK IF RKBA INCREMENTED CORRECTLY
;RKBA DID NOT INCREMENT CORRECTLY
;AFTER WRITE IF 2 SECTORS FROM A DATA
;BUFFER STARTING AT (PHYSICAL) BUS
;ADDRESS (177000-200776)
;GET EXPECTED
;DISK ADDRESS
;CHECK IF RKDA INCREMENTED CORRECTLY
;RKDA INCREMENTED WRONGLY AFTER
;A WRITE OF 2 SECTOR (0,1) FROM
;DATA BUFFER STARTING AT BUS
;ADDRESS (PHYSICAL) 177000.
;CHECK IF RKWC OVERFLOWED CORRECTLY
;RKWC DID NOT OVERFLOW TO 0 ON
;DOING A WRITE OF 2 SECTORS FROM
;BA = 177000
;NOW, READ IS DONE OF THE DATA
;THAT WAS WRITTEN TO SEE IF IT
;CAN BE READ CORRECTLY
;LUP TO 48 ON ERROR (SW 9)
;CLEAR THE 1000-WORD DATA
;BUFFER (PA 177000 TO 200776)
;ALL DONE?
;NOW, READ BACK INTO THE
;SAME BUFFER 2 SECTORS
;WRITTEN PREVIOUSLY
;ADDRESS THE DRIVE CYL 0, SEC 0, 0
;READ 2 SECTORS
;INTO THIS BUS ADDRESS
;READ, GO
;WAIT FOR CONTROL RDY
;ANY ERROR IN RKCS?
;ERROR BIT SET IN RKCS ON DOING
;A READ OF 2 SECTORS (0,1), CYL 0,
;SUR 0, INTO DATA BUFFER STARTING
;AT BUS ADDRESS 177000, NOTE AFTER
;177776, RKBA WILL OVERFLOW (0)
;INTO MEX BITS (BIT 4) OF RKCS,
;IF THE ENTIRE TRANSFER (1000 WORD)
;WAS DONE RKBA WILL CONTAIN 1000

```

```

2281
2282
2283 006750 012705 177764      68:    MOV      #=14,R5
2284 006754 012702 137000      MOV      #137000,R2
2285
2286 006760 012701 000001      MOV      #1,R1
2287
2288 006764 020112      78:    CMP      R1,(R2)
2289 006766 001414      BEQ     98
2290 006770 005705      TST     R5
2291 006772 001420      BEQ     98
2292 006774 005205      INC     R5
2293
2294 006776 010137 001162      MOV      R1,$REGO
2295
2296 007002 011237 001164      MOV      (R2),$REG1
2297 007006 104107      ERROR   107
2298
2299
2300
2301
2302 007010 104421 002214      TYPN&G ,MSG13
2303 007014 004737 017774      JSR      PC,TYPDBO
2304
2305
2306 007020 062702 000002      88:    ADD      #2,R2
2307 007024 005201      INC     R1
2308 007026 022701 001001      CMP      #1001,R1
2309 007032 001354      BNE     78
2310
2311
2312 007034 005037 177572      98:    CLR      #SRO

```

```

;AND BIT 4 OF RKCS (MEX) WILL BE SET.
;REPORT ONLY 12 ERRORS.
;STARTING ADDRESS OF BUFFER
;(PA=177000)
;INITIALIZE DATA PATTERN
;CORRECT DATA WORD RECVD?
;REPORT THIS ERROR?
;COUNT ERRORS
;GET EXPECTED DATA WORD
;GET DATA WORD RECVD
;DATA COMPARISON ERROR ON DOING A
;READ OF 2 SECTORS (0,1, CYL 0, SURFACE 0)
;INTO DATA BUFFER (PHYSICAL ADDRESS
;177000 TO 200776)
;TYPE "PHYSICAL BUS ADDRESS"
;TYPE THE 6 DIGIT PHYSICAL BUS ADDRESS
;WHERE THE DATA ERROR OCCURRED.
;INCREMENT POINTER TO BA
;CHECKED THE ENTIRE BUFFER?
; **OFF
;TURN OFF MEMORY MANAGEMENT

```

```

2313 ;*****
2314 ;TEST 6 CHECK DATA TRANSFER FROM 28K TO 32K
2315 ;THIS TEST DOES A WRITE OF 4K WORDS FROM A BUFFER LOCATED AT 28K
2316 ;THEN THE DATA IS READ BACK INTO THE SAME BUFFER(28K-32K) AND IS
2317 ;CHECKED TO SEE IF IT IS CORRECT. NOTE THAT THE BUFFER IS FILLED
2318 ;WITH ALL 1'S BEFORE DOING THE READ,
2319
2320 ;THE WRITE IS DONE STARTING AT CYLINDER 0, SECTOR 0, SURFACE 0,
2321 ;*****
2322 007A40 000004 TST6: SCOPE
2323
2324 007A42 023727 017772 001740 CMP 0LSTBK,#1740 ;32K OR MORE OF MEMORY?
2325 007A50 103002 BHS 10 ;YES
2326 007A52 000137 007374 JNP TST7 ;IF NOT, DONT DO THIS TEST
2327 007A56 012737 001600 172352 18: MOV #1600,0SKIPARE ;MAP 28-32K THRU PAR 5
2328 007A64 012737 000001 177572 MOV #1,00SRO ;TURN ON MEM MANAGEMENT
2329
2330 ;WRITE A COUNT PATTERN (1=10000) INTO DATA BUFFER (PHYSICAL ADDRESS
2331 ;160000=177776). THIS DATA BUFFER WILL BE USED FOR WRITING ON THE DISK,
2332 ;*****
2333 007A72 012737 007100 001110 MOV #28,0LPERR ;LUP TO 28 ON ERROR
2334 007100 104416 CON,RESET
2335 007102 104420 DRV,RESET
2336 007104 012700 000001 MOV #1,R0 ;INITIALIZE DATA PATTERN TO BE WRITTEN
2337 007110 012701 120000 MOV #120000,R1 ;INITIALIZE BA (PA=160000) 28K
2338 007114 010021 38: RO,(R1)+ ;WRITE COUNT PATTERNS (1=10000)
2339 007116 005200 INC R0 ;INTO DATA BUFFER (PA 160000 TO
2340 007120 020027 010001 CNP R0,#10001 ;177776, 28K=32K)
2341 007124 001373 BNE 38
2342
2343 007126 013777 001502 172074 MOV 0DRV,0RKDA ;WRITE ON SEC 0, CYL 0, SUR1
2344 007134 012777 170000 172062 MOV #=10000,0RKWC ;4K WORDS
2345 007142 012777 160000 172056 MOV #160000,0RKBA ;FROM THIS BUS ADDRESS
2346 007150 012777 000003 172044 MOV #3,0RKCS ;WRITE, GO
2347
2348 007156 104417 CON,RDY ;WAIT FOR CONTROL READY
2349
2350 007160 004737 020112 JSR PC,CHKCS ;ANY ERROR IN RKCS?
2351 007164 104102 ERROR 102 ;'ERR' BIT SET IN RKCS ON DOING
2352 ;A WRITE OF 4K WORDS FROM 160000
2353 ;(28K-32K). DISK WRITE BEGAN ON
2354 ;SEC 0, CYL 0, SUR 0, IF ALL 4K
2355 ;WORDS WERE WRITTEN RKBA SHOULD OVERFLOW
2356 ;AND CONTAIN 0. BIT 4 OF RKCS
2357 ;(MEX BIT) SHOULD BE 1
2358
2359 007166 004737 020204 JSR PC,CHKMEX ;CHECK IF RKBA OVERFLOWED AND MEX
2360 ;BITS (4) IN RKCS WAS SET,
2361 007172 104106 ERROR 106 ;MEX BIT 4 NOT SET AFTER OVERFLOW OF
2362 ;RKBA. WRITE OF 4K WORDS (28K=32K) WAS DONE.
2363
2364 ;RETURN HERE IF NO ERROR
2365 007174 012703 010000 MOV #10000,R3 ;WORD COUNT
2366 007100 012704 160000 MOV #160000,R4 ;STARTING BUS ADDRESS
2367 007104 004737 020150 JSR PC,CHKBA ;CHECK IF RKBA INCREMENTED CORRECTLY
2368 007110 104104 ERROR 104 ;RKBA DID NOT OVERFLOW TO AFTER A WRITE IF 4K

```

```

2369 ;WORDS (160000 TO 177776)
2370
2371 007112 004737 020230 JSR PC,CHKWC ;CHECK RKWC OVERFLOWED CORRECTLY
2372 007116 104109 ERROR 106 ;RKWC DID NOT OVERFLOW TO 0
2373 ;AFTER A WRITE OF 4K WORD (160000
2374 ;TO 177776)
2375
2376 ;RETURN ADDRESS FOR LUPING
2377 007220 012737 007226 001110 48: MOV #48,0LPERR
2378 007226 104416 CON,RESET
2379 007230 104420 DRV,RESET
2380
2381 007932 012701 120000 58: MOV #120000,R1 ;CLEAR THE DATA BUFFER BEFORE
2382 007936 005021 CLR (R1)+ ;DOING A READ INTO IT
2383 007940 022701 140000 CNP #140000,R1
2384 007944 001374 BNE 58
2385
2386 007946 013777 001502 171754 MOV 0DRV,0RKDA ;READ FROM SEC 0, SUR 0, CYL 0
2387 007954 012777 170000 171742 MOV #=10000,0RKWC ;4K WORDS
2388 007952 012777 160000 171736 MOV #160000,0RKBA ;INTO THIS BUS ADDRESS
2389
2390 007970 012777 000005 171724 MOV #5,0RKCS ;READ, GO
2391 007276 104417 CON,RDY ;WAIT FOR CTRL RDY
2392
2393 007100 004737 020112 JSR PC,CHKCS ;ANY ERROR IN RKCS?
2394 007104 104102 ERROR 102 ;ERROR BIT SET IN RKCS ON DOING
2395 ;A READ OF 4K WORDS INTO MEMORY
2396 ;(160000=177776)
2397 007106 012705 177764 MOV #=14,R5 ;REPORT 12 ERRORS AT MOST
2398 007112 012702 120000 MOV #120000,R2 ;STARTING ADDRESS OF BUFFER
2399 ;(PA=160000)
2400 007116 012701 000001 MOV #1,R1 ;INITIALIZE DATA PATTERN
2401
2402 007122 020112 68: CMP R1,(R2) ;CORRECT DATA WORD?
2403 007124 001413 BEQ 78
2404 007126 010137 001162 MOV R1,0REG0 ;GET EXPECTD DATA WORD
2405 007132 011237 001164 MOV (R2),0REG1 ;GET DATA WORD RECD
2406 007136 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 007140 104421 002214 TYPMSG #MSG13 ;TYPE "PHYSICAL BUS ADDRESS"
2411 007144 004737 017774 JSR PC,TYPDSC ;TYPE OUT THE PHYSICAL BUS ADDRESS
2412 ;WHERE DATA
2413 007150 005205 INC R5 ;ERROR OCCURRED, R1 CONTAINS
2414 007152 001406 BEQ 68 ;THE VIRTUAL ADDRESS
2415
2416 007154 062702 000002 78: ADD #2,R2 ;POINTER TO NEXT BA
2417 007160 005201 INC R1 ;NEXT PATTERN
2418 007162 022701 001000 CMP #1000,R1 ;ALL DONE?
2419 007166 001355 BNE 68 ;NO
2420
2421 007170 005037 177572 88: CLR 00SRO ;TURN OFF MEM MANAGEMENT

```

```

2422 ;*****
2423 ;*TEST 7 PFRFORM THE LARGEST POSSIBLE DATA TRANSFER
2424 ;THIS TEST PERFORMS THE LARGEST DATA TRANSFER POSSIBLE
2425 ;WITHIN AVAILABLE MEMORY.
2426
2427 ;1) IF KT11 IS PRESENT A WRITE OF 16K WORDS IS DONE ON
2428 ;THE DISK (PROVIDED 20K OR MORE IS PRESENT).
2429
2430 ;2) IF KT11 IS NOT PRESENT THEN ALL BUT TOP 1.5K OF MEMORY WILL BE USED
2431 ;FOR WRITING (RKDP MONITOR).
2432
2433 ;ALL DATA TRANSFERS ARE DONE STARTING AT BA 'PGEND'.
2434 ;ALL WRITES ARE DONE BEGINNING AT SECTOR 0, CYLINDER 0,
2435 ;SURFACE 0.
2436
2437 ;AFTER DOING THE 'WRITE' A 'WRITE CHECK' IS DONE
2438 ;TO SEE IF THE WRITE WAS DONE CORRECTLY. ANY WRITE
2439 ;CHECK ERROR IS REPORTED.
2440 ;*****
2441 TST7: SCOPE
2442 TST 8KT11 ;MEMORY MANAGEMENT PRESENT?
2443 BPL 16 ;NO
2444 ;YES
2445 CMP 8LSTBK,81540 ;20K OR MORE?
2446 BGE XFR16K ;YES
2447
2448 10: MOV MAXBA,R3 ;FIGURE OUT THE MAXIMUM
2449 SUB #PGEND,R3 ;XFER THAT CAN BE DONE
2450 CLC
2451 ROR R3 ;FROM 'PGEND' TO THE
2452 ;'MAXBA'
2453 CLR R1
2454 MOV R3,=(SP) ;PUT DIVIDEND ON STACK (LG)
2455 CLR =(SP) ;(HIGH PART)
2456 MOV #400,=(SP) ;DIVISOR
2457 JSR PC,#8DIV ;GO TO DIVIDE ROUTINE
2458 XFR16K: TST (SP)+ ;POP OFF REMAINDER FROM STACK
2459 BEQ 20
2460 INC R1 ;GET # OF SECTORS REQUIRED TO
2461 ;DO WRITE OF # OF WORDS
2462 20: ADD (SP)+,R1 ;(CONTAINED IN R0). R1 CONTAINS
2463 ;# OF SECTORS
2464 CLR R2 ;FORM THE EXPECTED DISK
2465 SUB #14,R1 ;ADDRESS = AFTER THE WRITE
2466 BMI 40 ;IS DONE,
2467 ADD #20,R2
2468 BR 30
2469 40: ADD #14,R1 ;R2 CONTAINS EXPCD RKDA
2470 BIS R1,R2 ;AFTER WRITE IS DONE
2471 BR XFR
2472
2473 XFR16K: MOV #40000,R3 ;# OF WORDS = 16K
2474 MOV #124,R2 ;RKDA SHOULD INCREMENT TO
2475 ;THIS AFTER A TRANSFER OF 16K
2476 ;WORDS STARTING AT DISK
2477 ;ADDRESS = 0 (CYL 2,SUR 1,SEC 4)
    
```

```

2478
2479 007514 053702 001502 XFR: BIS QDRV,R2 ;ADD THE DRIVE # TO
2480 ;EXPCD RKDA AFTER XFER
2481 007520 012737 007526 001110 MOV #18,8LPERR ;LUP BAK TO '18' ON ERROR
2482 ;(SW 9)
2483 007526 104416 10: CON,RESET
2484 007530 104420 DRV,RESET
2485 007532 013777 001502 171470 MOV QDRV,@RKDA ;ADDRESS THE DRIVE, CYL 0,SUR 0,
2486 ;SEC 0
2487 007540 010377 171460 MOV R3,@RKWC
2488 007544 005477 171454 NEG @RKWC ;WORD COUNT (IF MORE THAN
2489 ;20K IS AVAILABLE A TRANSFER
2490 ;OF 20K WILL BE DONE, IF
2491 ;LESS THAN 20K, 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,@RKBA ;START WRITE FROM HERE
2499 007556 012777 000003 171436 MOV #3,@RKCS ;WRITE, GO
2500
2501 007564 104417 CON,RDY ;WAIT FOR CONTROL READY
2502
2503 007566 004737 020112 JSR PC,CHKCS ;CHECK 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 007900 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 RWKC OVERFLOWED
2518 007620 104105 ERROR 105 ;RWKC DID NOT OVERFLOW TO
2519 ;ZERO AFTER A WRITE
2520
2521 007622 012737 007630 001110 20: MOV #28,8LPERR ;LUP BAK TO '28' ON EROR (SW 9)
2522 007630 104416 CON,RESET
2523 007632 104420 DRV,RESET
2524 007634 013777 001502 171366 30: MOV QDRV,@RKDA ;ADDRESS THE DRIVE, CYL 0,SEC 0,SUR 0
2525 007642 010377 171356 MOV R3,@RKWC
2526 007646 005477 171352 NEG @RKWC
2527 007652 012777 032514 171346 MOV #PGEND,@RKBA ;STARTING BA
2528
2529 007660 012777 000407 171334 40: MOV #407,@RKCS ;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 102
2535
2536 007676 032777 040000 171316      BIT 0BIT14,0RKCS  ;SKIP CHECKING FOR WCE IF HE SET
2537 007704 001030          BNE 76
2538 007706 032777 000001 171304 58:  BIT 0WCE,0RKER  ;WRITE CHECK ERROR?
2539 007714 001406          BEQ 68              ;NO
2540
2541 007716 004737 022032          JSR  PC,GT4RG      ;GET RKCS,ER,DS,DA
2542 007722 017737 171300 001166      MOV  0RKBA,0REG2
2543 007730 104110          ERROR 110
2544
2545
2546
2547
2548
2549
2550
2551 007732 008777 171366          68:  TST  0RKWC
2552 007736 001413          BEQ 76              ;WAS THE ENTIRE DATA BUFFER
2553
2554
2555
2556
2557
2558
2559 007756 042777 000017 171240      BIC  017,0RKWC
2560
2561 007764 000738          BR   48              ;GO DO WRT-CHK FOR
2562
2563
2564
2565
2566 007766 000137 008366          78:  JMP  NXTDRV
  
```

```

2567
2568
2569
2570
2571
2572
2573
2574 007772 104416          EXRCR: CON,RESET
2575 007774 012700 001306          MOV  0KEY,RO
2576 010000 005020          18:  CLR  (RO)+
2577 010002 020027 001552          CMP  RO,0KWHR
2578 010006 001374          BNE 18
2579 010010 105737 001253          TSTB FRSTRT
2580 010014 001045          BNE 38              ;RESTARTED AT 210?
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
  
```

```

,SBTTL EXERCISER PROGRAM
;BEGINNING 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
;UNTILL NOW, DONT CLEAR THEM
;CLEAR LOCATIONS FROM 'HECH'
;TO 'PCMDN' (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 REPETITION 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
  
```



```

2623
2624 010160 012746 177777      MOV    #177777,=(SP)  ;PUT LOW DIVIDEND ON STACK
2625 010164 008046              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 028212      JSR   PC,@#SDIV     ;GO TO THE 'DIVIDE' SUBROUTINE
2629 010176 008726              TST   (SP)+         ;DISCARD THE REMAINDER, QUOTIENT IS
2630                          ;NOW ON TOP OF THE STACK
2631 010900 012637 001820      MOV    (SP)+,DRMAP   ;GET MAPPING FACTOR FOR DRIVES
2632 010904 012737 000804 001822  MOV    #804,CYLMAP   ;GET MAPPING FACTOR FOR CYLINDERS
2633 010912 012737 012527 001824  MOV    #8483,SECMAP  ;GET MAPPING FACTOR FOR SECTORS
2634 010920 012737 031463 001826  MOV    #13107,FNMHAP ;GET MAPPING FACTOR FOR FUNCTION
2635
2636 010926 013700 002084      MOV    MAXBA,R0     ;COMPUTE THE MAXIMUM ALLOWABLE
2637 010932 163700 002082      SUB    BASEBA,R0    ;WORD COUNT FOR DATA TRANSFERS
2638 010936 000241              CLC                    ;
2639 010940 008000              ROR    R0           ;CONVERT TO WORDS
2640
2641 010942 012746 177777      MOV    #177777,=(SP) ;PUT LOW DIVIDEND ON STACK
2642 010946 008046              CLR    =(SP)         ;CLEAR HIGH DIVIDEND AND PUSH
2643                          ;IT ON STACK
2644 010950 010048              MOV    R0,=(SP)     ;PUSH DIVISOR ON STACK
2645 010952 004737 028212      JSR   PC,@#SDIV     ;GO TO THE 'DIVIDE' SUBROUTINE
2646 010956 008726              TST   (SP)+         ;DISCARD THE REMAINDER, QUOTIENT IS
2647                          ;NOW ON TOP OF THE STACK
2648 010960 008216              INC   (SP)          ;
2649 010962 012637 001830      MOV    (SP)+,BAMAP  ;SAVE THE MAPPING FACTOR FOR BUS ADDRESS
  
```

```

2650
2651 ;THE ENTIRE DISK (ALL DRIVES) IS WRITTEN WITH RANDOM PATTERNS. THE FIRST
2652 ;WORD OF EACH SECTOR GIVES THE NUMBER OF WORDS (2'S COMPLEMENT) WRITTEN IN
2653 ;THAT SECTORS. REST OF THE DATA WORDS FOR THE SECTOR ARE GENERATED USING
2654 ;THE ABSOLUTE DISK ADDRESS AS THE RANDOM SEED.
2655 ;IF THE PROGRAM WAS RESTARTED AT 210 THEN CHECK IF SW 4 IS SET. IF IT IS
2656 ;THEN DO NOT REWRITE THE DISK WITH RANDOM PATTERNS. IF SW 4 IS NOT SET THEN
2657 ;ALL THE DISKS (PRESENT) ARE WRITTEN WITH RANDOM PATTERNS.
2658 010966 105737 001253      WRDSEI TSTB FRSTRT ;RESTARTED AT 210?
2659 010972 001407              BEQ   1$           ;NO
2660 010974 105037 001253      CLRB  FRSTRT      ;YES, CLEAR THE RESTART FLAG
2661 010980 032777 000020 170632 BIT    #SW4,#SWR   ;SW 4 SET?
2662 010986 001401              BEQ   1$           ;NO
2663 010990 000540              BR    BEGEX1     ;YES, DONT REWRITE ALL DISKS WITH
2664                          ;RANDOM PATTERNS
2665 010992 012737 010330 001110 18:  MOV    #28,@LPERR  ;LUP TO '28' ON ERROR, SW 9 SET!
2666 010994 012702 001254      MOV    #PDR,R2     ;POINTER TO DRIVE #'S TABLE
2667 010996 013703 001264      MOV    DRVPRS,R3   ;# OF DRIVES PRESENT
2668 010998 012700 011410 28:  MOV    #4872,,R0   ;NUMBER OF SECTORS PER DISK
2669 010999 012201              MOVB  (R2)+,R1     ;GET THE FIRST AVAILABLE DRIVE
2670 010999 042701 177770      BIC   #177770,R1  ;POSITION THE BITS (15,14,13)
2671 010999 010137 001502      MOV    R1,QDRV
2672 010999 000241              CLC                    ;
2673 010999 006001              ROR    R1           ;
2674 010999 006001              ROR    R1           ;
2675 010999 006001              ROR    R1           ;
2676 010999 006001              ROR    R1           ;
2677 010999 010177 170644      MOV    R1,@RKDA   ;BASE DISK ADDRESS
2678
2679 010999 013704 002054      MOV    MAXBA,R4     ;CALCULATE MAXIMUM BUFFER
2680 010999 163704 002052      SUB    BASEBA,R4   ;
2681 010999 006204              ASR    R4           ;CONVERT TO WORDS
2682 010999 042704 000377      BIC   #377,R4     ;KEEP ONLY WHOLE BLOCKS
2683 010999 000304              SWAB  R4           ;
2684 010999 020427 000014      CMP    R4,#12     ;MAX OF 12 SECTORS
2685 010999 003402              BLE   3$           ;
2686 010999 012704 000014      MOV    #12,,R4
2687
2688 010999 010401 38:  MOV    R4,R1
2689 010999 020400              CMP    R4,R0
2690 010999 003401              BLE   4$           ;
2691 010999 010001              MOV    R0,R1
2692 010999 000301 48:  SWAB  R1
2693 010999 005401              NEG   R1
2694 010999 010137 010446      MOV    R1,#8
2695 010999 010177 170562      MOV    R1,@RKWC
2696 010442 004837 016704      JSR   R5,GENBUF   ;GENERATE RANDOM DATA BUFR
2697 010446 000000 58:  ,WORD 0
2698 010450 032514              ,WORD DBUF        ;STARTING ADDRESS OF DATA BUFR
2699
2700 010452 012777 032514 170546  MOV    #DBUF,@RKBA ;FROM THIS BUS ADDRESS
2701 010460 012777 000003 170534  MOV    #3,@RKCS   ;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
  
```

```

2706 ;ON THE DISK, YOU ARE ADVISED TO USE
2707 ;BASIC AND DYNAMIC TESTS.
2708
2709 010476 032777 000400 170434 BIT %BIT6,%BWR ;TYPE CURRENT STATUS
2710 010404 001438 BEQ 88
2711 010406 032700 000377 BIT %377,R0
2712 010412 001032 BNE 88
2713 010414 104401 010922 TYPE ,656 ;;TYPE ASCII STRING
2714 010420 000421 BR 648 ;;GET OVER THE ASCII
;1658: ;ASCII <18><12>/WRITTING RANDOM PATTERN, DRIVE /
648:
2715 010464 MOV QDRV,=(SP)
2717 010464 013746 001502 TYPE ,BYTE 1
2718 010470 104403 ,BYTE 0
2719 010472 001
2720 010473 000
2721
2722 010474 104401 001313 TYPE ,%CRLF
2723 010600 104407 681 CKSWR ;CHECK FOR SOFTWARE SWR CHANGE
2724 010602 160400 SUB R4,R0 ;DECREMENT SECTOR COUNT
2725 010604 003304 BGT 36
2726
2727 010606 008303 DEC R3 ;MORE DRIVES TO DO?
2728 010610 001247 BNE 26
2729
;IF SW 3 IS SET, ENABLE THE LINE CLOCK AND INITIALISE COUNTS.
2730
2731
2732 010412 032777 000010 170320 BEGRX1: BIT %SW3,%BWR ;KNILL CLOCK PRESENT?
2733 010420 001403 BEQ BEGRX ;IF NOT, SKIP, NOTE:IF KNILL IS
;NOT PRESENT SW3 SHOULD NOT BE SET
;OTHERWISE, A TIMEOUT WILL OCCUR,
2734
2735 010622 052777 000100 170404 B18 %BIT6,%KNWLS ;ENABLE KNILL CLOCK
2736 ;SERVICED AT PRIORITY 7 (KNBRVE)
2737

```

```

2738 ;THE PROGRAM IS GOING TO LOOP BACK TO THIS POINT AT THE END OF A PASS.
2739
2740 010630 104416 BEGRX: 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 JBR PC,%GENSRQ ;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 'GENSRQ' 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 PPRVL,=(SP) ;NEW P8W, RAISE PRIORITY
2754 010654 012746 001062 MOV %18,=(SP) ;RETURN PC *****
2755 010660 000002 RTI
2756
2757 010662 005737 001264 18: TST DRVPRS ;ANY DRIVES IN SYSTEM?
2758 010666 001040 BNE 26
2759 010670 104401 010676 TYPE ,656 ;;TYPE ASCII STRING
2760 010674 000432 BR 648 ;;GET OVER THE ASCII
;1658: ;ASCII <18><12>/HALTING = PRESS CONTINUE TO RESTART AT '200'/'<18><12><12><12>
648:
2761
2762 010762 HALT
2763 010762 000000 JMP START
2764 010764 000137 003376
2765
2766 010770 012700 001306 28: MOV %KEY,R0
2767 010774 005720 TST (R0)+ ;ANY UNFINISHED COMMAND
2768 010776 100004 BPL 48 ;IN Q
2769 011000 020027 001326 CMP R0,%KEY+20
2770 011004 001373 BNE 36
2771 011006 004737 014500 JBR PC,%GENSRQ ;IF NOT, GO GENERATE 8 MORE COMMANDS
2772 011012 000460 BR CHFAPN
2773
2774
2775 011014 012700 001306 48: MOV %KEY,R0
2776 011020 032710 010000 58: BIT %BIT12,(R0) ;ANY HIGH PRIORITY COMMAND IN Q
2777 011024 001404 BEQ 68
2778 011026 008710 TST (R0) ;FINISHED?
2779 011030 100402 BNE 68 ;YES
2780 011032 108710 TSTB (R0) ;IN EXECUTION?
2781 011034 100168 BPL PRICMND ;NO, GO PROCESS HIGH PRIORITY
2782
2783 011036 005720 68: TST (R0)+ ;CHECK THE ENTIRE Q
2784
2784 011040 020027 001326 CMP R0,%KEY+20
2785 011044 001368 BNE 68
2786
2787 ;FIND OUT IF THERE IS A WRITE CHECK FUNCTION TO BE DONE IMMEDIATELY AFTER
2788 ;A WRITE, THAT WAS DONE PREVIOUSLY.
2789
2790
2791 011046 005737 001456 TST WCFLG ;IS WRITE CHECK TO FOLLOW
2792 011052 100040 BPL CHFAPN ;WRITE? IF NOT, BRANCH
2793 ;YES

```

```

2794 011A54 013700 001456      MOV      WCFLG,R0      ;GET WC FLAG
2795 011A60 042700 177400      BIC      #177400,R0   ;LOWER BYTE CONTAINS KEY:X2 (OFFSET FROM
2796                                     ;KEY), OF THE WRITE FUNCTION
2797 011A64 016001 002032      MOV      PCMND(R0),R1 ;GET POINTER
2798 011A70 062700 001306      ADD      #KEY,R0      ;FROM ADDRESS OF THE KEY
2799                                     ;WHICH WAS USED FOR PREVIOUS
2800                                     ;WRITE
2801 011A74 022761 000002 000002    CMP      #2,(R1)      ;CHECK THAT THE FUNCTION
2802 011102 001413              BEQ      78           ;WAS A WRITE
2803 011104 011037 001162      MOV      (R0),#REG0   ;GET KEY CONTENTS
2804 011110 016137 000002 001164    MOV      2(R1),#REG1  ;GET THE FUNCTION INDICATED BY THIS KEY
2805 011116 104027              ERROR    27           ;IF NOT, ERROR
2806                                     ;BEFORE DOING A WRITE CHECK A WRITE IS
2807                                     ;ALWAYS DONE, OCCURANCE OF THIS ERROR
2808                                     ;INDICATES THAT SOMEHOW AN ATTEMPT IS BEING
2809                                     ;MADE TO DO WRITE CHECK BEFORE A WRITE IS
2810                                     ;PERFORMED, THE KEY IN ERROR MESSAGE WAS
2811                                     ;THE ONE WHICH GAVE RISE TO THIS ATTEMPT.
2812                                     ;THE FUNCTION CODE IS THE FUNCTION ASSOCIATED
2813                                     ;WITH THAT KEY
2814 011120 005037 001456      CLR      WCFLG        ;ABORT THIS WRITE CHECK
2815 011124 052710 100000      BIS      #BIT15,(R0)
2816 011130 000647              BR       QMNGER
2817 011132 012761 000006 000002 78:  MOV      #6,2(R1)    ;SET UP BITS FOR WRT CHK
2818                                     ;NOW
2819 011140 042710 174340      BIC      #174340,(R0) ;CLEAR OUT THE UNNECESSARY
2820                                     ;FLAGS FROM THE KEY
2821 011144 052710 000100      BIS      #BIT6,(R0)  ;SET FLAG FOR WRITE CHECK, FOR
2822                                     ;THIS KEY
2823 011150 000137 011612      JMP      EXECUT
2824
2825                                     ;NO HIGH PRIORITY COMMAND IN Q
    
```

```

2826                                     ;NO HIGH PRIORITY COMMAND WAS FOUND IN THE Q, FIND OUT THE FIRST AVAILABLE
2827                                     ;COMMAND IN THE Q, FOR EXECUTION.
2828 011154 005000      CHFAFN: CLR      R0      ;WAIT FOR ANY IMMEDIATE INTERRUPT
2829 011156 005046      CLR      -(SP)        ;NEW PSW
2830 011160 012746 011166      MOV      #16,=(SP)   ;RETURN FOR RTI
2831 011164 000002      RTI
2832
2833
2834 011166 105200      18:  INCB     R0
2835 011170 001376      BNE     #2
2836 011172 013746 001244      MOV     PPRVL,=(SP)  ;NEW PSW, LOCK OUT CPU
2837 011176 012746 011204      MOV     #28,=(SP)   ;RETURN FOR RTI *****
2838 011902 000002      RTI
2839
2840
2841 011904 012700 001306      28:  MOV     #KEY,R0
2842 011910 005710      CH1:  TST     (R0)        ;UNFINISHED COMMAND?
2843 011912 100436      BMI     CH2
2844 011914 105710      TSTB   (R0)        ;IN PROGRESS?
2845 011916 100434      BMI     CH2
2846 011920 011001      MOV     (R0),R1
2847 011922 042701 177770      BIC     #177770,R1
2848 011926 105761 001426      TSTB   BUSY(R1)    ;IS THIS DRIVE BUSY?
2849 011932 100426      BMI     CH2
2850 011934 105761 001436      TSTB   POS(R1)     ;HAS THIS DRIVE BEEN POSITIONED
2851                                     ;FOR ANY OTHER COMMAND, IF YES,
2852                                     ;SKIP, IF NOT, PROCEED
2853 011940 001023      BNE     CH2
2854
2855
2856 011942 010002      MOV     R0,R2
2857
2858 011944 000414      BR      28
2859                                     ;CHECK IF THERE IS ANY OTHER COMMAND
2860                                     ;ON A DRIVE THAT IS NOT THE SAME
2861                                     ;AS THE PREVIOUS DRIVE, THIS COMMAND
2862                                     ;SHOULD NOT BE IN PROGRESS
2863                                     ;AND SHOULD NOT HAVE BEEN COMPLETED
2864 011946 005712      18:  TST     (R2)
2865 011950 100412      BMI     28
2866 011952 105712      TSTB   (R2)        ;IN PROGRESS?
2867 011954 100410      BMI     28
2868 011956 011203      MOV     (R2),R3
2869 011960 042703 177770      BIC     #177770,R3
2870 011964 105763 001426      TSTB   BUSY(R3)    ;IS THE DRIVE BUSY?
2871 011970 100402      BMI     28
2872 011972 020301      CMP     R3,R1
2873 011974 001447      BEQ     POSITION
2874                                     ;IS THIS COMMAND ON THE SAME DRIVE?
2875                                     ;IF YES, GO AND POSITION THE COMMAND
2876                                     ;POINTED TO BY R0, (BECAUSE THERE IS
2877                                     ;ONE MORE COMMAND THAT CAN BE PERFORMED
2878                                     ;ON THE SAME DRIVE)
2879                                     ;CHECK REST OF THE Q
2880 011976 005722      28:  TST     (R2)+
2881 011980 020227 001326      CMP     R2,#KEY+20
2882 011984 001360      BNE     18
2883
2884 011986 000470      BR      EXNSK
2885
2886 011988 000470      BR      EXNSK
2887
2888 011990 005720      CH2:  TST     (R0)+
2889 011992 020027 001326      CMP     R0,#KEY+20
    
```

```

2882 011916 001334          BNE      CH1
2883
2884
2885          ;NO (UNPOSITIONED) EXECUTABLE COMMAND WAS FOUND IN THE Q. CHECK IF THERE
2886          ;IS ANY POSITIONED COMMAND IN THE Q, WAITING TO BE EXECUTED.
2887
2888 011920 105777 167676      TSTB   @RKCS          ;IF THE CONTROLLER IS BUSY GO TO
2889 011924 100402              BMI     18             ;STATUS AND WAIT FOR INTERRUPTS
2890 011926 000137 021436      JMP     @STATUS       ;IF THE CONTROLLER IS NOT BUSY FIND
2891
2892 011932 012700 001306      18:    MOV     @KEY,RO   ;ANY POSITIONED COMMAND AND EXECUTE
2893 011936 032710 000020      28:    BIT     @BIT4,(RO)
2894 011942 001414              BEQ     38             ;IT
2895 011944 005710              TST    (RO)           ;MAKE SURE COMMAND IS POSITIONED
2896 011946 100412              BMI     34             ;
2897 011950 105710              TSTB   (RO)           ;COMMAND IS UNFINISHED,
2898 011952 100410              BMI     38             ;IT IS NOT IN PROGRESS,
2899 011954 011001              MOV     (RO),R1       ;THE DRIVE IS NOT IN BUSY
2900 011956 042701 177770      BIC    @177770,R1
2901 011962 105761 001426      TSTB   BUSY(R1)
2902 011966 100402              BMI     38             ;
2903 011970 000137 011612      JMP     EXECUT        ;GO EXECUTE THE COMMAND IF THE
2904
2905 011974 005720              38:    TST    (RO)+     ;ABOVE CONDITIONS ARE SATISFIED
2906 011976 020027 001326      CMP     RO,@KEY+20    ;CHECK ALL COMMANDS IN Q
2907 011982 001355              BNE     28             ;
2908 011984 000137 021436      JMP     @STATUS
2909
2910
2911          ;ENTER HERE IF THERE IS A PRIORITY COMMAND TO BE EXECUTED.
2912
2913
2914
2915 011910 000137 011612      PRICMN: JMP     EXECUT ;GO EXECUTE NON-SEEK COMMAND.
2916
2917          ;ENTER THIS IF A COMMAND IS TO BE PREPOSITIONED (BEFORE EXECUTING A
2918          ;FUNCTION)
2919
2920 011914 032710 000020      POSITION: BIT @BIT4,(RO) ;ALREADY POSITIONED?
2921 011920 001333              BNE     CH2           ;YES, GO BACK AND CHECK IF REST OF THE
2922                          ;COMMANDS IN THE Q ARE TO BE POSITIONED
2923
2924 011922 004737 012164      JSR    PC,@OSK       ;GO CHECK, & SET UP FLAGS
2925 011926 004737 020350      JSR    PC,@OSCHND    ;SAVE INFO ABOUT PRESENT & PAST COMAND
2926 011932 012777 000111 167562  MOV     @11,@RKCS    ;POSITION THE COMMAND
2927 011940 005046              CLR    @SP            ;NEW PSM, DROP PRIORITY
2928 011942 012766 011450      MOV     @18,@SP      ;RETURN FOR RTI
2929 011946 000002              RTI
2930
2931 011950 105737 001535      18:    TSTB   INTIFL       ;WAIT FOR INTERRUPT
2932 011954 001775              BEQ     =4            ;
2933 011956 012777 013256 167584  MOV     @INTHND,@RKVEC ;RE-ESTABLISH RK INTERRUPT VECTOR
2934 011964 000137 011154      JMP     @CHAFN       ;GO BACK AND CHECK IF ANY COMMANDS TO BE
2935                          ;POSITIONED OR EXECUTED
2936
2937
    
```

```

2938          ;IS THERE ANY POSITIONED COMMAND IN
2939          ;THE Q. FIND OUT? IF THERE IS
2940          ;ONE EXECUTE THE POSITIONED COMMAND,
2941          ;BUT FIRST POSITION THE COMMAND (KEY)
2942          ;POINTED TO BY RO
2943
2944 011970 012701 001306      EXNRK: MOV     @KEY,R1
2945 011974 032711 000020      18:    BIT     @BIT4,(R1) ;HEADS POSITIONED?
2946 011980 001412              BEQ     28             ;
2947 011982 005711              TST    (R1)           ;FUNCTION COMPLETED?
2948 011984 100410              BMI     28             ;YES, BRANCH
2949 011986 105711              TSTB   (R1)           ;FUNCTION IN PROGRESS?
2950 011990 100406              BMI     28             ;
2951 011992 011103              MOV     (R1),R2
2952 011994 042703 177770      BIC    @177770,R2
2953 011996 105762 001426      TSTB   BUSY(R2)
2954 011998 100005              BPL     38             ;DRIVE BUSY?
2955
2956 011926 005721              28:    TST    (R1)+     ;CHECK ALL COMMANDS IN Q
2957 011930 020127 001326      CMP     R1,@KEY+20
2958 011934 001387              BNE     18             ;
2959
2960 011936 000425              BR     EXECUT        ;NO POSITIONED COMMAND WAS
2961                          ;FOUND IN THE Q. SO EXECUTE
2962                          ;COMMAND POINTED TO BY RO
2963
2964
2965
2966          ;AN ALREADY POSITIONED COMMAND HAS
2967          ;BEEN FOUND.
2968 011940 010137 001536      38:    MOV     R1,@AVKEY   ;SAVE POINTER TO THE COMMAND(KEY)
2969 011944 004737 012164      JSR    PC,@OSK       ;WHICH IS ALREADY POSITIONED
2970                          ;GO AND POSITION THE COMMAND(KEY)
2971                          ;POINTED TO BY RO
2972 011950 004737 020350      JSR    PC,@OSCHND    ;SAVE INFO ABOUT PAST & PRESENT COMAND
2973 011954 012777 000111 167440  MOV     @11,@RKCS    ;SEEK, IDE, GO
2974 011962 005046              CLR    @SP            ;ALLOW FIRST INTERRUPT
2975 011964 012766 011972      MOV     @48,@SP      ;RETURN FOR RTI *****
2976 011970 000002              RTI
2977
2978 011972 105737 001535      48:    TSTB   INTIFL       ;DID INTERRUPT OCCUR?
2979 011976 001775              BEQ     48             ;BR IF NOT
2980 011600 012777 013256 167432  MOV     @INTHND,@RKVEC ;REESTABLISH INTERRUPT ADDRESS
2981 011606 013700 001536      MOV     @AVKEY,RO    ;RESTORE THE SAVED POINTER TO KEY
    
```

```

2982                                     ;EXECUTE THE COMMAND POINTED TO BY R0
2983                                     ;R0 CONTAINS THE POINTER TO THE
2984                                     ;COMMAND KEY
2985 011412 011001          EXECUT: MOV    (R0),R1          ;GET DRIVE NO
2986 011414 042701 177770          BIC    #177770,R1
2987 011420 003710          T&T    (R0)
2988 011422 100004          BPL    18
2989 011424 011037 001162          MOV    (R0),@REGO
2990 011430 104030          ERROR   30
2991                                     ;GET KEY
2992                                     ;IF NOT, ERROR
2993                                     ;AN ATTEMPT WAS MADE TO REEXECUTE A COMMAND
2994                                     ;THAT HAS ALREADY BEEN EXECUTED BEFORE,
2995                                     ;(ON DRIVE NO, GIVEN IN ERROR MESSAGE)
2996 011432 000512          BR     ABRT1
2997 011434 105710 181          TSTB  (R0)
2998 011436 100004          BPL    38
2999 011440 011037 001162          MOV    (R0),@REGO
3000 011444 104030          ERROR   30
3001                                     ;GET KEY
3002                                     ;IF YES, ERROR
3003                                     ;AN ATTEMPT WAS MADE TO REEXECUTE A COMMAND
3004                                     ;THAT IS IN PROGRESS (ON DRIVE NO, GIVEN
3005                                     ;IN ERROR MESSAGE)
3006 011446 000504          BR     ABRT1
3007 011450 105781 001426          TSTB  BUSY(R1)
3008 011454 100004          BPL    48
3009 011456 010137 001162          MOV    R1,@REGO
3010 011460 104002          ERROR   2
3011 011464 000470          BR     ABRT
3012                                     ;GET DRIVE #
3013                                     ;'BUSY' FLAG FOR THE DRIVE (CONTAINED
3014                                     ;IN R1) IS SET, INDICATING THAT THE DRIVE
3015                                     ;IS 'BUSY' AND ENGAGED IN AN ACTIVITY;
3016                                     ;STILL AN ATTEMPT WAS MADE TO INITIATE
3017                                     ;A FUNCTION ON THIS DRIVE. THIS IS AN
3018                                     ;UNEXPECTED ERROR CONDITION.
3019                                     ;CONTROL READY SET?
3020                                     ;YES
3021 011666 105777 167330          TSTB  @RKS5
3022 011672 100404          BMI    58
3023 011674 004737 022032          JSR   PC,GT4RG
3024 011700 104003          ERROR   3
3025 011702 000461          BR     ABRT
3026                                     ;CONTROL READY IS NOT SET. THIS IS AN
3027                                     ;UNEXPECTED ERROR CONDITION AT THIS
3028                                     ;POINT OF EXECUTION, CONTROL SHOULD
3029                                     ;BE NORMALLY READY.
3030                                     ;GET THE FIRST ITEM FROM LIST
3031                                     ;DISK ADDRESS
3032                                     ;R/W/S RDY SET?
3033                                     ;YES
3034 011704 011002          MOV    (R0),R2
3035 011706 000302          SWAB  R2
3036 011710 042702 177770          BIC    #177770,R2
3037 011714 006302          ASL   R2
3038 011716 010204          MOV    R2,R4
3039 011720 016205 002032          MOV    PCHND(R2),R5
3040                                     ;FORM THE ADDRESS OF THE
3041                                     ;PARAMETER LIST FOR THIS
3042                                     ;COMMAND
3043 011724 011577 167300          MOV    (R5),@RKDA
3044                                     ;GET R/W/S RDY SET?
3045                                     ;YES
3046 011730 032777 000100 167260          BIT   @RWS,@RKDS
3047 011736 001006          BNE   68
3048 011740 010137 001250          MOV    R1,@RDV
3049 011744 004737 022032          JSR   PC,GT4RG
3050 011750 104004          ERROR   4
3051 011752 000435          BR     ABRT
3052                                     ;R/W/S RDY IS NOT SET FOR THE DRIVE.
3053                                     ;A (NON-SEEK) FUNCTION WAS SUPPOSED
3054                                     ;TO BE EXECUTED ON THIS DRIVE. THIS IS
3055                                     ;AN UNEXPECTED ERROR CONDITIONS AT THIS
    
```

```

3038                                     ;POINT OF EXECUTION R/W/S RDY SHOULD
3039                                     ;BE NORMALLY SET.
3040 011754 016503 000002          68:  MOV    2(R5),R3
3041                                     ;GET FUNCTION TO BE DONE
3042 011760 122703 000002          CMPB  #2,R3
3043 011764 001437          BEQ   WRFNC
3044                                     ;IS IT A WRITE FUNCTION?
3045                                     ;YES, IT IS WRITE
3046                                     ;IT'S NOT A WRITE FUNCTION
3047 011766 016503 000002          NWRFNC: MOV 2(R5),R3
3048 011772 052703 000501          BIS   #501,R3
3049 011776 016577 000004 167220          MOV   4(R5),@RKC
3050 012004 016577 000006 167214          MOV   6(R5),@RKBA
3051                                     ;GET FUNCTION TO BE DONE
3052                                     ;SET SSE,IDE,GO BITS
3053                                     ;GET WORD COUNT
3054                                     ;GET BUS ADDRESS
3055 012012 004737 020372          JSR   PC,FNCMND
3056                                     ;SAVE INFO ABOUT PAST & PRESENT COMAND
3057 012016 010377 167200          MOV   R3,@RKS5
3058                                     ;SET SSE,IDE, FUNCTION, GO
3059 012022 052710 000200          BIS   #BIT7,(R0)
3060 012026 052704 000200          BIS   #BIT7,R4
3061                                     ;SET FLAG INDICATING FUNCTION
3062                                     ;IN PROGRESS
3063                                     ;R4 CONTAINS OFFSET TO THE COMMAND KEY
3064                                     ;(FROM 'KEY') BITS 0-3, BIT 7 IS SET
3065 012032 110461 001426          MOVB  R4,BUSY(R1)
3066 012036 110437 001534          MOVB  R4,INTFLG
3067 012042 000137 021436          JMP   STATUS
3068                                     ;SET FLAG INDICATING DRIVE BUSY
3069                                     ;SET FLAG FOR INTERRUPT USE
3070 012046 004737 014426          ABRT: JSR  PC,DRVABT
3071 012052 004737 016540          JSR   PC,CHKDRV
3072 012056 000240          NOP
3073 012060 000137 010650          ABRT1: JMP  QMNGER
    
```

```

3067
3068
3069 012064 016537 000004 012104 WRFNC1 MOV 4(RS),18 ;THE FUNCTION TO BE EXECUTED IS A
;WRITE FUNCTION
3070 012072 016537 000006 012106 MOV 6(RS),28 ;GET # OF WORDS TO BE WRITTEN
;GET STARTING BA OF BUFFER
3071 012100 004837 016704 JBR R6,GENBUF ;GO GENERATE BUFFER
3072 012104 000000 181 ;WORD 0 ;SAVE # OF WORDS
3073 012106 000000 281 ;WORD 0 ;SAVE STARTING BUS ADDRESS OF BUFFER
3074 012110 052703 000101 BIS #101,R3 ;SET IDE AND GO BIT
3075 012114 013777 012104 MOV 18,0RKWC ;SET WORD COUNT
3076 012122 013777 012106 167102 MOV 28,0RKBA ;SET BUS ADDRESS
3077 012130 004737 020372 JBR PC,FNCMND ;SAVE INFO ABOUT PAST & PRESENT COMAND
3078
3079 012134 010377 167062 MOV R3,0RKCS ;ISSUE WRITE,IDE,GO
3080 012140 052710 000200 BIS 0BIT7,(R0) ;SET FLAG INDICATING FUNCTION IN
3081 012144 052704 000200 BIS 0BIT7,R4 ;PROGRESS
3082
3083 012150 110461 001426 MOV R4,BUSY(R1) ;SET FLAG INDICATING DRIVE BUSY
3084 012154 110437 001924 MOV R4,INTFLG ;SET FLAG FOR INTERRUPT USE
3085
3086
3087 012160 000137 021436 JMP STATUS ;R4 CONTAINS OFFSET TO THE COMMAND KEY
;(FROM 'KEY') BITS 0-3, BIT 7 IS SET
    
```

```

3088 012164 011001 POSK1 MOV (R0),R1 ;INITIAL CHECKING PRIOR TO DOING
3089 012166 042701 177770 BIC 0177770,R1 ;POSITIONING COMMAND POINTED TO BY R0
3090 012172 105761 001426 TSTB BUSY(R1) ;DRIVE BUSY?
3091 012176 100004 BPL 18
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 78 ;IN R1) IS SET, INDICATING THAT THE DRIVE
;IS 'BUSY' AND ENGAGED IN AN ACTIVITY,
;STILL AN ATTEMPT WAS MADE TO INITIATE
;POSITIONING ON THIS DRIVE. THIS IS AN
;UNEXCEPTED ERROR CONDITION.
3095
3096
3097
3098
3099 012210 105777 167006 181 TSTB 0RKCS ;CONTROL READY SET?
3100 012214 100404 BMI 28 ;YES
3101 012216 004737 022032 JBR PC,GT4RG ;GET RKCS, ER, DS, DA
3102 012222 104003 ERROR 3 ;CONTROL READY IS NOT SET. THIS IS AN
3103 012224 000454 BR 68 ;UNEXPECTED ERROR CONDITION AT THIS
;POINT OF EXECUTION, CONTROL SHOULD
;BE NORMALLY READY.
3104
3105
3106 012226 011002 281 MOV (R0),R2
3107 012230 000302 SWAB R2
3108 012232 042702 177770 BIC 0177770,R2
3109 012236 006302 AND R2
3110 012240 016203 002032 MOV PC,MND(R2),R3 ;STARTING WORD OF COMMAND TABLE
3111 012244 011377 166760 MOV (R1),0RKDA
3112 012250 032777 000100 166740 BIT 0RWS,0RKDS ;R/W/S RDY SET?
3113 012256 001006 BNE 38 ;YES
3114 012260 010137 001250 MOV R1,SRDRV ;GET DRIVE #, FOR TYPING SERIAL #
3115 012264 004737 022032 JBR 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 68 ;(POSITIONING) SEEK WAS SUPPOSED TO BE
;BE EXECUTED ON THIS DRIVE. THIS IS
;AN UNEXPECTED ERROR CONDITION. AT THIS
;POINT OF EXECUTION R/W/S RDY SHOULD
;BE NORMALLY SET.
3118
3119
3120
3121
3122 012274 110261 001426 381 MOV R2,BUSY(R1) ;SET FLAG INDICATING DRIVE BUSY
3123 012200 152761 000200 001426 BIS 0BIT7,BUSY(R1)
3124 012206 032710 000010 BIT 0BIT3,(R0) ;IS THIS A SEEK COMMAND
3125 012212 001006 BNE 48
3126 012214 112761 000377 001436 MOV 0377,POS(R1) ;SET FLAG INDICATING, THIS DRIVE POSITIONS
3127 012222 052710 000020 BIS 0BIT4,(R0) ;SET FLAG INDICATING THIS COMMAND
3128 012226 000402 BR 58
3129 012230 052710 000200 481 BIS 0BIT7,(R0) ;POSITIONED. SET FLAG INDICATING
;FUNCTION IN PROGRESS
3130
3131 012234 012777 012376 166676 581 MOV 0INT18K,0RKVEC ;SET UP RK11 VECTOR
3132 012242 013777 001244 166672 MOV 0PRLVL,0RKSTAT ;PSW ON INTERRUPT
3133 012250 105037 001535 CLRB INT1FL ;CLEAR FLAG INDICATING - INTERRUPT
3134
3135
3136 012256 004737 014426 681 RTS PC ;OCCURRED
3137 012262 004737 016940 JBR PC,DRVABT ;ABORT THE FUNCTION
3138 012266 000240 JBR PC,CHKDRV ;GO CHECK. DRIVE ERRORS
3139 012270 005726 NOP
3140 012272 000137 010680 781 TST (SP)+ ;POP THE RETURN ADDRESS
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 INTISK1 INCB INT1FL ;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 BIT SET?
3154 012424 001403 BEQ 18
3155 012426 004737 022222 JSR PC,RG48DR ;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 18: MOV #RKCS,R0 ;CHECK THERE IS A SEEK FUNCTION
3167 012440 042700 177760 BIC #177760,R0 ;IN RKCS
3168 012444 022700 000010 CMP #10,R0
3169 012450 001403 BEQ 28
3170 012452 004737 022032 JSR PC,GTARG ;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 168544 28: MOV #RKDA,R0 ;GET THE DRIVE # FROM RKDA
3178 012464 042700 017777 BIC #17777,R0
3179 012470 000241 CLC
3180 012472 006100 ROL R0
3181 012474 006100 ROL R0
3182 012476 006100 ROL R0
3183 012480 006100 ROL R0
3184 012502 010001 MOV R0,R1
3185 012404 105761 001426 TSTB BUSY(R1) ;CHECK THIS DRIVE NO. WAS BUSY
3186 012410 100403 BMI 38 ;OK, IF IT WAS
3187 012512 010137 001162 MOV R1,#REGO ;GET DRIVE # (FROM RKDA)
3188 012416 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 38: MOVB BUSY(R1),R0 ;FORM THE ADDRESS OF
3195 012524 042700 177600 BIC #177600,R0 ;THIS KEY
3196 012530 062700 001306 ADD #KEY,R0
    
```

```

3197
3198 012534 032710 000020 BIT #BIT4,(R0) ;IS THE KEY ACTIVE FOR 'POSITIONING'?
3199 012540 001003 BNE 48
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 ERROR MESSAGE)
3204
3205 012550 105761 001436 48: TSTB POS(R1) ;IS THE 'POSITIONING' FLAG SET?
3206 012554 001003 BNE 58
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. OCCURRENCE OF THIS
3213 ;ERROR INDICATED THAT THE DRIVE #
3214 ;(FROM RKDA)GIVING THIS INTERRUPT DID
3215 ;NOT HAVE ITS POSITIONING FLAG SET.
3216
3217 012564 008777 168432 88: TST #RKCS ;ANY ERROR?
3218 012570 100044 BPL 88 ;NO - RETURN
3219
3220 012572 032737 000040 001474 BIT #BITS,ERCODE ;SAME ERROR
3221 012600 001012 BNE 68
3222 012602 042737 000040 001474 BIC #BITS,ERCODE
3223 012610 001026 BNE 78
3224 012612 052737 000040 001474 BIS #BITS,ERCODE
3225
3226 012620 004737 022222 JSR PC,RG48DR ;GET RKCS, ER, DS, DA AND DRIVE # FOR
3227 ;TYPING SERIAL DRIVE #
3228 012624 104011 ERROR 11 ;BIT 18, 'ERR' SET IN RKKCS AFTER FIRST
3229 ;INTERRUPT - WHICH OCCURRED AFTER A
3230 ;POSITIONING SEEK WAS INITIATED. RKER
3231 ;WILL CONTAIN ERROR BIT/8
3232 012626 042710 000020 68: 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 88 ;NO
3241
3242 012666 004737 014426 78: JSR PC,DRVABT ;ABORT THE FUNCTION
3243 012672 005061 001446 CLR RETRY(R1)
3244 012676 005037 001474 CLR EPCODE
3245
3246 012702 000002 88: RTI
    
```

```

3247                                     ;THIS ROUTINE IS ENTERED UPON COMPLETION
3248                                     ;OF A SEEK OPERATION UNDER INTERRUPT MODE
3249
3250 012704 004737 020272          SKMP: JSR      PC,CHKCRDY      ;CONTROL READY SET?
3251 012710 104013                                     ;CONTROL RDY NOT SET, AFTER A SEEK
3252                                     ;DONE INTERRUPT!
3253
3254 012712 017746 166300          18:  MOV      @RKDS,-(SP)      ;GET DRIVE # THAT INTERRUPTED
3255 012716 004737 022170          JSR      PC,CROTFL      ;ROTATE BITS 15,14,13 TO POSITION
3256                                     ;0,1,2
3257 012722 012601                                     ;GET THE ROTATED BITS (DRIVE ID BITS)
3258 012724 105761 001426          MOV      (SP)+,R1      ;CHECK THAT DRIVE WAS BUSY
3259 012730 100403                                     ;
3260 012732 004737 022032          BHI     PC,GT4RG      ;
3261 012736 104014          JSR      PC,GT4RG      ;
3262          ERROR      14                                     ;'BUSY' FLAG FOR THE INTERRUPTING DRIVE
3263                                     ;(RKDS) WAS NOT SET, DRIVE # (15,14,13 - RKDS)
3264                                     ;INTERRUPTED AFTER SEEK WAS COMPLETE, BUT
3265                                     ;'BUSY' FLAG CORRESPONDING TO THAT DRIVE
3266                                     ;WAS CLEAR, IF THE CORRECT DRIVE WAS
3267                                     ;INTERRUPTING 'BUSY' FLAG SHOULD HAVE
3268 012740 116100 001426          28:  MOVBS  BUSY(R1),RO      ;BEEN SET,
3269 012744 042700 177600          BIC     #17800,RO      ;FORM THE ADDRESS OF THE
3270 012750 062700 001308          ADD     #KEY,RO        ;KEY
3271 012754 032710 000020          BIT     %BIT4,(RO)     ;CHECK THAT POSITION BIT WAS SET
3272 012760 001003          BNE     38
3273 012762 010137 001162          MOV     R1,%REGO      ;GET DRIVE NUMBER
3274 012766 104010          ERROR      10        ;POSITIONING FLAG (IN KEY) IS NOT SET
3275                                     ;SET FOR INTERRUPTING DRIVE (GIVEN IN ERROR
3276                                     ;MESSAGE)
3277 012770 105761 001436          38:  TSTB  POS(R1)        ;CHECK POSITIONING FLAG WAS SET
3278 012774 001003          BNE     48
3279 012776 010137 001162          MOV     R1,%REGO      ;
3280 013002 104010          ERROR      10        ;'POSITIONING' FLAG WAS CLEAR, WHEN
3281                                     ;DOING A (POSITIONING) SEEK, THE FLAG
3282                                     ;'POS#' IS SET TO INDICATE THAT DRIVE
3283                                     ;IS POSITIONING HEADS (SEEKING). THIS
3284                                     ;ERROR INDICATES THAT FOR THE INTERRUPTING
3285                                     ;DRIVE (RKDS) POSITIONING FLAG WAS
3286                                     ;NOT SET.
3287 013004 004737 016540          48:  JSR      PC,CHKDRV      ;GO, CHECK FOR DRV ERRORS (DPL,DRU,DRY,WPS)
3288 013010 000501          BR      PFTERR        ;IF THERE IS A DRV ERROR, RETURN HERE
3289                                     ;THE DRIVE HAS BEEN DESELECTED, ALL
3290                                     ;FURTHER COMANDS ON THAT DRIVE ABORTED
3291                                     ;RETURN HERE IF NO DRIVE ERRORS
3292 013012 017777 166200 166210          MOV     @RKDS,@RKDA   ;GET THE DRIVE # FROM DRIVE ID BITS
3293                                     ;IN RKDS AND ADDRESS THE DRIVE
3294 013020 032777 000100 166170          BIT     @RMS,@RKDS    ;R/W/S RDY SET?
3295 013026 001005          BNE     58            ;YES
3296 013030 004737 022032          JSR      PC,GT4RG      ;GET RKCS,ER,DS,DA
3297 013034 010137 001250          MOV     R1,@RDRV      ;GET DRIVE #, FOR TYPING SERIAL #
3298 013040 104013          ERROR      18        ;R/W/S RDY NOT SET FOR INTERRUPTING
3299                                     ;DRIVE (RKDS), AFTER SEEK WAS COMPLETED,
3300 013042 032777 001000 166148 58:  BIT     @SIN,@RKDS    ;'SIN' ERROR?
3301 013050 001007          BNE     PSINER        ;YES, BRANCH
3302

```

```

3303 013052 032777 040000 166142          BIT     @HE,@RKCS     ;ANY HARD ERROR?
3304 013060 001060          BNE     POKER        ;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 013A70 004737 022032 PSINER: JSR PC,GT4RG ;THERE WAS A 'BIN' ERROR ON
;DOING POSITIONING (SEEK)
3311 013074 010137 001250 MOV R1,SRDRV
3312 013100 104016 ERROR 16 ;GET DRIVE #, FOR TYPING SERIAL #
; 'BIN' ON DOING (POSITIONING) SEEK
3313 013102 042710 000020 BIC #BIT4,(R0) ;CLEAR 'POSITIONING' BIT
3314 013108 103081 001438 CLRBB POS(R1) ;CLEAR POSITIONING FLAG
3315 013112 108081 001428 CLRBB BUSY(R1) ;CLEAR BUSY FLAG FOR THIS DRIVE
3316 013116 004737 016152 JSR PC,CLRBBIN ;CLEAR 'BIN' ERROR
3317
3318 013122 004737 016264 JSR PC,BINCNT ;KEEP COUNT OF 'BIN' ERRORS, IF MORE
;THAN 'MAX' DESELECT THE DRIVE
3319
3320 013126 000432 BR PFTERR ;RETURN HERE IF COUNT=MAX
3321
3322
3323 013130 105261 001446 PSRTRY: INCB RETRY(R1) ;RETURN HERE IF COUNT LESS THAN MAX
;INCRMNT REENTRY COUNT
3324 013134 105061 001436 CLRBB POS(R1) ;CLEAR POSITION FLAG
3325 013140 105061 001426 CLRBB BUSY(R1) ;CLEAR BUSY FLAG
3326 013144 122761 000003 CNPB #3,RETRY(R1) ;DONE ALL RETRIES?
3327 013152 001403 BEQ 18 ;YES
3328 013154 052710 010000 BIS #BIT12,(R0) ;SET HI PRIORITY ON RETRY
3329 013160 000207 RTS PC ;RETURN
3330
3331 013162 052710 104000 18: BIS #104000,(R0) ;INDICATE COMMAND ABORTED
3332 013166 105061 001446 CLRBB 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
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 CLRBB POS(R1) ;CLEAR POSITIONING FLAG FOR THIS DRIVE
3351 013250 105061 001426 CLRBB BUSY(R1) ;CLEAR BUSY FLAG FOR THIS DRIVE
3352 013254 000725 BR PSRTRY
    
```

```

3353
3354
3355
3356 013256 105037 001534 INTHND: CLRBB INTFLG ;ENTER THIS ROUTINE WHEN AN INTERRUPT
;OCCURS, NOTE THERE IS ONE OTHER
;INTERRUPT ROUTINE- 'INTISK'
3357
3358 013262 013766 001244 000002 MOV PPRVLV,2(SP) ;CPU PRIORITY TO 3 ON RTI
3359 013270 004737 020272 JSR PC,CHKCRDY ;CONTROL READY SET?
3360 013274 104024 ERROR 24 ;CONTROL RDY CLEAR AFTER INTRUPT ON COMPLETION
3361
3362 013276 032777 020000 165716 18: BIT #SCP,0RKCS ;OF FUNCTION, IT SHOULD BE SET
3363 013304 001403 BEQ 28 ;SCP SET? SEARCH COMPLETE?
3364 013306 004737 012704 JSR PC,SKCMP ;GO TO SEEK COMPLETE ROUTINE
3365 013312 000002 RTI
3366 013314 017746 165710 28: MOV #RKDA,-(BP) ;GET THE DRIVE # TO WHICH
;THE COMMAND WAS ISSUED UNDER
3367
3368
3369 013320 004737 022170 JSR PC,CROTFL ;INTERRUPT MODE
;ROTATE BITS 15,14,13 TO POSITION
3370 10,1,2
3371 013324 012608 MOV (BP)+,R5 ;POP OFF THE DRIVE # FROM THE STACK
3372 013326 105765 001426 TSTB R5(R5) ;CHECK THAT DRIVE WAS BUSY
3373 013332 100403 BMI 38
3374 013334 010537 001162 MOV R5,#REG0
3375 013340 104007 ERROR 7 ;'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!
3382
3383 013342 105065 001436 38: CLRBB POS(R5) ;CLEAR THE POSITION FLAG
3384 013346 116500 001426 MOVBB BUSY(R5),R0 ;GET THE ADDRESS OF THE
3385 013352 042700 177760 BIC #177760,R0 ;COMMAND KEY
3386 013356 062700 001306 ADD #KEY,R0
3387 013362 032710 000010 BIT #BIT3,(R0) ;CHECK IT'S NOT A SEEK FUNCTION
3388 013366 001401 BEQ 48
3389 013370 104020 ERROR 20 ;(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
3392
3393
3394
3395
3396
3397 013372 105710 48: TSTB (R0) ;MISINTERPRETATION
3398 013374 100403 BMI 58 ;CHECK THAT FUNCTION IN
;PROGRESS BIT WAS SET
3399 013376 010537 001162 MOV R5,#REG0 ;GET DRIVE NUMBER
3400 013402 104031 ERROR 31 ;IF NOT, ERROR
3401
3402
3403
3404
3405 013404 011002 58: MOV (R0),R2 ;'FUNCTION IN PROGRESS' FLAG (IN THE KEY)
3406 013406 042702 177770 BIC #177770,R2 ;WAS NOT SET FOR THE INTERRUPTING DRIVE.
3407 013412 020502 CMP R5,R2 ;IF THIS DRIVE WAS BUSY DOING THE
3408 013414 001405 BEQ 68 ;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 010837 001162      MOV      R8, $REG0      ;GET EXPTD DRIVE NO.
3410 013422 010237 001164      MOV      R2, $REG1      ;GET DRIVE NO, RECVD
3411 013426 104032      ERROR   32              ;UNEXPECTED DRIVE NO, INTERRUPTED
3412 013430 006302      ABL     R2              68:
3413 013432 011005      MOV      (R0),R3
3414 013434 010037 001836      MOV      R0,$AVKEY
3415 013440 000303      SHAB   R3
3416 013442 042703 177770      BIC     $177770,R3      ;GET POSITION OF THE PARAMETER
3417 013446 006303      ABL     R3              ;LIST FROM THE KEY
3418 013480 017704 188846      MOV      $RKC6,R4
3419 013484 042704 177761      BIC     $177761,R4      ;CLEAR ALL BUT FUNCTION CODE
3420 013460 016308 002032      MOV      PC,$R3,R6      ;CHECK THAT THE FUNCTION IN
                                   ;RKC6 AND THE KEY ARE SAME,
3421
3422 013464 126804 000002      CNPB   2(R6),R4
3423
3424 013470 001406      BRG     78              ;IF NOT, ERROR
3425 013472 016837 000002 001162      MOV      2(R6), $REG0      ;GET EXPTD FUNCTION CODE IN RKC6
3426 013900 010437 001164      MOV      R4, $REG1      ;GET CODE RECVD
3427 013904 104033      ERROR   33              ;FUNCTION CODE THAT IS IN RKC6 AFTER THIS
                                   ;INTERRUPT OCCURED IS NOT THE EXPECTED ONE
3428
3429 013906 042710 000200      BIC     $BIT7,(R0)      ;CLEAR 'FUNCTION IN PROGRESS' BIT
3430 013912 142761 000200 001426      BIC     $BIT7,$UBY(R1)   ;CLEAR BUSY FLAG FOR THIS DRIVE
3431 013920 004737 016840      JBR     PC,$RDRV       ;CHECK FOR DRIVE ERRORS
3432 013924 000002      RTI
3433
3434
3435 013926 032777 001000 168462      BIT     $SIN,$RKC6      ;IF THERE WAS ANY DRIVE ERROR
3436 013934 001426      BEQ     100             ;Deselect the drive and exit
3437 013936 004737 022222      JBR     PC,$R6DR       ;IF NO ERROR, RETURN HERE
3438
3439 013942 104016      ERROR   16             ;SIN ERROR?
3440 013944 004737 016152      JBR     PC,$LR6SIN
3441 013950 004737 016264      JBR     PC,$R6CWF
3442
3443 013954 000002      RTI
3444
3445 013956 105261 001446      INCB   $RETRY(R1)
3446 013962 122761 000003 001446      CNPB   63,$RETRY(R1)
3447 013970 001405      BEQ     88
3448 013972 082710 010000      BIS     $BIT12,(R0)
3449 013976 042710 000020      BIC     $BIT4,(R0)
3450 013902 000002      RTI
3451
3452 013604 004737 014426      88:  JBR     PC,$RYABE   ;ABORT THIS FUNCTION
3453 013610 000002      96:  RTI
3454
3455 013612 032777 -040000 168402 108:  BIT     $NE,$RKC6      ;HARD ERROR?
3456 013620 001076      BNE     $RDRER
3457 013622 032777 100000 168372      BIT     $ERR,$RKC6    ;SOFT ERROR?
3458 013630 001002      BNE     $FTERR
3459 013632 000137 014300      JMP     $RDRER        ;YES
                                   ;NO

```

```

3460
3461
3462
3463 013636 032777 000001 168384 $FTERR: BIT     $WCE,$RKR      ;THERE WAS A SOFT ERROR
3464 013944 001417      BEQ     10              ;WCE?
3465
3466 013646 032737 000001 001474      BIT     $BIT0,$RCODE   ;ALREADY WORKING ON A WC ERROR?
3467 013654 001084      BNE     30              ;YES - IGNORE THIS ONE
3468 013656 042737 000001 001474      BIC     $BIT0,$RCODE   ;ANY OTHER ERROR?
3469 013664 001082      BNE     46              ;YES - ABORT THIS FUNCTION
3470 013666 032737 000001 001474      BIS     $BIT0,$RCODE   ;SET THE WC ERROR BIT
3471
3472 013674 005262 001632      INC     $CECN(R6)      ;INCREMENT WCE COUNT FOR
3473 013700 104421 002070      TYPN60 , $R6R        ;THIS DRIVE
3474
3475 013704 032777 000002 168306 18:  BIT     $C6E,$RKR     ;C6E?
3476 013712 001417      BEQ     28
3477
3478 013714 032737 000002 001474      BIT     $BIT1,$RCODE   ;ALREADY WORKING ON A C6 ERROR?
3479 013722 001031      BNE     36              ;YES - IGNORE THIS ONE
3480 013724 042737 000002 001474      BIC     $BIT1,$RCODE   ;ANY OTHER ERROR?
3481 013732 001027      BNE     46              ;YES - ABORT THIS FUNCTION
3482 013734 032737 000002 001474      BIS     $BIT1,$RCODE   ;SET THE C6 ERROR BIT
3483
3484 013742 005262 001652      INC     $C6ECN(R6)    ;INCREMENT C6E COUNT FOR THIS DRIVE
3485 013746 104421 002076      TYPN60 , $R6R
3486 013752 104421 002120      TYPN60 , $R6R
3487 013756 004737 021736      JBR     PC,$TYFFN      ;GO TYPE OUT THE FUNCTION THAT GAVE ERROR
3488 013762 004737 022064      JBR     PC,$TYTIN
3489 013766 004737 022226      JBR     PC,$RDRV
3490 013772 104021      ERROR   31
3491
3492
3493
3494 013774 022704 000004      CNPB   64,R4
3495 014000 001002      BNE     36
3496 014002 004737 017110      JBR     PC,$R6DRCK     ;GO CHECK THE DATA READ
3497
3498 014006 000137 014160      38:  JMP     $MERR
3499 014012 000137 014216      46:  JMP     $MADR

```

```

3500 ;IF THERE WAS A HARD
3501 ;ERROR DURING THE DATA
3502 ;TRANSFER ENTER HERE
3503
3504 014A16 032777 010000 165174 HRDERR: BIT #SKE,ORKER ;SKE?
3505 014A24 001421 BEQ 1#
3506
3507 014A26 032737 000004 001474 BIT #BIT2,ERCODE ;ALREADY WORKING ON A SK ERROR?
3508 014A34 001081 BNE CMNERR ;YES - IGNORE THIS ONE
3509 014A36 042737 000004 001474 BIC #BIT2,ERCODE ;ANY OTHER ERROR?
3510 014A44 001064 BNE CMNABT ;YES - ABORT THIS FUNCTION
3511 014A46 052737 000004 001474 BIS #BIT2,ERCODE ;SET THE SK ERROR BIT
3512
3513 014A54 005262 001602 INC SKECN(R2) ;INCREMENT COUNT FOR SKE
3514 014060 005262 001562 INC HECCN(R2) ;ON THIS DRIVE
3515
3516 014A64 104421 002062 TYPMSG ,MSG1
3517
3518 014A70 032777 167740 165122 1# BIT #167740,ORKER ;ANY HE BESIDES SKE?
3519 014A76 001417 BEQ 2#
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 HECCN(R2) ;INCREMENT HE COUNT FOR
3528 014132 104421 002104 TYPMSG ,MSG4 ;THIS DRIVE
3529
3530 014136 104421 002120 2# TYPMSG ,MSG5 ;PRINT 'ON DOING'
3531 014142 004737 021736 JBR PC,TYPFN ;OO PRINT OUT THE FUNCTION
3532 ;WHICH GAVE THIS ERROR
3533
3534 014146 004737 022064 JBR PC,GETINF
3535 014152 004737 022226 JBR PC,GTSDRV ;SAVE DRIVE #, FOR TYPING SERIAL #
3536 014156 104022 ERROR 22 ;HARD ERROR ON DOING DATA XFER
3537
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,#SAVKY ;INDICATE HIGH PRIORITY ON RETRY
3543 014A02 042777 000020 165326 BIC #BIT4,#SAVKY ;CLEAR 'POSITIONING HEADS', IF SET
3544 014A10 004737 016006 JBR PC,CLRERR ;CLEAR THIS ERROR
3545 014A14 000002 RTI
3546
3547 014A16 005037 001474 CMNABT: CLR ERCODE ;CLEAR ERROR CODE
3548 014A22 104421 002532 TYPMSG ,MSG27
3549 014A26 004737 014426 JBR PC,DRVABT ;ABORT THE FUNCTION
3550 014A32 032777 010000 164760 BIT #SKE,ORKER ;SEEK ERROR?
3551 014A40 001416 BEQ 3# ;NO
3552 014A42 104416 CON,RESET ;RESET THE CONTROLLER
3553 014A44 010137 001502 MOV R1,QDRV ;SET DRIVE NUMBER
3554 014A50 000241 CLC ;CLEAR THE 'C' BIT
3555 014A52 006037 001502 ROR QDRV ;LEFT JUSTIFY DRIVE NUMBER
    
```

```

3556 014A56 006037 001502 ROR QDRV ;LEFT JUSTIFY DRIVE NUMBER
3557 014A62 006037 001502 ROR QDRV ;LEFT JUSTIFY DRIVE NUMBER
3558 014A66 006037 001502 ROR QDRV ;LEFT JUSTIFY DRIVE NUMBER
3559 014272 104420 DRV,RESET ;RESET THE DRIVE
3560 014A74 104416 CON,RESET ;RESET THE CONTROLLER
3561 014A76 000002 3# RTI ;THIS DATA TRANSFER
    
```

```

3562
3563 ;IF THERE WAS NO HARD OR
3564 ;SOFT ERROR ON DATA TRANSFER
3565 014900 105761 001446 NOERROR: TSTB RETRY(R1) ;ENTER HERE
3566 014904 001406 BEQ 18
3567 014906 104421 002604 TYPMSG ,MSG28
3568 014912 116146 001446 MOVB RETRY(R1),*(SP)
3569 014916 104403 TYPDS
3570 014920 002 ,BYTE 2
3571 014921 000 ,BYTE 0
3572
3573 014922 008037 001474 18: CLR ERCODE
3574 014926 108061 001446 CLRB RETRY(R1)
3575 014932 042777 010000 168176 BIC %BIT13,%SAVKEY ;CLEAR PRIORITY BIT IF SET
3576 014940 004737 021006 JBR PC,STATSTC ;DO, COLLECT STATISTICS ON THIS
3577 ;DATA TRANSFER
3578
3579 014944 022704 000004 CMP #4,R4 ;WAS IT A 'READ' FUNCTION?
3580 014980 001002 BNE 28
3581 014982 004737 017110 JBR PC,DATCHK ;IF IT WAS 'READ', CHECK THE DATA
3582
3583
3584
3585 014956 022706 000002 28: CMP #2,R4 ;WAS IT A 'WRITE' FUNCTION?
3586 014962 001011 BNE 38
3587 014964 032777 000040 168144 BIT %BITS,%SAVKEY ;IS 'WRT CHK' TO FOLLOW THIS
3588 014972 001412 BEQ 48 ;'WRT' FUNCTION?
3589 014974 082703 100000 BIS %BIT15,R3 ;SET FLAG BIT TO INDICATE
3590 ;THAT WRITE CHECK SHOULD
3591 ;FOLLOW THIS WRITE
3592 014400 010337 001456 MOV R3,WCFLG ;SET UP WC FLAG TO INDICATE
3593 ;THE ABOVE, THE LOWER BYTE
3594 ;CONTAINS THE POINTER TO THE
3595 ;COMMAND LIST, WHICH WILL
3596 ;BE USED, FOR DOING WRITE CHECK
3597 014404 000407 BR 58 ;IF WRITE CHECK IS TO BE DONE, DONT
3598 ;SET BIT 15 OF THE KEY TILL WRT CHK
3599 ;IS DONE
3600
3601 014406 022704 000006 38: CMP #6,R4 ;WRT CHK FUNCTION?
3602 014412 001002 BNE 48
3603 014414 005037 001456 CLR WCFLG ;CLEAR WR CHK FLAG
3604
3605 014420 052710 100000 48: BIS %BIT15,(R0) ;SET FLAG TO INDICATE THIS
3606 014424 000002 58: 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 008262 001672 INC ABORT(R2)
3618 014454 026227 001672 000010 CMP ABORT(R2),#10
3619 014462 003402 BLE 18
3620 014464 000137 016312 JMP DSELCT ;DROP THE DRIVE
3621 014470 012602 16: MOV (SP)+,R2 ;RESTORE R2
3622 014472 104401 002165 TYPE ,MSG10 ;TYPE 'ABORTED'
3623 014476 000207 RTB PC ;RETURN
    
```

```

3624 ;THIS ROUTINE GENERATES THE 8 COMMAND REQUESTS AND PUT THEM IN THE QUEUE, THE
3625 ;FOLLOWING PARAMETERS ARE GENERATED RANDOMLY;
3626 ;1. DRIVE NUMBER ON WHICH THE COMMAND WILL BE PERFORMED.
3627 ;2. FUNCTION TO BE PERFORMED.
3628 ;3. DISK ADDRESS - CYLINDER, SURFACE, SECTOR.
3629 ;4. STARTING BUS ADDRESS.
3630 ;5. WORD COUNT FOR DATA TRANSFER.
3631 ;THE QUEUE IS LOCATED AT 'CMND' (TO 'CMND8'),
3632
3633
3634 014500 104407 GENBRQ: CKSWR
3635 014502 005337 DEC REPCNT ;DECREMENT REPETITION COUNT
3636 014506 001025 BNE ;CONTINUE IF NOT 0
3637 014510 013737 001236 002056 MOV PCNTR,REPCNT ;REESTABLISH REPETITION COUNT FOR EXERCISER
3638 014516 104401 014524 TYPE ,656 ;TYPE ASCIZ STRING
3639 014522 000407 BR 648 ;GET OVER THE ASCIZ
3640 ;;656: ,ASCIZ <15><12>/END OF PASS/
3641 648:
3642 014542 013746 001100 MOV $PASS,=(SP)
3643 014546 005216 INC (SP)
3644 014550 104405 TYPDS
3645 014552 004737 021116 JBR PC,REPSTAT
3646 014556 000137 022704 JMP $EOP
3647
3648 014562 032777 000400 164350 18: BIT $SW8,$SWR
3649 014570 001422 BEQ 28
3650 014572 032777 000001 164340 BIT $SW0,$SWR ;SWITCH 0 SET ?
3651 014600 001410 BEQ 88 ;BR IF NOT
3652 014602 008727 TST (PC)+ ;CHECK INDICATOR
3653 014604 000000 ,WORD 0 ;TYPE TIME INDICATOR
3654 014606 001013 BNE 28 ;BR IF TIME ALREADY TYPED
3655 014610 008237 014604 INC 76 ;INCREMENT THE INDICATOR
3656 014614 004737 026660 JBR PC,TIMTYP ;TYPE THE TIME (IF SWR 03 SET)
3657 014620 000406 BR 28 ;CONTINUE
3658 014622 008037 014604 88: CLR 76 ;CLEAR THE INDICATOR
3659 014626 104401 001213 TYPE ,$CRLF
3660 014632 004737 021116 JBR PC,REPSTAT
3661 014636 032777 000040 164274 28: BIT $SW5,$SWR ;HALT
3662 014644 001401 BEQ 36 ;NO
3663 014646 000000 HALT ;YES, PRESSING CONTINUE RESUMES PROG EXECUTION
3664
3665 014650 004737 022886 36: JBR PC,CNDPRS ;CHECK IF ANY DRIVES PRESENT
3666 014654 012700 001306 48: MOV $KEY,RO
3667 014660 010004 MOV RO,R4
3668 014662 008001 CLR R1
3669 014664 010120 58: MOV R1,(RO)+ ;CLEAR THE 8 COMMAND KEYS
3670 014666 062701 000400 ADD $400,R1 ;BITS 0,1,2,3,4,5,6,7
3671 014672 022701 004000 CMP $4000,R1 ;POSITION 0,1,2,3,4,5,6,7
3672 014676 001372 BNE 58
3673 014700 012700 001326 MOV $CMND,RO ;CLEAR THE PARAMETER TABLE
3674 014704 010005 MOV RO,R8
3675 014706 012701 177740 MOV $=40,R1 ;FOR THE 8 COMMANDS IN Q
3676 014712 005020 68: CLR (RO)+ ;(8X4) WORDS IN ALL
3677 014714 005201 INC R1
3678 014716 001375 BNE 68
3679
    
```

```

3680 014720 004737 015770 JBR PC,CLRFLGS ;CLEAR THE FLAGS PERTAINING TO THE 8
3681 ;COMMANDS IN THE Q
3682 ;R4 CONTAINS 'KEY'
3683 ;R5 CONTAINS 'CMND'
3684 014724 022737 000001 001264 GEN1: CMP #1,DRVPRS ;ONLY 1 DRV PRESENT?
3685 014732 001002 BNE 228 ;NO
3686 014734 005000 CLR RO ;YES
3687 014736 000420 BR 36
3688 014740 004737 025576 228: JBR PC,$RAND ;GENERATE A RANDOM NUMBER
3689 ;GET A RANDOM DRIVE NUMBER
3690 ;FROM THE AVAILABLE DRIVES
3691 014744 025730 R8DRV1
3692
3693 014746 013746 025732 MOV R8DRVH,=(SP) ;PUT LOW DIVIDEND ON STACK
3694 014752 005046 CLR =(SP) ;CLEAR HIGH DIVIDEND AND PUSH
3695 ;IT ON STACK
3696 014754 013746 001520 MOV DRMAP,=(SP) ;PUSH DIVISOR ON STACK
3697 014760 004737 025212 JBR PC,$$DIV ;GO TO THE 'DIVIDE' SUBROUTINE
3698 014764 005726 TST (SP)+ ;DISCARD THE REMAINDER, QUOTIENT IS
3699 ;NOW ON TOP OF THE STACK
3700 014766 012600 MOV (SP)+,RO
3701 014770 020037 001264 CMP RO,DRVPRS ;MAKE SURE CORRECT MAPPING IS DONE
3702 014774 001001 BNE 36
3703 014776 005300 DEC RO ;'QDRVE' CONTAINS RANDOM DRIVE NO.
3704 015000 005037 001502 38: CLR QDRV
3705 015004 116037 001254 001802 MOVB PDR(RO),QDRV
3706 015012 105737 001502 TSTB QDRV ;TEST IF TYPE F DRIVE
3707 015016 100016 BPL 326 ;NOT IF POSITIVE
3708
3709 015020 142737 000200 001502 BICB $200,QDRV ;CLEAR THE FLAG BIT
3710 015026 132737 000001 001802 BITB $1,QDRV ;ODD OR EVEN DRIVE ADDRESS
3711 015034 001404 BEQ 318
3712
3713 015036 005737 015724 TST ODDEVN ;MUST HAVE BEEN AN ODD ONE
3714 015042 001730 BEQ GEN1 ;INSURE THAT ODDS WHAT WE WANT
3715 015044 000403 BR 326
3716
3717 015046 005737 015724 318: TST ODDEVN ;MUST HAVE BEEN AN EVEN ONE
3718 015052 001332 BNE 228 ;NO GOOD = TRY AGAIN
3719
3720 015054 004737 025576 328: JBR PC,$RAND ;GENERATE A RANDOM NUMBER
3721 015060 025734 R8FUN1
3722
3723 015062 013746 025736 MOV R8FUNH,=(SP) ;PUT LOW DIVIDEND ON STACK
3724 015066 005046 CLR =(SP) ;CLEAR HIGH DIVIDEND AND PUSH
3725 ;IT ON STACK
3726 015070 013746 001526 MOV FNMAP,=(SP) ;PUSH DIVISOR ON STACK
3727 015074 004737 025212 JBR PC,$$DIV ;GO TO THE 'DIVIDE' SUBROUTINE
3728 015100 005726 TST (SP)+ ;DISCARD THE REMAINDER, QUOTIENT IS
3729 ;NOW ON TOP OF THE STACK
3730 015102 021627 000003 CMP (SP),#3 ;MAKE SURE CORRECT FUNCTION IS SELCTD
3731 015106 001001 BNE 208
3732 015110 005316 DEC (SP)
3733 015112 012637 001512 208: MOV (SP)+,QFNC ;THE FUNCTION THAT CAN BE PERFORMED
3734
3735 015116 004737 025576 JBR PC,$RAND ;GENERATE A RANDOM NUMBER
    
```

```

3736 01522 025740 RSCYL
3737
3738 01524 013746 025742 MOV RSCYL,=(SP) ;PUT LOW DIVIDEND ON STACK
3739 01530 005046 CLR =(SP) ;CLEAR HIGH DIVIDEND AND PUSH
3740 ;IT ON STACK
3741 01532 013746 001522 MOV CYLMAP,=(SP) ;PUSH DIVISOR ON STACK
3742 01536 004737 025212 JBR PC,@@DIV ;GO TO THE 'DIVIDE' SUBROUTINE
3743 01542 005726 T&T (SP)+ ;DISCARD THE REMAINDER, QUOTIENT IS
;NOW ON TOP OF THE STACK
3744
3745 01544 012637 001504 MOV (SP)+,GCYL ;(FROM 0-312)
3746 01540 022737 000313 001504 CMP #313,GCYL
3747 01546 001002 BNE 48
3748 01540 005337 001504 DEC GCYL ;'GCYL' CONTAINS RANDOM CYLINDER NO.
3749
3750 015164 48:
3751
3752 015164 013746 025742 MOV RSCYL,=(SP) ;PUT LOW DIVIDEND ON STACK
3753 015170 005046 CLR =(SP) ;CLEAR HIGH DIVIDEND AND PUSH
3754 ;IT ON STACK
3755 01572 013746 001524 MOV @ECMAP,=(SP) ;PUSH DIVISOR ON STACK
3756 01576 004737 025212 JBR PC,@@DIV ;GO TO THE 'DIVIDE' SUBROUTINE
3757 015902 005726 T&T (SP)+ ;DISCARD THE REMAINDER, QUOTIENT IS
;NOW ON TOP OF THE STACK
3758
3759 015904 012637 001510 MOV (SP)+,@SEC ;(BETWEEN 0-13/8)
3760 015910 022737 000014 001510 CMP #14,@SEC
3761 015916 001002 BNE 58
3762 015920 005337 001510 DEC @SEC ;'QSEC' CONTAINS RANDOM SEC #
3763
3764 015924 022737 077777 025742 58: CMP #77777,RSCYLH ;SELECT SURFACE # RANDOMLY
3765 015932 101005 BHI 68
3766 015934 012737 000020 001506 MOV #BIT4,@SUR ;SURFACE 1
;R1 CONTAINS THE MAXM WORD COUNT BASED ON
;AVAILABLE DISK SPACE.
;R2 CONTAINS THE MAXM WORD COUNT BASED ON
;AVAILABLE MEMORY SPACE.
3767
3768
3769
3770
3771 015942 005001 CLR R1
3772 015944 000404 BR 78
3773
3774 015946 005037 001506 68: CLR @SUR ;SURFACE 0
3775 015952 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, CHANGES 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 015956 023727 001504 000306 78: CMP GCYL,#306 ;IS THE RANDOM CYL # GREATER THAN 306?
3784 015964 002003 BGE 98
3785 015966 012701 177777 88: MOV #177777,R1 ;IF YES, MAKE SURE THAT THE
3786 015972 000431 BR 218 ;WORD COUNT IS SELECTED PROPERLY
;COMPUTE MAXM WORD COUNT BASED ON
;AVAILABLE DISK SPACE
3787
3788
3789 015974 012700 000014 98: MOV #14,R0 ;COMPUTE # OF SECTORS AVAILABLE ON
3790 015980 163700 001510 SUB @SEC,R0 ;CYLINDERS SELECTED
3791 015984 060001 ADD R0,R1
    
```

```

3792 015906 012703 000312 MOV #312,R3 ;COMPUTE # OF SECTORS AVAILABLE
3793 015912 163703 001504 SUB GCYL,R3 ;BEYOND THE CYLINDER SELECTED
3794 015916 012746 000030 MOV R3,=(SP) ;PUT THE MULTIPLIER ON THE STACK
3795 015922 010346 MOV R3,=(SP) ;PUT THE MULTIPLICAND ON THE STACK
3796 015924 004737 025100 JBR PC,@@MULT ;CALL THE MULTIPLY ROUTINE
3797 015930 012616 MOV (SP)+,(SP) ;DISREGARD THE HSB'S
3798 015932 012603 MOV (SP)+,R3 ;GET THE LSB'S OF THE PRODUCT
3799 015934 060103 ADD R1,R3 ;COMPUTE TOTAL # OF SECTORS AVAILABLE
3800 015936 012746 000400 MOV #400,=(SP) ;PUT THE MULTIPLIER ON THE STACK
3801 015942 010346 MOV R3,=(SP) ;PUT THE MULTIPLICAND ON THE STACK
3802 015944 004737 025100 JBR PC,@@MULT ;CALL THE MULTIPLY ROUTINE
3803 015950 012616 MOV (SP)+,(SP) ;DISREGARD THE HSB'S
3804 015952 012603 MOV (SP)+,R3 ;GET THE LSB'S OF THE PRODUCT
3805 015954 010301 MOV R3,R1 ;COMPUTE TOTAL # OF WORDS=SPACE
;AVAILABLE ON DISK FROM THE SELECTED
;CYL #
3806
3807 ;COMPUTE MAXM WORD COUNT BASED ON
3808 ;AVAILABLE MEMORY SPACE.
3809
3810 015956 004737 025576 218: JBR @RAND PC,@RAND ;GENERATE RANDOM NUMBER
3811 015962 025744 RBBAL
3812
3813 015964 013746 025746 MOV RBBAN,=(SP) ;PUT LOW DIVIDEND ON STACK
3814 015970 005046 CLR =(SP) ;CLEAR HIGH DIVIDEND AND PUSH
3815 ;IT ON STACK
3816 015972 013746 001530 MOV @ANAP,=(SP) ;PUSH DIVISOR ON STACK
3817 015976 004737 025212 JBR PC,@@DIV ;GO TO THE 'DIVIDE' SUBROUTINE
3818 015402 005726 T&T (SP)+ ;DISCARD THE REMAINDER, QUOTIENT IS
;NOW ON TOP OF THE STACK
3819
3820 015404 012637 001514 MOV (SP)+,@BUSAD ;BE USED
3821 015410 006337 001514 ABL @BUSAD
3822 015414 063737 002052 001514 ADD @BASEBA,@BUSAD ;FORM THE RANDOM BUS-ADDRESS
;BY ADDING RANDOM OFFSET TO
;THE BASE BUS-ADDRESS
3823
3824
3825 015422 013703 002054 MOV MAXBA,R3 ;COMPUTE # OF WORDS THAT
3826 015426 163703 001514 SUB @BUSAD,R3 ;CAN BE TRANSFERRED, USING THE
;ABOVE GENERATED BUS-ADDRESS WITHOUT
;CAUSING A NXH
3827
3828 015432 000241 CLC R3
3829 015434 006003 ROR R3
3830 015436 010302 MOV R3,R2
3831
3832 015440 020201 108: CMP R2,R1 ;SELECT SMALLER OF THE TWO
3833 015442 103401 SLO 118 ;WORD-COUNTS THAT WILL BE
;USED FOR GENERATING A RANDOM
;WORD COUNT)
3834 015444 010103 MOV R1,R3
3835
3836 ;R3 CONTAINS THE MAXM WORD COUNT
3837 ;POSSIBLE, COMPUTE THE WORD COUNT
;MAPPING FACTOR FROM THIS.
3838
3839 015446 118:
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 JBR PC,@@DIV ;GO TO THE 'DIVIDE' SUBROUTINE
3846 015462 005726 T&T (SP)+ ;DISCARD THE REMAINDER, QUOTIENT IS
;NOW ON TOP OF THE STACK
3847
    
```

3848	015464	005216		INC	(8P)				
3849	015466	012637	001832	MOV	(8P)+,WCMAP				;WORD COUNT MAPPING FACTOR
3850									
3851	015472	004737	028976	JBR	PC,8RAND				;GENERATE A RANDOM NUMBER
3852	015476	028980		R8WCL					
3853									
3854	015400	013746	028982	MOV	R8WCH,-(8P)				;PUT LOW DIVIDEND ON STACK
3855	015406	008046		CLR	-(8P)				;CLEAR HIGH DIVIDEND AND PUSH
3856									;IT ON STACK
3857	015406	013746	001832	MOV	WCMAP,-(8P)				;PUSH DIVISOR ON STACK
3858	015412	004737	028212	JBR	PC,88DIV				;GO TO THE 'DIVIDE' SUBROUTINE
3859	015416	005726		TST	(8P)+				;DISCARD THE REMAINDER, QUOTIENT IS
3860									;NOW ON TOP OF THE STACK
3861	015420	012637	001816	MOV	(8P)+,QWRCNT				;QWRCNT CONTAINS THE RANDOM
3862									;WORD COUNT THAT WILL BE USED
3863	015424	005737	001816	TST	QWRCNT				;MAKE SURE THE WORD COUNT IS
3864	015430	003004		BGT	128				;NOT 0
3865	015432	008437	001816	NEG	QWRCNT				;TAKE CARE OF ZERO AND NEG NUMBS
3866	015436	005237	001816	INC	QWRCNT				
3867	015442	113700	001802	128:	MOV8	QDRV,R0			
3868	015446	000241		CLC					
3869	015450	006000		ROR	R0				
3870	015452	006000		ROR	R0				;POSITION THE DRIVE NUMBER IN
3871	015454	006000		ROR	R0				;BITS 15,14,13
3872	015456	006000		ROR	R0				
3873	015460	013701	001504	MOV	QCYL,R1				
3874	015464	000301		SWAB	R1				
3875	015466	000241		CLC					
3876	015470	006001		ROR	R1				;POSITION THE CYLINDER NUMBER
3877	015472	006001		ROR	R1				;IN BITS 12-5
3878	015474	006001		ROR	R1				
3879	015476	050100		BIS	R1,R0				
3880	015400	053700	001510	BIS	QSEC,R0				;R0 CONTAINS THE COMPLETE DISK
3881	015404	053700	001506	BIS	QSUR,R0				;ADDRESS
3882	015610	010025		MOV	R0,(R8)+				;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,OFNC				
3887	015620	001412		BEQ	28				;READ
3888	015622	003014		BGT	148				;READ CHECK
3889	015624	012725	000002 18:	MOV	#2,(R8)+				;WRITE FUNCTION CODE
3890	015530	023727	025752 077777	CMP	R8WCH,#777777				;SHOULD WRITE CHECK BE DONE
3891	015436	101010		BHI	158				;AFTER THE WRITE?
3892	015440	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	158				
3896									
3897	015646	012725	000004 28:	MOV	#4,(R8)+				;READ FUNCTION CODE
3898	015652	000402		BR	158				
3899									
3900	015654	012725	000012 148:	MOV	#12,(R8)+				;READ CHECK FUNCTION CODE
3901									
3902	015660	013715	001516 158:	MOV	QWRCNT,(R8)				;INSERT THE WORD COUNT (RKWC)
3903	015664	005425		NEG	(R8)+				;2'S COMPLEMENT

3904	015666	013725	001514	MOV	QSUBAD,(R8)+				;INSERT THE SUB ADDRESS (RKBA) FOR
3905									;THIS COMMAND IN THE PARAMETER
3906									;TABLE
3907	015672	053724	001502 168:	BIS	QDRV,(R4)+				;SET THE DRIVE NUMBER INSIDE
3908									;THE KEY FOR THIS COMMAND
3909	015676	020427	001326	CMP	R4,#KEY+20				;GENERATED 8 COMMANDS IN
3910									;THE QUEUE?
3911	015702	001402		BEQ	178				
3912	015704	000137	014724	JMP	GEN1				;IF NOT, GO BACK AND GENERATE
3913									;THE NEXT COMMAND AND THE
3914									;PARAMETERS (RKWC, BA, DA)
3915	015710	005237	015724 178:	INC	ODDEVN				;CHANGE FROM ODD/EVEN TO EVEN/ODD
3916	015714	042737	177776 015724	BIC	#177776,ODDEVN				;KEEP ONLY ONE BIT
3917	015722	000207		R8	PC				;ALL 8 COMMANDS HAVE BEEN
3918									;GENERATED IN THE TASK-QUEUE
3919	015724	000000		ODDEVN;	0				

```

3920 ;ENTER THIS CODE IF SW 9 HAS BEEN SET AND LOOPING HAS TO BE DONE ON ERROR
3921 ;CONTROL IS TRANSFERRED TO THIS, ON RETURN FROM THE ERROR HANDLER '%KRRON'.
3922
3923
3924 015726 004737 016006 EXCLPLU: 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 18: BIC #114220,(R1)+
3929 015744 020127 001326 CMP R1,#KEY+20
3930 015750 001373 BNE 18
3931 015752 012601 MOV (SP)+,R1
3932 015754 004737 015770 JSR PC,CLRFLGS ;CLEAR OUT THE VARIOUS FLAGS PERTAINING
3933 ;TO THE 9 COMMANDS IN THE 9
3934 015760 012706 001100 MOV #STACK,SP ;REESTABLISH STACK POINTER
3935 015764 001137 010650 JMP QMNGER ;START OVER AGAIN, PROCESS THE COMMANDS
3936 ;IN THE 9 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 ;CLRFLGS
3945 ;THIS ROUTINE CLEARS OUT THE VARIOUS FLAGS USED FOR THE 9 COMMANDS IN THE QUEUE.
3946
3947 015770 012700 001426 CLRFLGS: MOV #BUSY,RO ;CLEAR THE 9 BUSY FLAGS
3948 015774 005020 18: CLR (RO)+
3949 015776 020027 001462 CMP RO,#QSCNT+2 ;ALL DONE?
3950 016002 001374 BNE 18 ;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 015765 001426 18: TSTB BUSY(R5) ;WAS THIS DRIVE BUSY SEEKING?
3966 016024 100035 BPL 44 ;NO
3967 016026 005037 CLR TIMER
3968 016032 032777 000100 163156 28: BIT #RWS,#RKDS ;R/W/S SET?
3969 016040 001015 BNE 38 ;YES
3970 016042 005237 001472 INC TIMER ;KEEP TIME
3971 016046 001371 BNE 28 ;WAIT FOR R/W/S RDY
3972 016050 004737 022222 JSR PC,RQ48DR ;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 38
3978 016066 004737 022222 JSR PC,RQ48DR ;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 116304 001426 38: MOV# 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 CLAB BUSY(R5) ;MAKE THIS DRIVE FREE, AVAILABLE
3988
3989
3990 016120 062777 020000 163102 48: ADD #20000,#RKDA ;ADDRESS THE NEXT POSSIBLE DRIVE
3991 016126 005205 INC R5 ;INCREMENT COUNT
3992 016130 022709 000010 CMP #10,R5 ;ALL DONE
3993 016134 001331 BNE 18 ;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
3999 016146 012604 MOV (SP)+,R4
4000 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 @RKDA,QWRCNT ;SAVE DISK ADDRESS
4010 016160 004737 016006 JSR PC,CLRERR ;GO, WAIT FOR OTHER DRIVES TO COMPLETE
4011 ;THEIR SEEKS(IF THEY ARE DOING ANY)
4012 ;THEN DO CON,RESET TO CLR THE ERROR.
4013 016164 013777 001516 163036 MOV QWRCNT,@RKDA ;ADDRESS THE DRIVE AGAIN
4014 016172 004737 020314 JSR PC,DRCHND ;SAVE INFO ABOUT THE PAST & PRESENT COMAND
4015 016176 012777 000015 163016 MOV @IS,@RKCS ;DO DRIVE RESET ON THE
4016 ;DRIVE
4017 ;INDICATED IN RKDA
4018 016904 104417 CON,RDY
4019 016906 005037 001472 CLR TIMER
4020 016912 032777 000100 162776 18: BIT @RWS,@RKDS ;WAIT FOR R/W/S RDY TO SET
4021 016920 001015 BNE 28
4022 016922 005237 001472 INC TIMER
4023 016926 001371 BNE 16
4024 016930 004737 022222 JSR PC,RC48DR ;GET RKCS, ER, DS, DA AND DRIVE # FOR
4025 ;TYPING SERIAL DRIVE #
4026 016934 104004 ERROR 4 ;R/W/S RDY DID NOT SET AFTER
4027 ;DOING DRIVE RESET, TIMED OUT
4028 016936 032777 001000 162752 BIT @SIN,@RKDS
4029 016944 001403 BEQ 28
4030 016946 004737 022222 JSR PC,RC48DR ;GET RKCS, ER, DS, DA AND DRIVE # FOR
4031 ;TYPING SERIAL DRIVE #
4032 016952 104016 ERROR 16 ;A DRIVE RESET WAS DONE ON THIS DRIVE
4033 ;TO CLEAR 'SIN', BUT 'SIN' DID NOT GET
4034 ;CLEARED
4035
4036 016954 004737 020340 28: JSR PC,CRCHND ;SAVE INFO ABOUT THIS COMMAND
4037 016960 104416 CON,RESET ;DO IT TO CLEAR OUT MASK F/FS
4038 016962 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 016964 105261 001622 SINCNT: INCB @SINC(R1) ;INCREMENT 'SIN' COUNT FOR THIS DRIVE
4052 016970 122761 000005 001622 CMPB #5,@SINC(R1) ;5 ERRORS OCCURRED?
4053 016976 101403 BLOS 18 ;YES
4054 016980 062716 000002 ADD #2,(R1) ;ADJUST PC FOR RETURN TO THE RIGHT POINT
4055 016984 000207 RTS PC ;RETURN
4056
4057 016986 000137 016312 18: JMP DSELECT ;5 ERRORS OCCURRED, GO DESELECT
    
```

```

4058 ;DSELECT
4059 ;THIS ROUTINE IS ENTERED WHEN A DRIVE IS TO BE DESELECTED (TAKEN
4060 ;OUT OF SELECTION LIST), BECAUSE THE FATAL ERRORS ON THAT DRIVE
4061 ;HAS REACHED A MAXIMUM COUNT, R1 CONTAINS THE DRIVE NUMBER THAT
4062 ;THAT IS TO BE DESELECTED, THE DRIVE IS DESELECTED IF 1. TOTAL SIM COUNT
4063 ;FOR THAT DRIVE REACHES THE MAXIMUM ALLOWABLE 2, IF A FATAL ERROR
4064 ;LIKE DRIVE UNSAFE, DRIVE POWER LOW OCCURS, 3. IF WPS GETS SET, OR DRY
4065 ;IS CLEAR.
4066
4067
4068
4069 016912 012705 001253 DSELECT: NOV $PDR=R1,R8
4070 016916 013702 001264 NOV DRVPR8,R2 ;NUMBER OF DRIVES BEING TESTED
4071 016922 062702 001253 ADD $PDR=R1,R2 ;FOR END ADDRESS OF TABLE
4072 016926 005208 10: INC R8 ;LOCATE THE DRIVE (TO BE
4073 016930 111503 177600 MOVB (R8),R3 ;DESELECTED) IN THE TABLE
4074 016932 042703 BIC $177600,R3 ;DROP THE F FLAG
4075 016936 020301 CMP R3,R1 ;IS THIS THE ONE
4076 016940 001403 BEQ 20 ;CONTAINING AVAILABLE DRIVES
4077 016942 020502 CMP R5,R2 ;FINISHED ?
4078 016944 103770 BLO 10 ;BR IF NOT
4079 016946 000472 BR 110 ;DRIVE WAS NOT FOUND IN TABLE, EXIT
4080 016950 111502 20: MOVB (R8),R2 ;GET THE DRIVE NUMBER
4081 016952 020527 001264 50: CMP R5,$PDR+10 ;IS THIS DRIVE $ THE LAST ENTRY IN TABLE?
4082 016956 001406 BEQ 40 ;YES
4083 016960 010504 MOV R5,R4
4084 016962 005205 INC R5 ;IF NOT, TAKE OUT THIS DRIVE $ FROM
4085 016964 112524 30: MOVB (R5)+,(R4)+ ;THE MIDDLE AND PUSH UP THE
4086 016966 022704 001263 CMP $PDR+7,R4 ;REST OF THE ENTRIES
4087 016972 001374 BNE 30
4088 016976 105065 177777 40: CLRB *(R5) ;CLEAR LAST ENTRY IN TABLE
4089
4090 ;THE DRIVE $ TYPED OUT WAS DESELECTED
4091 ;BECAUSE ERROR COUNT EXCEEDED THE
4092 ;MAXIMUM ALLOWABLE
4093 016400 104401 002336 TYPE ,NSG10 ;TYPE "DRIVE DROPPED"
4094 016404 010146 MOV R1,=(SP) ;PUSH DRIVE NUMBER ON STACK
4095 016406 104403 TYP05 ;TYPE IT ON THE TERMINAL
4096 016410 001 ,BYTE 1
4097 016411 000 ,BYTE 0
4098 016412 004737 026766 JBR PC,$NOTYP ;GO TYPE OUT SERIAL NO OF THE DRIVE,
4099 ;IF $W 1 IS SET.
4100 016416 005337 001264 DEC DRVPRS ;DECREMENT THE TOTAL NUMBER OF
4101 ;DRIVES PRESENT
4102 016422 004737 022686 JBR PC,CHOPRS ;CHECK IF ANY DRIVES PRESENT
4103 ;IF NONE GOT TO END OF PASS, $EOP
4104
4105 016426 012746 177777 MOV $177777,=(SP) ;PUT LOW DIVIDEND ON STACK
4106 016432 005046 CLR *(SP) ;CLEAR HIGH DIVIDEND AND PUSH
4107 ;IT ON STACK
4108 016434 013746 001264 MOV DRVPR8,=(SP) ;PUSH DIVISOR ON STACK
4109 016440 004737 025212 JBR PC,$$DIV ;GO TO THE "DIVIDE" SUBROUTINE
4110 016444 005726 TGT (SP)+ ;DISCARD THE REMAINDER, QUOTIENT IS
4111 ;NOW ON TOP OF THE STACK
4112 016446 012637 001520 MOV (SP)+,DRMAP ;TO BE USED FOR GENERATING RANDOM
4113 ;DRIVE NUMBERS.
    
```

```

4114 016452 012704 001306 MOV $KEY,R4 ;DESELECT THE COMMANDS
4115 016456 011405 60: MOV (R4),R5 ;IN THE "COMMAND 0" CORRESPONDING
4116 016460 042705 177770 BIC $177770,R5 ;TO THE DESELECTED DRIVE
4117 016464 020105 CMP R1,R5
4118 016466 001002 BNE 70
4119 016470 052714 104000 BIS $104000,(R6) ;INDICATE COMMAND DESELECTED
4120 016474 005724 70: TGT (R4)+ ;(AND COMPLETED)
4121 016476 022704 001326 CMP $KEY+20,R4
4122 016502 001365 BNE 66
4123
4124 016506 105702 80: TSTB R2 ;"F" TYPE DRIVE ?
4125 016506 100012 SPL 110 ;NO - JUST EXIT
4126 016510 032701 000001 BIT $1,R1 ;ODD OR EVEN DRIVE NUMBER
4127 016514 001403 BEQ 90
4128 016516 042701 000001 BIC $1,R1
4129 016522 000402 BR 100
4130 016524 052701 000001 90: BIS $1,R1
4131 016530 000137 016312 100: JMP DSELECT ;DROP CORRESPONDING DRIVE
4132
4133
4134 016534 000137 010630 110: JMP BEGNEX ;GO RESTART EXERCISOR 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, R1
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 016440 017746 162464 CHKDRV: MOV @RKDA,@(SP) ;SAVE RKDA
4149 016944 010146 MOV R1,@(SP) ;GET DRIVE #
4150 016446 000241 CLC
4151 016950 006016 ROR (SP)
4152 016452 006016 ROR (SP)
4153 016454 006016 ROR (SP)
4154 016456 006016 ROR (SP)
4155 016460 012677 162444 MOV (SP)+,@RKDA ;ADDRESS THE DRIVE TO BE CHECKED
4156 016464 032777 010000 162424 BIT @DPL,@RKDS ;DRIVE POWER LOW?
4157 016472 001403 BEQ 18
4158 016474 004737 022222 JSR PC,RG48DR ;GET RKCS, ER, DS, DA AND DRIVE #
4159 ;FOR TYPING SERIAL NUMBER
4160 016400 104035 ERROR 35 ;DRIVE POWER LO, *NOTE 1 ABOVE
4161 016402 032777 002000 162406 18: BIT @DRU,@RKDS ;DRIVE
4162 016410 001403 BEQ 28
4163 016412 004737 022222 JSR PC,RG48DR ;GET RKCS, ER, DS, DA AND DRIVE #
4164 ;FOR TYPING SERIAL NUMBER
4165 016416 104036 ERROR 36 ;DRIVE UNSAFE BIT IS SET
4166 ;*NOTE 1 ABOVE
4167 016620 032777 000040 162370 28: BIT @WPS,@RKDS ;WRITE PROTECT SET?
4168 016926 001403 BEQ 38
4169 016630 004737 022222 JSR PC,RG48DR ;GET RKCS, ER, DS, DA AND DRIVE #
4170 ;FOR TYPING SERIAL NUMBER
4171 016634 104037 ERROR 37 ;WPS SET, CHECK WRTE PROTECT SWITCH ON DRIVE
4172 ;*NOTE 1 ABOVE
4173 016636 032777 000200 162352 38: BIT @DRY,@RKDS ;DRIVE READY CLEAR?
4174 016444 001004 BNE 48
4175 016446 004737 022222 JSR PC,RG48DR ;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 016554 000411 BR 58
4180
4181 016656 032777 012040 162332 48: BIT #12040,@RKDS ;ANY ERROR?
4182 016464 001005 BNE 58 ;YES
4183 016466 012677 162336 MOV (SP)+,@RKDA ;RESTORE RKDA
4184 016672 062716 000002 ADD #2,(SP) ;ADJUST RETURN ADDRESS
4185 016676 000207 RTS PC
4186
4187 016700 000137 016312 58: JMP DSELCT
    
```

```

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 #) = $MINUM
4194 ; 2) COMPLEMENT OF THE ABOVE WORD
4195 ;CALL: JSR R5,GENBUF
4196 ; X
4197 ; Y ;X IS THE WORD COUNT (2'S COMPLEMENT)
4198 ; Y IS THE STARTING ADDRESS OF THE
4199 ; MEMORY BUFFER.
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 18: MOV @RKDA,R2 ;GET THIS RANDOM SEED
4206 016720 010237 025756 MOV R2,RSDTL
4207 016724 010237 025754 MOV R2,RSDTL
4208 016730 005137 025754 COM RSDTL ;GET LO RANDOM SEED
4209
4210 016734 022703 177400 CMP #400,R3 ;IF THE BUFFER IS MORE THAN
4211 016740 003003 SGT 28 ;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 38 ;BEGINS WITH RANDOM DATA
4215
4216 016750 062703 000400 28: ADD #400,R3 ;WORDS GENERATED USING THAT
4217 016754 012705 177400 MOV #400,R5 ;SECTOR ADDRESS AS THE RANDOM
4218 ;SEED
4219 016760 010524 38: 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 016762 005205 INC R5 ;ALL DONE?
4223 016764 001427 BEQ 88
4224
4225 016766 004737 025576 48: JSR PC,@RAND ;GENERATE DATA WORDS
4226 016772 025754 RSDTL
4227 016774 012737 000002 017104 MOV #2,RCNT
4228 017002 013737 025756 017106 MOV RSDTL,RNUM
4229 017010 000406 BR 68
4230 017012 005337 017104 58: DEC RCNT
4231 017016 001763 BEQ 48
4232 017020 013737 025754 017106 MOV RSDTL,RNUM
4233 017026 005737 017106 68: TST RNUM
4234 017032 001767 BEQ 58
4235 017034 013724 017106 MOV RNUM,(R4)+ ;FILL THE BUFFER. DON'T USE
4236 017040 005205 INC R5 ;ALL DONE?
4237 017042 001351 BNE 48 ;NO
4238
4239 017044 005703 88: TST R3 ;ANY MORE DATA WORDS (FOR
4240 017046 001412 BEQ 108 ;REST OF THE SECTORS)?
4241 ;YES
4242 017050 010246 MOV R2,@(SP)
4243 017052 042716 177760 BIC #177760,(SP) ;ABSOLUTE DISK ADDRESS & ITS
    
```

```

4244 017A56 022726 000013      CMP      #13,(SP)+      ;COMPLEMENT) TO USE FOR
4245 017A62 011002              BNE      98             ;GENERATING DATA WORDS
4246 017A64 062702 000004      ADD      #4,R2         ;OF THE NEXT BLOCK
4247                                ;
4248 017A70 008202              98:     INC      R2     ;HI RANDOM SEED
4249 017A72 000712              BR       18
4250                                ;
4251 017A74 104415              108:    RESREG      ;RESTORE REGISTERS
4252 017A76 062705 000004      ADD      #4,R8         ;ADJUST RETURN ADDRESS
4253 017102 000305              RTS      R8            ;RETURN
4254                                ;
4255 017104 000000              RCNT:   0
4256 017106 000000              RNUM:   0
    
```

```

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,NC,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-R8
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'B COMP)
4269 017122 016305 000006      MOV      8(R3),R8     ;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 18,14,12 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      @DATA,R2     ;FORM THE ADDRESS OF DATA ERROR COUNT
4278                                ;FOR THIS DRIVE
4279 017146 012737 177764 001840      MOV      #14,RCOUNT   ;
4280 017154 011337 025756              MOV      (R3),RSDTH   ;CREATE RANDOM SEEDS TO
4281 017160 013737 025756 025754      MOV      RSDTH,RSDTL  ;BE USED FOR CHECKING DATA
4282 017166 005137 025754              COM      RSDTL
4283                                ;
4284 017172 022704 177400              18:     CMP      #400,R4    ;DATA IS CHECKED IN 1 SECTOR
4285 017176 003003              SGT      28           ;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       36
4289                                ;
4290 017206 062704 000400              28:     ADD      #400,R4
4291 017212 012703 177400              MOV      #400,R3
4292 017216 012500              38:     MOV      (R8)+,R0
4293                                ;SAVE THE FIRST WORD OF THE SECTOR,
4294                                ;FIRST WORD OF EVERY SECTOR IS A COUNT
4295                                ;(2'B COMP) INDICATING # OF WORDS ACTUALLY
4296                                ;WRITTEN IN THAT SECTOR
4297                                ;INCREMENT COUNT OF # OF WORDS (WRITTEN)
4298                                ;IN THE SECTOR
4299 017220 008200              INC      R0
4300                                ;INCREMENT COUNT OF DATA WORDS TO BE CHECKED
4301 017222 008203              INC      R3
4302 017224 001465              BEQ     148          ;BRANCH, IF DONE
4303                                ;
4304 017226 004737 025754 025754      48:     JSR      PC,@RAND ;GENERATE RANDOM DATA WORD
4305 017232 025754              RSDTL
4306 017234 012737 000002 017104      MOV      #2,RCNT
4307 017242 013737 025756 017106      MOV      RSDTH,RNUM
4308 017250 000406              BR       108
4309 017252 005337 017104              98:     DEC      RCNT
4310 017256 001763              BEQ     48
4311 017260 013737 025754 017106      MOV      RSDTL,RNUM
4312 017266 005737 017106              108:    TST      RNUM
4313 017272 001767              BEQ     98
4314                                ;
4315 017274 005700              TST      R0
    
```

```

4313 017774 001401      BFW      88
4314 017700 005200      INC      R0
4315
4316 017702 023715 017106      58:    CMP      RNUM,(R5)      ;EXPCTD WORD = RECVN WORD?
4317 017706 001431      BEQ      88              ;YFS
4318
4319 017710 005700      TST      R0
4320 017712 001005      HNE      88
4321 017714 008715      TBT      (R5)
4322 017716 001423      BEQ      88
4323 017720 005037 001164      CLR      $REG1
4324 017724 000403      BR       78
4325
4326 017726 013737 017106 001164 68:    MOV      RNUM,$REG1      ;SAVE EXPCTD DATA WORD
4327 017734 005212      INC      (R2)           ;INCRMT DATA EROR COUNT FOR THIS DRIVE
4328 017736 008737 001540      TBT      ECOUNT         ;STORE ONLY 12 (DEC) DATA ERRORS
4329 017742 001413      BEQ      88             ;IF MORE EXIT
4330 017744 010937 001162      MOV      R5,$REG0       ;SAVE ERROR BUS ADDRESS
4331 017750 011537 001166      MOV      (R5),$REG0     ;SAVE ERROR DATA WORD
4332 017754 010137 001170      MOV      R1,$REG3
4333 017760 004737 022226      JSR      PC,$TSDRV      ;SAVE DRIVE #, FOR TYPING SERIAL #
4334 017764 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 017766 005237 001540      INC      ECOUNT
4342
4343 017772 005725      88:    TBT      (R5)+
4344 017774 005203      INC      R3
4345 017776 001313      BNE      46             ;DONE CHECKING?
4346
4347 017700 005704      148:   TBT      R4
4348 017702 001427      BEQ      178            ;ANY MORE SECTOR BLOCKS
4349                                     ;TO CHECK? IF NOT, EXIT
4350 017704 010146      MOV      R1,=(SP)
4351
4352 017706 042716 177760      BIC      $177760,(SP)   ;GET THE NEW RANDOM SEEDS
4353 017712 022726 000013      CMP      $13,(SP)+     ;(ABSOLUTE DISK ADDRESS & ITS COMPLEMENT)
4354 017716 001002      BNE      156            ;TO USE FOR GENERATING DATA WORDS
4355 017720 062701 000004      ADD      $4,R1         ;OF THE NEXT BLOCK
4356
4357 017724 005201      158:   INC      R1
4358 017726 032777 000002 161564      BIT      %CSE,$RKRER   ;IF THERE WAS A CSE THEN CHECK
4359 017734 001403      BEQ      168            ;ONLY THOSE SECTORS THAT WERE READ
4360 017736 020177 161566      CMP      R1,$RKDA
4361 017742 001407      BEQ      176
4362 017744 010137 025756      MOV      R1,$RDTH
4363 017750 010137 025754      MOV      R1,$RDTL
4364 017754 005137 025754      COM      RDTL
4365 017760 000644      BR       18
4366 017762 104415      178:   RESREG
4367 017764 000207      RTS      PC             ;RESTORE R0=R5
    
```

```

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                                     ;*LSTBK WILL CONTAIN THE LAST BANK AS A 3AF
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                                     ;*      DETERMINED BY ROUTINE
4383
4384 017766 010046      $SIZE:  MOV      R0,=(SP)     ;SAVE R0 ON THE STACK
4385 017770 010146      MOV      R1,=(SP)     ;SAVE R1 ON THE STACK
4386 017772 010246      MOV      R2,=(SP)     ;SAVE R2 ON THE STACK
4387 017774 010346      MOV      R3,=(SP)     ;SAVE R3 ON THE STACK
4388 017776 013746 000004      MOV      $ERRVEC,=(SP) ;SAVE PRESENT ERROR VECTOR P6 & PC
4389 017702 013746 000006      MOV      $ERRVEC+2,=(SP)
4390 017706 010600      MOV      $P,R0        ;SAVE THE STACK POINTER
4391                                     ;SET THE ERRVEC P6 TO THE PRESENT P8
4392 017710 104400      TRAP
4393 017712 012637 000006      MOV      (SP)+,$ERRVEC+2 ;PUSH OLD P6W AND PC ON STACK
4394 017716 012701 003776      MOV      $3776,R1     ;SAVE THE P6W IN $ERRVEC+2
4395 017722 105727      TSTB    (PC)+         ;SETUP ADDRESS
4396 017724 000200      $KT11: .WORD    200     ;USE MEMORY MANAGEMENT?
4397 017726 100062      BPL     $CORE         ;SET TO USE MEMORY MANAGEMENT
4398 017730 012737 017666 000006      MOV      $KTINEX,$ERRVEC ;BR IF NO
4399 017736 005737 177572      TST     $SRO          ;SET FOR TIMEOUT
4400 017742 052737 100900 017524      BIS     $10000,$KT11  ;KT11 ARE YOU THERE?
4401 017750 005046      CLR     ~($P)         ;YES==SET KT11 KEY
4402 017752 012702 172340      MOV     $KIPAR0,R2    ;INITIALIZE FOR "PAR" LOADING
4403 017756 012703 000010      MOV     "$D8,R3       ;ADDRESS OF "FIRST" "PAR"
4404 017762 012762 077406 177740 18:   MOV     $77406,-40(R2) ;LOAD EIGHT "PAR,"S" AND EIGHT "PDR,"S"
4405 017770 011622      MOV     (SP),(R2)+    ;PDR = 4K, UP, READ/WRITE
4406 017772 062716 000200      ADD     $200,(SP)    ;LOAD "PAR"
4407 017776 077307      SOB     R3,18        ;UPDATE FOR NEXT "PAR"
4408 017800 012742 177600      MOV     $177600,=(R2) ;LOOP UNTIL ALL EIGHT ARE LOADED
4409 017804 005042      CLR     ~(R2)        ;SETUP KIPAR7 FOR I/O
4410 017806 012737 017624 000004      MOV     $2,$ERRVEC   ;SETUP KIPAR6 FOR TESTING
4411 017814 012737 000020 172516      MOV     $20,$SR3     ;CATCH TIMEOUT IF NO SR3
4412 017822 000401      BR      38          ;ENABLE 22 BIT MODE
4413 017824 022626      28:    CMP     (SP)+,(SP)+  ;THIS PDP-11 HAS A SR3 REGISTER
4414                                     ;CLEAN OFF THE STACK==NO SR3
4414 017826 005237 177572      38:    INC     $SRO        ;TURN ON MEMORY MANAGEMENT
4415 017832 012737 017656 000004      MOV     $KTOUT,$ERRVEC ;SET FOR TIME OUT
4416 017840 005737 143776      48:    TST     $143776     ;TRAP ON NON-EX-MEM
4417 017844 062712 000040      ADD     $4,(R2)      ;MAKE A 1K STEP
4418 017850 023712 172356      CMP     $KIPAR7,(R2) ;LAST ONE?
4419 017854 101371      BHI     48           ;NO==TRY IT
4420 017856 011202      $KTOUT: MOV     (R2),R2    ;GET LAST BANK+1
4421 017860 005037 177572      CLR     $SRO        ;TURN OFF MEMORY MANAGEMENT
4422 017864 000421      BP      $SIZEX
4423 017866 042737 100000 017524 $KTINEX: BIC     $100000,$KT11 ;KT11 NON=EXISTENT
    
```

```

4424 017674 012737 017724 000004 SCORE: MOV #SCROUT,@ERRVEC ;;SET FOR TIMEOUT
4425 017702 005002 CLN R2 ;;SET UP BANK
4426 017704 062701 004000 181 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 18 ;;NO=TRY AGAIN
4431 017724 162701 004000 SCROUT: SUB #4000,R1
4432 017730 162702 000040 SSIZER: SUB #40,R2 ;;DROP BACK
4433 017734 010006 MOV R0,SP ;;RESTORE THE STACK
4434 017736 012637 000006 MOV (SP)+,@ERRVEC+2 ;;RESTORE ERROR VECTOR
4435 017742 012637 000004 MOV (SP)+,@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 ;TYPD80
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 SM 13 IS SET.
4450
4451 017774 032777 020000 161136 TYPD80: BIT #SM13,@SWR ;;INHIBIT TYPEOUT?
4452 020002 001042 BNE 28 ;;YES
4453 020004 010346 MOV R3,-(SP)
4454 020006 010446 MOV R4,-(SP)
4455 020010 010846 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 KIPAR(R3),R4 ;;GET THE BASE PAGE ADDRESS FROM
4463 ;;KIPAR
4464 020030 005003 CLR R3 ;;ROTATE LEFT 6 TIMES (MULTIPLY
4465 020032 012706 177772 181 MOV #6,R5 ;;BY 100 OCTAL) TO GET THE
4466 020036 006104 ROL R4 ;;BASE SUB ADDRESS (PHYSICAL)
4467 020040 005206 INC R6 ;;R3 CONTAINS MSB=2 BITS
4468 020042 001375 BNE 18
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 020048 062604 ADD (SP)+,R4 ;;FORM THE ENTIRE PHYSICAL
4475 020054 005803 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 $REG0,-(SP) ;;PUSH POINTER TO PA ON STACK
4480 020072 004737 024464 JSR PC,$$SUPR8 ;;CONVERT THE 18 BIT BINARY
4481 ;;ADDRESS TO OCTAL ASCII NUMBERS ON RETURN
4482 ;;POINTER TO THE FIRST ASCII CHARACTERS
4483 ;;IS ON STACK
4484 020076 004737 038014 JSR PC,$$SUPR8 ;;TYPE OUT THE OCTAL 6 DIGIT
4485 ;;PHYSICAL ADDRESS.
4486
4487 020102 012606 MOV (SP)+,R6
4488 020104 012604 MOV (SP)+,R4
4489 020106 012603 MOV (SP)+,R3
4490
4491 020110 000207 281 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 @RKCS ;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,@RKDA ;DID RKDA INCREMENT CORRECTLY?
4509 020132 001485 BEQ COMRET ;YES
4510 020134 010237 001162 MOV R2,@REG0 ;GET EXPCTD RKDA
4511 020140 017737 161064 001164 MOV @RKDA,@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,@RKBA ;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 ERROR MESSAGE
4530
4531 020176 017737 161024 001164 MOV @RKBA,@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 020204 017746 161012 CHKMEX: MOV @RKCS,=(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 RKWC 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 @RKWC ;RKWC OVERFLOWED?
4549 020234 001424 BEQ COMRET ;YES
4550 020236 017737 160766 001162 MOV @RKDA,@REG0
4551 020244 017737 160754 001164 MOV @RKWC,@REG1
4552 020252 000207 RTS PC ;RETURN TO THE ERROR MESSAGE
4553
4554 ;CHKRWS
4555 ;THIS ROUTINE CHECKS IF R/W/S RDY BIT IN RKDS IS SET. IF IT IS NOT SET RETURN
4556 ;IS MADE TO THE ERROR MESSAGE FOLLOWING THE JSR CALL, IF IT IS, THE RETURN
4557 ;ADDRESS IS ADJUSTED TO SKIP OVER THE ERROR MESSAGE ON RETURN.
4558
4559 020254 032777 000100 160734 CHKRWS: BIT @RWS,@RKDS ;RWS RDY SET?
4560 020262 001011 BNE COMRET ;YES
4561 020264 004737 022032 JSR PC,GT4RG ;GET RKCS, ER, DS, DA
4562 020270 000207 RTS PC ;RETURN TO THE ERROR MESSAGE
4563
4564 ;CHKCRDY
4565 ;THIS ROUTINE CHECKS IF CONTROL READY BIT IN RKCS IS SET, IF IT IS NOT,
4566 ;RETURN IS MADE TO THE ERROR MESSAGE FOLLOWING THE JSR CALL, IF IT IS,
4567 ;RETURN ADDRESS IS ADJUSTED TO SKIP OVER THE ERROR MESSAGE.
4568
4569 020272 105777 160724 CHKCRDY: TSTB @RKCS ;CONTROL READY SET?
4570 020276 100403 BMI COMRET ;YES
4571 020300 004737 022032 JSR PC,GT4RG ;GET RKCS, ER, DS, DA
4572 020304 000207 RTS PC ;RETURN TO THE ERROR MESSAGE
4573
4574 COMRET: ADD #2,(SP) ;ADJUST RETURN ADDRESS TO SKIP OVER MESSAGE
4575 020312 000207 RTS PC
    
```

```

4579 ;THIS ROUTINE KEEPS A HISTORY OF THE COMMANDS THAT ARE BEING EXECUTED
4580 ;ON THE RK11. 'PRSPNC' 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 'PRSPNC' ARE PUSHED INTO 'PSTFNC'
4602 ;AND SAVED, BEFORE PUTTING INFO ABOUT THE PRESENT COMMAND IN 'PRSPNC'.
4603
4604 ;RO CONTAINS ADDRESS OF THE COMMAND KEY, AT THE TIME OF ENTRY.
4605
4606 020114 012737 100400 001512 DRCMND: MOV #BIT15+BIT8,QFNC
4607 020122 017746 160702 MOV SRDA,=(SP) ;SAVE DRIVE #
4608 020126 004737 022170 JSR PC,CROTLE
4609 020132 052637 001512 BIS (SP)+,QFNC
4610 020138 000424 BR P2
4611
4612 020140 012737 140000 001512 CRCMND: MOV #BIT15+BIT14,QFNC
4613 020146 000420 BR P2
4614
4615 020150 011037 001512 POSCMND: MOV (RO),QFNC
4616 020154 042737 177770 001512 BIC #177770,QFNC ;GET DRIVE NO.
4617 020162 052737 100200 001512 BIS #BIT15+BIT7,QFNC
4618 020170 000407 BR P2
4619
4620 020172 005037 001512 FNCMND: CLR QFNC
4621 020176 010046 P1: MOV RO,=(SP)
4622 020400 162716 001306 SUB #KEY,(SP)
4623 020404 042637 001512 BIS (SP)+,QFNC
4624
4625 020410 013737 001462 001464 P2: MOV PRSPNC,PSTFNC
4626 020416 013737 001512 001462 MOV QFNC,PRSPNC
4627 020424 000207 RTS PC
    
```

```

4628 ;HISTRY
4629 ;THIS ROUTINE TYPES OUT INFORMATION ABOUT THE FUNCTION THAT WAS
4630 ;BEING PERFORMED ON THE RK AT THE TIME OF ERROR AND THE FUNCTION
4631 ;THAT WAS PERFORMED JUST BEFORE THAT FUNCTION (WHICH LED TO
4632 ;THE ERROR). THIS ROUTINE IS CALLED WHEN AN ERROR OCCURS AND SW 12
4633 ;IS SET.
4634
4635 020426 010046 HISTRY: MOV RO,=(SP)
4636 020430 010146 MOV R1,=(SP)
4637
4638 020432 012700 001462 MOV #PRSPNC,RO
4639 020436 104401 020746 TYPE ,MH1
4640 020442 104401 020772 TYPE ,MH3
4641 020446 104401 020762 TYPE ,MH2
4642 020452 005710 66: TST (RO)
4643 020454 100053 BPL 36 ;READ, READ CHECK, WRITE, WRITE CHECK, SEEK
4644 020456 105710 TSTB (RO)
4645 020460 100427 BMI 28 ;POSITIONING (SEEK)
4646 020462 032710 040000 BIT #BIT14,(RO)
4647 020466 001014 BNE 18 ;CONTROL RESET
4648
4649 020470 104401 020476 TYPE ,668 ;TYPE ASCIZ STRING
4650 020474 000410 BR 668 ;GET OVER THE ASCIZ
4651 ;668: ,ASCIZ /DRESET ON DRV /
4652 020516 668: BR 78
4653 020516 000425 18:
4654 020520 668:
4655 020520 104401 020526 TYPE ,678 ;TYPE ASCIZ STRING
4656 020524 000404 BR 668 ;GET OVER THE ASCIZ
4657 020536 668: ,ASCIZ /CRESET/
4658 020536 000463 668: BR 68
4659 020540 668:
4660 020540 104401 020546 TYPE ,698 ;TYPE ASCIZ STRING
4661 020544 000412 BR 668 ;GET OVER THE ASCIZ
4662 020572 698: ,ASCIZ /POSITIONING DRIVE /
4663 020572 011046 78: MOV (RO),=(SP)
4664 020574 042716 177770 BIC #177770,(SP) ;TYPE DRIVE NO.
4665 020600 104401 TYPDC
4666 020602 000441 BR 48
4667
4668 020604 011001 38: MOV (RO),R1
4669 020606 016101 002032 MOV PCMND(R1),R1 ;GO TYPE OUT THE FUNCTION
4670
4671 020612 016104 000002 MOV 2(R1),R4 ;BEING PERFORMED
4672 020616 004737 021736 JSR PC,TYPFN
4673 020622 104401 020630 TYPE ,718 ;TYPE ASCIZ STRING
4674 020626 000403 BR 708 ;GET OVER THE ASCIZ
4675 ;718: ,ASCIZ <15><12>/DA=/
4676 708:
4677 020636 011146 MOV (R1),=(SP) ;TYPE OUT DISK ADDRESS
4678 020640 104402 TYPDC
4679 020642 104401 020650 TYPE ,738 ;TYPE ASCIZ STRING
4680 020646 000403 BR 728 ;GET OVER THE ASCIZ
    
```



```

4684      ;;738: ,ASCIZ / BA=/
4685      728:
4686 020456      MOV      6(R1),-(SP)
4687 020462 104402      TYPOC
4688 020464 104401 020672      TYPE      ,758      ;;TYPE ASCIZ STRING
4689 020470 000403      BR        748      ;;GET OVER THE ASCIZ
4690      ;;759: ,ASCIZ / WC=/
4691      748:
4692 020700      MOV      4(R1),-(SP)
4693 020704 104402      TYPOC
4694
4695 020706 020027 001464      48:  CHP      RO,#PSTFNC
4696 020712 001410      BEQ      58
4697 020714 005720      TST      (R0)+
4698 020716 104401 020746      TYPE      ,MH1
4699 020722 104401 020775      TYPE      ,MH4
4700 020726 104401 020762      TYPE      ,MH2
4701 020732 000647      BR        68
4702 020734 104401 001213      58:  TYPE      ,8CRLF
4703 020740 012601      MOV      (SP)+,R1
4704 020742 012600      MOV      (SP)+,R0
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
4711 020775 120      044522 051117 MH3:  ,ASCIZ /AT/
4712 021002 052080 000117 MH4:  ,ASCIZ /PRIOR TO/
4713      ,EVEN
    
```

```

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      28      ;FORM THE OFFSET FOR THE
4727 021022 062702 000004      18:  ADD      #4,R2      ;"WORDS XFERRED COUNTS"-
4728 021026 005301      DEC      R1      ;NWRTL, NRDL
4729 021030 001374      BNE      18
4730 021032 016500 000004      28:  MOV      4(R5),R0      ;GET WORD COUNT (RKWC) FROM
4731 021036 005400      NEG      R0      ;THE PARAMETER TABLE
4732
4733 021040 005777 160156      TST      8(RKWC)      ;ANY ERROR DURING THE XFER?
4734 021044 100004      BPL      38
4735
4736 021046 017703 160152      MOV      8(RKWC),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      38:  CMP      #2,R4      ;WRITE FUNCTION?
4741 021062 001005      BNE      58
4742
4743 021064 060062 001732      48:  ADD      R0,NWRTL(R2)      ;YES, ADD THE # OF WORDS
4744 021070 005562 001734      ADC      NWRTH(R2)      ;XFERRED (WRITE)
4745 021074 000404      BR        68      ;NOTE IT'S 2-WORD COUNT LO, HI
4746
4747 021076 060062 001772      58:  ADD      R0,NRDL(R2)      ;ADD THE # OF WORDS READ
4748      ;NOTE THAT WRT CHK,
4749      ;READ CHK ARE ALSO CONSIDERED
4750      ;TO BE "READ"
4751 021102 005562 001774      ADC      NRDH(R2)      ;CARRY OVER TO THE HI WORD
4752
4753 021106 012603      68:  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
    
```

```

4757 ;REPSTAT
4758 ;THIS ROUTINE REPORTS ERROR STATISTICS AND DATA-TRANSFER STATISTICS.
4759
4760 021116 104401 002377 REPSTA: TYPE ,MSG26
4761 021122 013700 001264 MOV DRVPRS,R0
4762 021126 012701 001254 MOV #PDR,R1
4763
4764 021132 104401 001213 18: TYPE ,@CRLF
4765 021136 112102 MOV# (R1)+,R2
4766 021140 042702 177770 BIC #177770,R2
4767 021144 010246 MOV R2,-(SP)
4768 021146 104403 TYPDS
4769 021150 003 ,BYTE 3
4770 021151 000 ,BYTE 0
4771 021152 104401 002662 TYPE ,BLNK3
4772
4773 021156 005004 CLR R4
4774 021160 010203 MOV R2,R3
4775 021162 001404 BEQ 38
4776 021164 062704 000004 28: ADD #4,R4
4777 021170 005303 DEC R3
4778 021172 001374 BNE 28
4779
4780 021174 104401 002664 38: TYPE ,BLNK31
4781 021190 010446 MOV R4,-(SP)
4782 021202 062716 001732 ADD #NRRL,(SP)
4783 021206 004737 024604 JSR PC,@DB2D
4784 021212 004737 025014 JSR PC,@UPRS
4785 021216 104401 002664 TYPE ,BLNK31
4786 021222 010446 MOV R4,-(SP)
4787 021224 062716 001772 ADD #NRDL,(SP)
4788 021230 004737 024604 JSR PC,@DB2D
4789 021234 004737 025014 JSR PC,@UPRS
4790
4791 021240 006302 ASL R2
4792
4793 021242 104401 002664 TYPE ,BLNK31
4794 021246 016246 001692 MOV C@ECN(R2),-(SP)
4795 021252 104405 TYPDS
4796
4797 021254 104401 002664 TYPE ,BLNK31
4798 021260 016246 001632 MOV W@ECN(R2),-(SP)
4799 021264 104405 TYPDS
4800
4801 021266 104401 002664 TYPE ,BLNK31
4802 021272 016246 001712 MOV D@TER(R2),-(SP)
4803 021276 042716 100000 BIC #100000,(SP) ;DONT TYPE A NEGATIVE NO.
4804 021282 104405 TYPDS
4805
4806 021284 104401 002664 TYPE ,BLNK31
4807 021290 016246 001562 MOV H@ECN(R2),-(SP)
4808 021294 104405 TYPDS
4809
4810 021296 005300 DEC R0 ;FINISHED WITH THE DRIVES ?
4811 021298 001304 BNE 18 ;BR IF NOT
4812 021222 104401 002474 TYPE ,MSG26A ;REST OF SUMMARY MESSAGE
    
```

```

4813 021226 013700 001264 MOV DRVPRS,R0 ;NUMBER OF DRIVES
4814 021232 012701 001254 MOV #PDR,R1 ;'DRIVES PRESENT' TABLE ADDRESS
4815 021236 104401 001213 48: TYPE ,@CRLF ;CR-LF
4816 021242 112102 MOV# (R1)+,R2 ;DRIVE ADDRESS
4817 021244 042702 177770 BIC #177770,R2 ;LEAVE ONLY DRIVE NUMBER
4818 021250 010246 MOV R2,-(SP) ;PUT ON STACK FOR TYPEOUT
4819 021252 104403 TYPDS ;TYPE IT IN OCTAL
4820 021254 003 ,BYTE 3 ;TYPE 3 CHARACTERS
4821 021255 000 ,BYTE 0 ;SUPPRESS LEADING ZEROS
4822 021256 104401 002662 TYPE ,BLNK3 ;3 BLANKS
4823 021262 006302 ASL R2 ;CONVERT TO A WORD TABLE INDEX
4824 021264 104401 002664 TYPE ,BLNK31
4825 021270 016246 001602 MOV @ECN(R2),-(SP)
4826 021274 104405 TYPDS
4827
4828
4829 021276 104401 002664 TYPE ,BLNK31
4830 021402 016246 001672 MOV @ORT(R2),-(SP)
4831 021406 104405 TYPDS
4832
4833 021410 006202 ASR R2
4834 021412 104401 002664 TYPE ,BLNK31
4835 021416 116246 001622 MOV @IN(R2),-(SP)
4836 021422 104405 TYPDS
4837
4838 021424 005300 DEC R0
4839 021426 001343 BNE 48
4840 021430 004737 026660 JSR PC,@TIMTYP ;TYPE THE TIME
4841 021434 000207 RTS PC
    
```

```

4842 ;STATUS
4843 ;THIS ROUTINE IS NORMALLY ENTERED WHEN THE PROGRAM IS WAITING FOR THE
4844 ;CONTROLLER TO FINISH WHAT IT IS DOING, THERE ARE TWO DOUBLE PRECISION
4845 ;COUNTS KEPT IN THIS ROUTINE.
4846 ;CICNT,CICNT1
4847 ;THIS COUNT KEEPS TRACK OF HOW LONG THE CONTROLLER HAS BEEN BUSY AFTER
4848 ;A COMMAND WAS INITIATED, THE CONTROLLER SHOULD FINISH WHATEVER IT IS DOING
4849 ;BEFORE THIS COUNT EXPIRES. IF IT DOES NOT, THERE IS AN ERROR CONDITION AND
4850 ;IT IS SO REPORTED.
4851
4852 ;QSCNT
4853 ;THIS COUNT IS INITIALIZED EVERY TIME 8 COMMANDS ARE GENERATED, THE COUNT
4854 ;IS INCREMENTED EVERY TIME THIS ROUTINE IS ENTERED, ALL THE 8 COMMANDS
4855 ;SHOULD BE DONE BEFORE THIS COUNT EXPIRES, IF THEY DO NOT AN ERROR CONDITION
4856 ;IS REPORTED, THIS COUNT HAS BEEN KEPT PRIMARILY TO INSURE THAT THE PROGRAM
4857 ;DOES NOT GET CAUGHT IN AN INDEFINITE LOOP, BECAUSE OF AN ERROR CONDITION.
4858
4859 021436 005046          STATUS: CLR  -(SP)          ;DROP PRIORITY AND WAIT FOR INT
4860 021440 012746 021446  MOV  #18,-(SP)        ;RETURN FOR RTI
4861 021444 000002          RTI
4862
4863 ;NOTE THAT THE INTERRUPTS ARE ALLOWED ONLY
4864 ;AT CERTAIN PLACES IN THE PROGRAM, BECAUSE
4865 ;IT MAKES TROUBLESHOOTING OF FAILURES EASY.
4866 ;OTHER PLACES WHERE INTERRUPTS ARE ALLOWED
4867 ;TO TAKE PLACE:
4868 ;'CHFAFN', FIRST INTERRUPT AFTER ISSUING
4869 ;A SEEK FUNCTION.
4870 021446 005237 001460  18:  INC  QSCNT
4871 021452 001456          BEQ  QEROR
4872 021454 105777 157542  TSTB @RKC6
4873 021460 100514          BMI  CNOBSY
4874
4875 021462 005037 001466          CLR  CICNT
4876 021466 012737 177761 001470  MOV  #-17,CICNT1
4877
4878 021474 105737 001534  CBSY: TSTB INTFLG          ;FOR A NON-SEEK COMMAND;
4879 ;INTFLG= BIT 7 IS SET, BITS 0-3 CONTAIN
4880 ;OFFSET TO THE COMMAND KEY (FROM KEY),
4881 ;FOR WHICH THIS INTERRUPT IS EXPCTD.
4882 ;WHEN THE INTERRUPT OCCURS & 'INTHND' IS
4883 ;ENTERED 'INTFLG' IS CLEARED.
4884 021500 001507          BEQ  SEXIT
4885 021502 005237 001466          INC  CICNT
4886 021506 001372          BNE  CBSY
4887 021510 005237 001470          INC  CICNT1
4888 021514 001367          BNE  CBSY
4889
4890 ;TIMED OUT WHILE WAITING FOR THE INTERRUPT.
4891 ;ONE OF THE COMMANDS DID NOT INTERRUPT
4892 021516 113700 001534  NIEROR: MOV  INTFLG,R0
4893 021522 042700 177760          BIC  $177760,R0
4894 021526 010003          MOV  R0,R3
4895 021530 062700 001306          ADD  $KEY,R0
4896 021534 011037 001172          MOV  (R0),$REG4
4897 021540 042737 177770 001172  BIC  $177770,$REG4
  
```

```

4898 021546 013737 001172 001250  MOV  $REG4,$RDRV          ;GET DRIVE #, FOR TYPING SERIAL #
4899
4900 021554 104421 002245          TYPMSG ,MSG15          ;PRINT 'DRIVE # DIDN'T INTERRUPT AFTER'
4901 021560 016305 002032          MOV  PC,MND(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  26          ;COMMAND TYPED OUT IN ERROR MESSAGE DID
4906 ;NOT INTERRUPT ON COMPLETION.
4907 021602 052710 104000          BIS  $BIT15+$BIT11,(R0) ;INDICATE THAT FUNCTION IS ABORTED
4908 021606 000444          BR   SEXIT
4909
4910
4911 021610 005037 001460          QEROR: CLR  QSCNT          ;REESTABLISH COUNT
4912 021614 004737 022032          JSR  PC,GT4RG
4913 021620 104026          ERROR  26
4914 ;ALL 8 COMMANDS SHOULD BE DONE BY NOW, TIMED
4915 ;OUT, THE PROGRAM IS WAITING FOR ONE OF THE
4916 ;COMMANDS IN THE Q TO BE FINISHED AND THIS
4917 ;DID NOT HAPPEN OR FOR SOME OTHER REASON THE
4918 ;'FINISHED' FLAG (BIT 15) OF ONE OF THE 8
4919 ;COMMAND KEYS WAS NOT SET, VARIOUS FLAGS 'POS'(-7)
4920 ;'BUSY'(-7), 'KEY'(-8) CONTAIN INFORMATION
4921 ;ABOUT THE STATUS OF THE SYSTEM.
4922 021622 032777 020000 157310  BIT  $6W13,$6WR          ;INHIBIT TYPEOUT?
4923 021630 001024          BNE  28          ;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          18:  TYPE ,%CRLF
4929 021656 012046          MOV  (R0)+,-(SP)        ;TYPE OUT CONTENTS OF ALL KEYS
4930 021660 104402          TYPDC          ;KEY=KEYS
4931 021662 104401 002662          TYPE ,BLNK$3
4932 021666 005046          CLR  -(SP)
4933 021670 112116          MOV  (R1)+,(SP)        ;TYPE OUT CONTENTS OF ALL BUSY FLAGS
4934 021672 104403          TYPDS          ;BUSY=BUSY?
4935 021674 003          ,BYTE 3
4936 021675 000          ,BYTE 0
4937 021676 005202          INC  R2
4938 021700 001364          BNE  18          ;DONE?
4939 ;NO
4940 021702 004737 016006          28:  JSR  PC,CLRERR        ;MAKE SURE THERE IS NO HEAD MOVEMENT ON
4941 ;ANY DRIVE & THEN DO CONTROL RESET
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 187174 TYPFN: HIT      SW13,SWWP ;INHIBIT TYPEOUT?
4958 021744 001031 000000 000000 BNE     SW      ;YES
4959 021746 020427 000002 000002 CMP     R4,R4  ;WRITE?
4960 021752 001002 000000 000000 BNE     SW      ;WRITE?
4961 021754 104401 002133 000000 TYPE   ,M8G6
4962 021760 022704 000004 000004 16:    CMP     R4,R4  ;READ?
4963 021764 001002 000000 000000 BNE     SW
4964 021766 104401 002141 000000 TYPE   ,M8G7
4965 021772 022704 000012 000012 28:    CMP     R4,R4  ;READ CHECK?
4966 021776 001002 000000 000000 BNE     SW
4967 022000 104401 002156 000000 TYPE   ,M8G9
4968 022004 022704 000006 000006 38:    CMP     R4,R4  ;WRITE CHECK?
4969 022010 001002 000000 000000 BNE     SW
4970 022012 104401 002146 000000 TYPE   ,M8G8
4971 022016 022704 000010 000010 48:    CMP     R4,R4  ;SEEK?
4972 022022 001002 000000 000000 BNE     SW
4973 022024 104401 002201 000000 TYPE   ,M8G11
4974 022030 000207 000000 000000 58:    RTS     PC
    
```

```

4975 ;GT4RG
4976 ;GET CONTENTS OF RKCS, RKER, RKDS, RKDA
4977
4978 022032 017737 157172 001170 GT4RG: MOV     BRKDA,REG3
4979 022040 017737 157156 001162 GT4RG: MOV     BRKCS,REG0
4980 022046 017737 157146 001164 MOV     BRKER,REG1
4981 022054 017737 157136 001166 MOV     BRKDS,REG2
4982 022062 000207 000000 000000 RTS     PC
4983
4984
4985 ;GETINF
4986 ;THIS ROUTINE GETS CONTENTS OF RKCS, RKER, RKDS, THEN IT BREAKS DOWN THE
4987 ;CONTENTS OF RKDA INTO ITS COMPONENTS: CYLINDER, SECTOR, SURFACE AND DRIVE
4988 ;NUMBER.
4989
4990 022064 004737 022040 000000 GETINF: JBR     PC,GT4RG
4991 022070 010046 000000 000000 MOV     R0,-(SP)
4992 022072 010146 000000 000000 MOV     R1,-(SP)
4993 022074 010246 000000 000000 MOV     R2,-(SP)
4994 022076 012700 001200 001200 MOV     ,BRKDS+2,R0
4995 022102 017701 157122 000000 MOV     BRKDA,R1
4996 022106 010102 000000 000000 MOV     R1,R2
4997 022110 042702 177760 000000 BIC     $177760,R2
4998 022114 010240 000000 000000 MOV     R2,-(R0)
4999 022116 006201 000000 000000 ABR     R1
5000 022120 006201 000000 000000 ABR     R1
5001 022122 006201 000000 000000 ABR     R1
5002 022124 006201 000000 000000 ABR     R1
5003 022126 010102 000000 000000 MOV     R1,R2
5004 022130 042702 177776 000000 BIC     $177776,R2
5005 022134 010240 000000 000000 MOV     R2,-(R0)
5006 022136 006201 000000 000000 ABR     R1
5007 022140 010102 000000 000000 MOV     R1,R2
5008 022142 042702 177400 000000 BIC     $177400,R2
5009 022146 010240 000000 000000 MOV     R2,-(R0)
5010 022150 000301 000000 000000 SWAB   R1
5011 022152 042701 177770 000000 BIC     $177770,R1
5012 022156 010140 000000 000000 MOV     R1,-(R0)
5013 022160 012602 000000 000000 MOV     (SP)+,R2
5014 022162 012601 000000 000000 MOV     (SP)+,R1
5015 022164 012600 000000 000000 MOV     (SP)+,R0
5016 022166 000207 000000 000000 RTS     PC
    
```

```

5017 ;CROTLF
5018 ;CALL: MOV #NO,=(SP) ;PUSH NO. TO BE ROTATED ON STACK
5019 ; JSR PC,CROTLF
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 CROTLF: 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 RG4SDRV: 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 022170 JSR PC,CROTLF
5047 022236 012637 001250 MOV (SP)+,SRDRV ;SAVE THE DRIVE #
5048 022242 000207 RTS PC
    
```

```

5049 ;SHTTL 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 TIMEOUT
5055 022250 013777 001502 156752 MOV @DRV,@RKDA
5056 022256 012777 000015 156736 MOV #15,@RKCS
5057 022264 104417 CON,RDY
5058 022266 032777 000100 156722 18: BIT #100,@RKDS ;DID R/W/S RDY SET?
5059 022274 001026 BNE 26 ;YES
5060 022276 012746 177760 MOV #-20,=(SP) ;NO, WAIT FOR R/W/S
5061 022282 005216 INC (SP)
5062 022284 001376 BNE ,=2
5063 022286 005726 TST (SP)+
5064 022290 005237 022354 INC TIMEOUT
5065 022294 001364 BNE 18
5066 022296 032777 020000 156614 BIT #SW13,@SWR ;INHIBIT TYPEOUT?
5067 022298 001012 BNE 26 ;YES
5068 022300 104401 001213 TYPE ,@CPLF ;TIMED OUT, R/W/S RDY DID NOT SET
5069 022302 104401 027746 TYPE ,EM4 ;REPORT ERROR
5070 022304 104401 002206 TYPE ,MSG12
5071 022306 011646 MOV (SP),=(SP)
5072 022308 162716 000002 SUB #2,(SP)
5073 022310 104402 TYP0C
5074 022312 000002 28: RTI
5075 022314 000000 TIMEOUT: 0
    
```

```

5076          .SBTTL CON,RESET - CONTROL RESET ROUTINE
5077          ;CON,RESET
5078          ;CONTROL RESET ROUTINE
5079          ;CON,RDY
5080          ;CONTROL READY ROUTINE
5081
5082 022456 012777 000001 156636 CN,RST: MOV    #1,0RKCS
5083 022464 005037 001472 CN,RDY: CLR    TIMER
5084 022470 105777 156626 1$: TSTB    0RKCS      ;DID CONTROL RDY SET?
5085 022474 100451 BMI     2$      ;YES
5086 022476 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,@SWR ;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 104401 TYP0C  TYPE
5098 022442 104401 022450 TYPE   ,65$      ;TYPE ASCIZ STRING
5099 022446 000421 BR     64$      ;GET OVER THE ASCIZ
5100          ;65$: .ASCIZ <15><12>/CONTROLLER NOT READY = 0RKCS=/
5101 022412 64$: 64$: .ASCIZ
5102 022412 017746 156504 MOV    0RKCS,-(SP)
5103 022416 104402 TYP0C  TYPE
5104 022420 000002 2$: 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 022422 032777 020000 156410 TY,MSG: BIT    #SW13,@SWR ;INHIBIT TYPEOUT?
5113 022430 001005 BNE    2$      ;YES
5114 022432 017637 000000 022542 MOV    0(SP),1$ ;GET POINTER TO ASCII STRING
5115 022440 104401 TYPE   TYPE
5116 022442 000000 1$: .WORD    0
5117
5118 022444 062716 000002 2$: ADD    #2,(SP) ;ADJUST RETURN ADDRESS TO SKIP OVER POINTER
5119 022450 000002 RTI
    
```

```

5120          ,SBTTL KWSRVE = KW11L CLOCK SERVICE ROUTINE
5121          ;THIS ROUTINE SERVICES THE INTERRUPT FROM THE KW11L 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      16          ;OVERFLOWED?
5130 022560 000002              RTI
5131 022562 012737 177704 001560 18:  MOV      #-60,;KWCOUNT  ;RESET 60 CPS COUNT
5132 022570 005237 001556      INC      KWSEC      ;COUNT SECONDS
5133 022574 001401              BEQ      26          ;OVERFLOWED?
5134 022576 000002              RTI          ;RETURN
5135 022600 012737 177704 001556 28:  MOV      #-60,;KWSEC   ;RESET "SECONDS" COUNT
5136 022606 005237 001554      INC      KWMIN     ;COUNT MINUTES
5137 022612 001005              BNE     36          ;OVERFLOWED?
5138 022614 012737 177704 001554  MOV      #-60,;KWMIN   ;RESET "MINUTES" COUNT
5139 022622 005237 001552      INC      KWHR     ;COUNT HOURS
5140
5141 022626 000002              38:  RTI          ;RETURN
5142
5143
5144          ;WATIME
5145          ;ROUTINE PROVIDES SOME WAITING TIME.
5146
5147 022630 013746 022652      WATIME: MOV      28, -(SP)  ;COUNTER VALUE
5148 022634 005237 022654      18:  INC      36          ;COUNT
5149 022640 001375              BNE     18          ;HANG IN THERE UNTIL COUNT WRAPS AROUND
5150 022642 005216              INC     (SP)       ;COUNT AGAIN
5151 022644 001373              BNE     18          ;GO THROUGH MINOR LOOP AGAIN
5152 022646 005726              TST    (SP)+      ;RESTORE THE STACK POINTER
5153 022650 000207              RTS     PC         ;RETURN
5154 022652 177730              28:  ,WORD 177730   ;VALUE FOR APPROX 15 SEC DELAY
5155 022654 000000              38:  ,WORD 0        ;"MINOR" LOOP COUNTER
  
```

```

5156          ;CHPDRS
5157          ;THIS ROUTINE CHECKS IF THERE ANY DRIVES PRESENT (ON LINE), IF THERE
5158          ;ARE, A RETURN IS MADE. IF THERE ARE NONE PRESENT, A MESSAGE IS PRINTED OUT.
5159          ;THE STACK POINTER IS RE-INITIATED TO 1100 AND CONTROL IS TRANSFERRED
5160          ;TO THE END OF PASS ROUTINE, $EOP, BEFORE PASSING CONTROL TO $EOP, SOME
5161          ;TIME IS KILLED (WATIME), THIS IS DONE TO KEEP THE NUMBER OF MESSAGES
5162          ;(END OF PASS #X) TO A SMALL AMOUNT.
5163
5164 022656 005737 001264      CHPDRS: TST     DRVPRS   ;ANY DRIVES PRESENT?
5165 022662 001401              BEQ     18          ;NO
5166 022664 000207              RTS     PC         ;YES, EXIT
5167 022666 104401 002225      18:  TYPE  ,MSG14     ;NO, GIVE A MESSAGE
5168 022672 004737 022630      JSR    PC,WATIME  ;KILL SOME TIME
5169 022676 012706 001100      MOV    #STACK,SP  ;REINITIALIZE STACK
5170 022702 000400              BR     $EOP       ;GO TO END OF PASS ROUTINE
5171
  
```

```

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 THERE'S A MONITOR GO TO IT
5178                                ;*IF THERE ISN'T JUMP TO BEGNEX
5179
5180                                $EOP:
5181                                SCOPE
5182                                CLR $STNM                ;;ZERO THE TEST NUMBER
5183                                INC $PASS                ;;INCREMENT THE PASS NUMBER
5184                                BIC #10000,$PASS        ;;DON'T ALLOW A NEG. NUMBER
5185                                DEC (PC)+                ;;LOOP?
5186                                $EOPCT: ,WORD 1
5187                                BGT $DOAGN                ;;YES
5188                                MOV (PC)+,@(PC)+        ;;PSTORE COUNTER
5189                                $ENDCT: ,WORD 1
5190                                $EOPCT
5191                                $GET42: MOV @#42,R0        ;;GET MONITOR ADDRESS
5192                                REQ $DOAGN                ;;BRANCH IF NO MONITOR
5193                                RESET                    ;;CLEAR THE WORLD
5194                                $ENDAD: JSR PC,(R0)        ;;GO TO MONITOR
5195                                NOP                        ;;SAVE ROOM
5196                                NOP                        ;;FOR
5197                                NOP                        ;;ACT11
5198
5199                                $DOAGN: JMP @(PC)+        ;;RETURN
5200                                $RTNAD: ,WORD BEGNEX
  
```

```

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 150144                                TSTB @#TKS                ;;CHAR THERE?
5214                                023000 100071                                BPL 15$                ;;IF NO, DON'T WAIT AROUND
5215                                023002 117746 150140                                MOVB @#TKB,@(SP)                ;;SAVE THE CHAR
5216                                023006 042716 177600                                BIC #C177,(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                                CMPR $AUTOB,#1                ;;ARE WE RUNNING IN AUTO-MODE?
5220                                023026 001456                                REQ 15$                ;;BRANCH IF YES
5221
5222                                023030 104401 023511                                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 , $MNEW                ;;PROMPT FOR NEW SWR
5227                                023052 005046 190: CLR @(SP)                ;;CLEAR COUNTER
5228                                023054 005046                                CLR @(SP)                ;;THE NEW SWR
5229                                023056 105777 150062 78: TSTB @#TKS                ;;CHAR THERE?
5230                                023062 100375                                HPL 78                        ;;IF NOT TRY AGAIN
5231
5232                                023064 117746 150056                                MOVB @#TKB,@(SP)                ;;PICK UP CHAR
5233                                023070 042716 177600                                BIC #C177,(SP)                ;;MAKE IT 7-BIT ASCII
5234
5235
5236
5237                                023074 021627 000025 98: CMP (SP),#25                ;;IS IT A CONTROL-U?
5238                                023100 001005                                BNE 10$                ;;BRANCH IF NOT
5239                                023102 104401 023504                                TYPE , $CNTLU                ;;YES, ECHO CONTROL-U ("U")
5240                                023106 062706 000006 206: ADD #6,SP                ;;IGNORE PREVIOUS INPUT
5241                                023112 000757                                BR 19$                ;;LET'S TRY IT AGAIN
5242
5243
5244                                023114 021627 000015 108: 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
5248                                023126 001403                                BEQ 11$                ;;BRANCH IF YES
5249                                023130 016677 000002 156002                                MOV 2(SP),@SWR                ;;SAVE NEW SWR
5250                                023136 062706 000006 118: ADD #6,SP                ;;CLEAR UP STACK
5251                                023142 104401 001213 148: TYPE , $CRLF                ;;ECHO <CR> AND <LF>
5252                                023146 123727 001135 000001                                CMPB $INTAG,#1                ;;RE-ENABLE TTY KBD INTERRUPTS?
5253                                023154 001003                                BNE 15$                ;;BRANCH IF NOT
5254                                023156 012777 000100 155760                                MOV #100,@#TKS                ;;RE-ENABLE TTY KBD INTERRUPTS
5255                                023164 000002 158: RTI                        ;;RETURN
5256                                023166 004737 024414 166: JSR PC,$TYPEPC                ;;ECHO CHAR
5257                                023172 021627 000006                                CMP (SP),#60                ;;CHAR < 0?
  
```



```

5277 023176 022420          BLT      188          ;;BRANCH IF YES
5278 023100 021627 000067    CMP      (SP),#67    ;;CHAR > ??
5279 023104 013015          BGT      188          ;;BRANCH IF YES
5280 023106 042726 000060    HIC      #60,(SP)+  ;;STRIP-OFF ASCII
5281 023112 005766 000002    TST      2(SP)      ;;IS THIS THE FIRST CHAR
5282 023116 001403          BEQ      176          ;;BRANCH IF YES
5283 023120 006316          ASL      (SP)       ;;NO, SHIFT PRESENT
5284 023122 006316          ASL      (SP)       ;; CHAR OVER TO MAKE
5285 023124 006316          ASL      (SP)       ;; ROOM FOR NEW ONE.
5286 023126 005266 000002    178: INC      2(SP)    ;;KEEP COUNT OF CHAR
5287 023132 056616 177776    BIS      -2(SP),(SP) ;;SET IN NEW CHAR
5288 023136 000707          BR       78          ;;GET THE NEXT ONE
5289 023140 104401 001212    188: TYPE    #QUES    ;;TYPE ?<CR><LF>
5290 023144 000720          BR       208        ;;SIMULATE CONTROL-U
5291          ,DSABL  LSB
5292
5293          ;;*****
5294          ;;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
5295          ;*CALL:
5296          ;*
5297          ;*      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
5298          ;*      RETURN HERE    ;;CHARACTER IS ON THE STACK
5299          ;*
5300          ;*
5301          ;*
5302          ;*
5303          ;*
5304          ;*
5305          ;*
5306          ;*
5307          ;*
5308          ;*
5309          ;*
5310          ;*
5311          ;*
5312          ;*
5313          ;*
5314          ;*
5315          ;*
5316          ;*
5317          ;*
5318          ;*
5319          ;*
5320          ;*
5321          ;*
5322          ;*
5323          ;*
5324          ;*
5325          ;*
5326          ;*
5327          ;*
5328          ;*
5329          ;*
5330          ;*
5331          ;*
5332          ;*
5333          ;*
5334          ;*
5335          ;*
5336          ;*
5337          ;*
5338          ;*
5339          ;*
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          ;*
5378          ;*
5379          ;*
5380          ;*
5381          ;*
5382          ;*
5383          ;*
5384          ;*
5385          ;*
5386          ;*
5387          ;*
5388          ;*
5389          ;*
5390          ;*
5391          ;*
5392          ;*
5393          ;*
5394          ;*
5395          ;*
5396          ;*
5397          ;*
5398          ;*
5399          ;*
5400          ;*
5401          ;*
5402          ;*
5403          ;*
5404          ;*
5405          ;*
5406          ;*
5407          ;*
5408          ;*
5409          ;*
5410          ;*
5411          ;*
5412          ;*
5413          ;*
5414          ;*
5415          ;*
5416          ;*
5417          ;*
5418          ;*
5419          ;*
5420          ;*
5421          ;*
5422          ;*
5423          ;*
5424          ;*
5425          ;*
5426          ;*
5427          ;*
5428          ;*
5429          ;*
5430          ;*
5431          ;*
5432          ;*
5433          ;*
5434          ;*
5435          ;*
5436          ;*
5437          ;*
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          ;*
5494          ;*
5495          ;*
5496          ;*
5497          ;*
5498          ;*
5499          ;*
5500          ;*
5501          ;*
5502          ;*
5503          ;*
5504          ;*
5505          ;*
5506          ;*
5507          ;*
5508          ;*
5509          ;*
5510          ;*
5511          ;*
5512          ;*
5513          ;*
5514          ;*
5515          ;*
5516          ;*
5517          ;*
5518          ;*
5519          ;*
5520          ;*
5521          ;*
5522          ;*
5523          ;*
5524          ;*
5525          ;*
5526          ;*
5527          ;*
5528          ;*
5529          ;*
5530          ;*
5531          ;*
5532          ;*
5533          ;*
5534          ;*
5535          ;*
5536          ;*
5537          ;*
5538          ;*
5539          ;*
5540          ;*
5541          ;*
5542          ;*
5543          ;*
5544          ;*
5545          ;*
5546          ;*
5547          ;*
5548          ;*
5549          ;*
5550          ;*
5551          ;*
5552          ;*
5553          ;*
5554          ;*
5555          ;*
5556          ;*
5557          ;*
5558          ;*
5559          ;*
5560          ;*
5561          ;*
5562          ;*
5563          ;*
5564          ;*
5565          ;*
5566          ;*
5567          ;*
5568          ;*
5569          ;*
5570          ;*
5571          ;*
5572          ;*
5573          ;*
5574          ;*
5575          ;*
5576          ;*
5577          ;*
5578          ;*
5579          ;*
5580          ;*
5581          ;*
5582          ;*
5583          ;*
5584          ;*
5585          ;*
5586          ;*
5587          ;*
5588          ;*
5589          ;*
5590          ;*
5591          ;*
5592          ;*
5593          ;*
5594          ;*
5595          ;*
5596          ;*
5597          ;*
5598          ;*
5599          ;*
5600          ;*
5601          ;*
5602          ;*
5603          ;*
5604          ;*
5605          ;*
5606          ;*
5607          ;*
5608          ;*
5609          ;*
5610          ;*
5611          ;*
5612          ;*
5613          ;*
5614          ;*
5615          ;*
5616          ;*
5617          ;*
5618          ;*
5619          ;*
5620          ;*
5621          ;*
5622          ;*
5623          ;*
5624          ;*
5625          ;*
5626          ;*
5627          ;*
5628          ;*
5629          ;*
5630          ;*
5631          ;*
5632          ;*
5633          ;*
5634          ;*
5635          ;*
5636          ;*
5637          ;*
5638          ;*
5639          ;*
5640          ;*
5641          ;*
5642          ;*
5643          ;*
5644          ;*
5645          ;*
5646          ;*
5647          ;*
5648          ;*
5649          ;*
5650          ;*
5651          ;*
5652          ;*
5653          ;*
5654          ;*
5655          ;*
5656          ;*
5657          ;*
5658          ;*
5659          ;*
5660          ;*
5661          ;*
5662          ;*
5663          ;*
5664          ;*
5665          ;*
5666          ;*
5667          ;*
5668          ;*
5669          ;*
5670          ;*
5671          ;*
5672          ;*
5673          ;*
5674          ;*
5675          ;*
5676          ;*
5677          ;*
5678          ;*
5679          ;*
5680          ;*
5681          ;*
5682          ;*
5683          ;*
5684          ;*
5685          ;*
5686          ;*
5687          ;*
5688          ;*
5689          ;*
5690          ;*
5691          ;*
5692          ;*
5693          ;*
5694          ;*
5695          ;*
5696          ;*
5697          ;*
5698          ;*
5699          ;*
5700          ;*
5701          ;*
5702          ;*
5703          ;*
5704          ;*
5705          ;*
5706          ;*
5707          ;*
5708          ;*
5709          ;*
5710          ;*
5711          ;*
5712          ;*
5713          ;*
5714          ;*
5715          ;*
5716          ;*
5717          ;*
5718          ;*
5719          ;*
5720          ;*
5721          ;*
5722          ;*
5723          ;*
5724          ;*
5725          ;*
5726          ;*
5727          ;*
5728          ;*
5729          ;*
5730          ;*
5731          ;*
5732          ;*
5733          ;*
5734          ;*
5735          ;*
5736          ;*
5737          ;*
5738          ;*
5739          ;*
5740          ;*
5741          ;*
5742          ;*
5743          ;*
5744          ;*
5745          ;*
5746          ;*
5747          ;*
5748          ;*
5749          ;*
5750          ;*
5751          ;*
5752          ;*
5753          ;*
5754          ;*
5755          ;*
5756          ;*
5757          ;*
5758          ;*
5759          ;*
5760          ;*
5761          ;*
5762          ;*
5763          ;*
5764          ;*
5765          ;*
5766          ;*
5767          ;*
5768          ;*
5769          ;*
5770          ;*
5771          ;*
5772          ;*
5773          ;*
5774          ;*
5775          ;*
5776          ;*
5777          ;*
5778          ;*
5779          ;*
5780          ;*
5781          ;*
5782          ;*
5783          ;*
5784          ;*
5785          ;*
5786          ;*
5787          ;*
5788          ;*
5789          ;*
5790          ;*
5791          ;*
5792          ;*
5793          ;*
5794          ;*
5795          ;*
5796          ;*
5797          ;*
5798          ;*
5799          ;*
5800          ;*
5801          ;*
5802          ;*
5803          ;*
5804          ;*
5805          ;*
5806          ;*
5807          ;*
5808          ;*
5809          ;*
5810          ;*
5811          ;*
5812          ;*
5813          ;*
5814          ;*
5815          ;*
5816          ;*
5817          ;*
5818          ;*
5819          ;*
5820          ;*
5821          ;*
5822          ;*
5823          ;*
5824          ;*
5825          ;*
5826          ;*
5827          ;*
5828          ;*
5829          ;*
5830          ;*
5831          ;*
5832          ;*
5833          ;*
5834          ;*
5835          ;*
5836          ;*
5837          ;*
5838          ;*
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          ;*
5884          ;*
5885          ;*
5886          ;*
5887          ;*
5888          ;*
5889          ;*
5890          ;*
5891          ;*
5892          ;*
5893          ;*
5894          ;*
5895          ;*
5896          ;*
5897          ;*
5898          ;*
5899          ;*
5900          ;*
5901          ;*
5902          ;*
5903          ;*
5904          ;*
5905          ;*
5906          ;*
5907          ;*
5908          ;*
5909          ;*
5910          ;*
5911          ;*
5912          ;*
5913          ;*
5914          ;*
5915          ;*
5916          ;*
5917          ;*
5918          ;*
5919          ;*
5920          ;*
5921          ;*
5922          ;*
5923          ;*
5924          ;*
5925          ;*
5926          ;*
5927          ;*
5928          ;*
5929          ;*
5930          ;*
5931          ;*
5932          ;*
5933          ;*
5934          ;*
5935          ;*
5936          ;*
5937          ;*
5938          ;*
5939          ;*
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          ;*
5996          ;*
5997          ;*
5998          ;*
5999          ;*
6000          ;*
    
```

```

5313 023100 101405          BLOS     48          ;;BR IF YES
5314 023102 104410          RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
5315 023104 112613          MOV      (SP)+,(R3) ;;GET CHARACTER
5316 023106 122713 000177    108: CMP      #177,(R3) ;;IS IT A RUBOUT
5317 023112 001003          BNE      38          ;;SKIP IF NOT
5318 023114 104401 001212    48: TYPE    #QUES    ;;TYPE A '?'
5319 023120 000763          BR       18          ;;CLEAR THE BUFFER AND LOOP
5320 023122 111337 023472    38: MOV      (R3),98    ;;ECHO THE CHARACTER
5321 023126 104401 023472          TYPE     ,98
5322 023132 122723 000015          CMP      #15,(R3)+  ;;CHECK FOR RETURN
5323 023136 001356          BNE      28          ;;LOOP IF NOT RETURN
5324 023140 105063 177777          CLRB    -(R3)      ;;CLEAR RETURN (THE 15)
5325 023144 104401 001214          TYPE    ,4LF      ;;TYPE A LINE FEED
5326 023150 012603          MOV      (SP)+,R3  ;;RESTORE R3
5327 023152 011646          MOV      (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
5328 023154 016666 000004 000002    MOV      4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
5329 023162 012766 023474 000004          MOV      #TTYIN,4(SP)
5330 023170 000002          RTI          ;;RETURN
5331 023172 000          98: ,BYTE  0          ;;STORAGE FOR ASCII CHAR, TO TYPE
5332 023173 000          ,BYTE  0          ;;TERMINATOR
5333 023174 000010          ,BLKB  8          ;;RESERVE 8 BYTES FOR TTY INPUT
5334 023104 052536 005015 000    %CNTLU: ,ASCIIZ /"U/<15><12> ;;CONTROL "U"
5335 023111 136 006507 000012    %CNTLG: ,ASCIIZ /"G/<15><12> ;;CONTROL "G"
5336 023116 005015 053523 020122    %MSWR:  ,ASCIIZ <15><12>/MSWR = /
5337 023124 020075 000          %MNEW:  ,ASCIIZ / NEW = /
5338 023127 040 047040 053505          %MNEW:  ,ASCIIZ / NEW = /
5339 023134 036440 000040
    
```

```

5340          ,SBTTL READ AN OCTAL NUMBER FROM THE TTY
5341
5342          ;*****
5343          ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
5344          ;*CHANGE IT TO BINARY.
5345          ;*CALL:
5346          ;*      RDOCT          ;;READ AN OCTAL NUMBER
5347          ;*      RETURN HERE    ;;LOW ORDER BITS ARE ON TOP OF THE STACK
5348          ;*                    ;;HIGH ORDER BITS ARE IN $HIOCT
5349
5350 023440 011646          SRDOCT: MOV      (SP),=(SP)          ;;PROVIDE SPACE FOR THE
5351 023442 016666          MOV      4(SP),2(SP)          ;;INPUT NUMBER
5352 023450 010046          MOV      RO,=(SP)          ;;PUSH RO ON STACK
5353 023452 010146          MOV      R1,=(SP)          ;;PUSH R1 ON STACK
5354 023454 010246          MOV      R2,=(SP)          ;;PUSH R2 ON STACK
5355 023456 104411          18:  RDLIN          ;;READ AN ASCII LINE
5356 023460 012600          MOV      (SP)+,RO          ;;GET ADDRESS OF 1ST CHARACTER
5357 023462 005001          CLR      R1          ;;CLEAR DATA WORD
5358 023464 005002          CLR      R2
5359 023466 112046          28:  MOVBS   (RO)+,=(SP)          ;;PICKUP THIS CHARACTER
5360 023470 001412          BEQ     38          ;;IF ZERO GET OUT
5361 023472 006301          ASL     R1          ;;*2
5362 023474 006102          ROL     R2
5363 023476 006301          ASL     R1          ;;*4
5364 023480 006102          ROL     R2
5365 023482 006301          ASL     R1          ;;*8
5366 023484 006102          ROL     R2
5367 023486 042716          BIC     #'C',(SP)          ;;STRIP THE ASCII JUNK
5368 023412 063601          ADD     (SP)+,R1          ;;ADD IN THIS DIGIT
5369 023414 000764          BR      28          ;;LOOP
5370 023416 005726          38:  TST     (SP)+          ;;CLEAN TERMINATOR FROM STACK
5371 023420 010166          MOV     R1,12(SP)          ;;SAVE THE RESULT
5372 023424 010237          MOV     R2,$HIOCT
5373 023430 012602          MOV     (SP)+,R2          ;;POP STACK INTO R2
5374 023432 012601          MOV     (SP)+,R1          ;;POP STACK INTO R1
5375 023434 012600          MOV     (SP)+,RO          ;;POP STACK INTO RO
5376 023436 000002          RTI
5377 023440 000000          $HIOCT: ,WORD 0          ;;HIGH ORDER BITS GO HERE
    
```

```

5379          ,SBTTL READ A DECIMAL NUMBER FROM THE TTY
5380
5381          ;*****
5382          ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
5383          ;*CHANGE IT TO BINARY, IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
5384          ;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN=LINE FEED WILL BE TYPED.
5385          ;*THE COMPLETE NUMBER MUST BE RETYPED, THE INPUT IS TERMINATED BY THE
5386          ;*USER TYPING A CARRIAGE RETURN, THE RANGE OF THE INPUT NUMBER IS
5387          ;*POSITIVE 32767 TO NEGATIVE 32768.
5388          ;*CALL:
5389          ;*      RDDEC          ;;READ A DECIMAL NUMBER
5390          ;*      RETURN HERE    ;;NUMBER IS ON TOP OF THE STACK
5391
5392 023442 011646          SRDDEC: MOV      (SP),=(SP)          ;;PROVIDE SPACE FOR
5393 023444 016666          MOV      4(SP),2(SP)          ;;THE INPUT NUMBER
5394 023452 010046          MOV      RO,=(SP)          ;;PUSH RO ON STACK
5395 023454 010146          MOV      R1,=(SP)          ;;PUSH R1 ON STACK
5396 023456 010246          MOV      R2,=(SP)          ;;PUSH R2 ON STACK
5397 023460 104411          18:  RDLIN          ;;READ AN ASCII LINE
5398 023462 012600          MOV      (SP)+,RO          ;;ADDRESS OF 1ST CHAR.
5399 023464 010037          MOV      RO,68          ;;SAVE INCBASE OF BAD INPUT
5400 023470 005046          CLR      -(SP)          ;;CLEAR DATA WORD
5401 023472 005002          CLR      R2          ;;SIGN SET POSITIVE
5402 023474 122710          000085          CMPB    #'-',(RO)          ;;SEE IF A MINUS SIGN WAS TYPED
5403 023700 001001          BNE     28          ;;BR IF NO MINUS SIGN
5404 023702 112002          MOVBS   (RO)+,R2          ;;SAVE FOR LATER USE
5405 023704 112001          28:  MOVBS   (RO)+,R1          ;;PICKUP THIS CHARACTER
5406 023706 001424          BEQ     38          ;;GET OUT IF ZERO
5407 023710 122701          000060          CMPB    #'0',R1          ;;MAKE SURE THIS CHARACTER
5408 023714 003032          BGT     58          ;;IS A DIGIT BETWEEN 0 & 9
5409 023716 122701          000071          CMPB    #'9',R1
5410 023722 002427          BLT     58
5411 023724 032718          170000          BIT     #'C7777',(SP)          ;;DON'T LET NUMBER GET TO BIG
5412 023730 001024          BNE     58          ;;BR IF NUMBER WOULD OVERFLOW
5413 023732 006316          ASL     (SP)          ;;*2
5414 023734 011646          MOV     (SP),=(SP)          ;;SAVE FOR LATER
5415 023736 006316          ASL     (SP)          ;;*4
5416 023740 006316          ASL     (SP)          ;;*8
5417 023742 062616          ADD     (SP)+,(SP)          ;;*10.
5418 023744 102416          BVS     58          ;;OVERFLOW ISN'T ALLOWED
5419 023746 162701          000060          SUB     #'0',R1          ;;STRIP AWAY THE ASCII JUNK
5420 023752 060116          ADD     R1,(SP)          ;;ADD IN THIS DIGIT
5421 023754 102412          BVS     58          ;;OVERFLOW ISN'T ALLOWED
5422 023756 000752          BR      28          ;;LOOP
5423 023760 005702          38:  TST     R2          ;;CHECK IF NUMBER IS NEG
5424 023762 001401          BEQ     48          ;;BR IF NO
5425 023764 005416          NEG     (SP)          ;;YES==NEGATE THE NUMBER
5426 023766 012666          MOV     (SP)+,12(SP)          ;;SAVE THE RESULT
5427 023772 012602          MOV     (SP)+,R2          ;;POP STACK INTO R2
5428 023774 012601          MOV     (SP)+,R1          ;;POP STACK INTO R1
5429 023776 012600          MOV     (SP)+,RO          ;;POP STACK INTO RO
5430 024000 000002          RTI          ;;RETURN
5431
5432 024002 005726          58:  TST     (SP)+          ;;CLEAN PARTIAL NUMBER FROM STACK
5433 024004 105010          CLRB   (RO)          ;;SET A TERMINATOR
    
```



```

5494 024502 012501      MOV      (SP)+,R1      ;;POP STACK INTO R1
5495 024504 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
5496 024506 104401 024234  TYPE      ,#DBLK      ;;NOW TYPE THE NUMBER
5497 024512 016666 000002 000004  MOV      2(SP),4(SP)   ;;ADJUST THE STACK
5498 024520 012616      MOV      (SP)+,(SP)
5499 024522 000002      RTI
5500 024524 021420      #DTRL: 10000,        ;;RETURN TO USER
5501 024526 001750        1000,
5502 024530 000144        100,
5503 024532 000012        10,
5504 024534 000004      #DBLK: ,BLKW 4
    
```

```

5505      #BTTL 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 024544 105737 001157      #TYPE: TSTB  #TPFLG      ;;IS THERE A TERMINAL?
5523 024550 100002          BPL  18                ;;BR IF YES
5524 024552 000000          HALT          ;;HALT HERE IF NO TERMINAL
5525 024554 000407          BR  38                ;;LEAVE
5526 024556 010046          MOV  RO,-(SP)         ;;SAVE RO
5527 024560 017600 000002      MOV  #2(SP),RO       ;;GET ADDRESS OF ASCIZ STRING
5528 024564 112046          MOV  (RO)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
5529 024566 001005          BNE  48                ;;BR IF IT ISN'T THE TERMINATOR
5530 024570 005726          TST  (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
5531 024572 012600 608:      MOV  (SP)+,RO       ;;RESTORE RO
5532 024574 062716 000002      ADD  #2,(SP)         ;;ADJUST RETURN PC
5533 024580 000002          RTI
5534 024582 122716 000011      CMPB #HT,(SP)       ;;BRANCH IF <HT>
5535 024586 001430          BEQ  88                ;;BRANCH IF NOT <CRLF>
5536 024590 122716 000200      CMPB #CRLF,(SP)    ;;BRANCH IF NOT <CRLF>
5537 024594 001006          BNE  58                ;;BRANCH IF NOT <CRLF>
5538 024596 005726          TST  (SP)+           ;;POP <CR><LF> EQUIV
5539 024598 104401          TYPE          ;;TYPE A CR AND LF
5540 024598 001213          #CRLF
5541 024598 105037 024460      CLRB          ;;CLEAR CHARACTER COUNT
5542 024598 000755          BR  28                ;;GET NEXT CHARACTER
5543 024598 004737 024414      JSR  PC,#TYPEC      ;;GO TYPE THIS CHARACTER
5544 024598 123726 001156      CMPB #FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHAR.?
5545 024598 001350          BNE  28                ;;IF NO GO GET NEXT CHAR.
5546 024598 013746 001154      MOV  #NULL,-(SP)    ;;GET # OF FILLER CHAR. NEEDED
5547                                ;;AND THE NULL CHAR.
5548 024598 105366 000001      DECB 1(SP)          ;;DOES A NULL NEED TO BE TYPED?
5549 024598 002770          BLT  68                ;;BR IF NO=GO POP THE NULL OFF OF STACK
5550 024598 004737 024414      JSR  PC,#TYPEC      ;;GO TYPE A NULL
5551 024598 105337 024460      DECB #CHARCNT      ;;DO NOT COUNT AS A COUNT
5552 024598 000770          BR  78                ;;LOOP
5553
5554      ;HORIZONTAL TAB PROCESSOR
5555
5556 024598 112716 000040      88:  MOV  # ,(SP)       ;;REPLACE TAB WITH SPACE
5557 024598 004737 024414      90:  JSR  PC,#TYPEC      ;;TYPE A SPACE
5558 024598 132737 000007 024460      BITB #7,#CHARCNT   ;;BRANCH IF NOT AT
5559 024598 001372          BNE  98                ;;TAB STOP
5560 024598 005726          TST  (SP)+           ;;POP SPACE OFF STACK
    
```

```

5561 024112 000724      HP      28      ;GET NEXT CHARACTER
5562 024414 108777      TSTR   08TPB   ;WAIT UNTIL PRINTER IS READY
5563 024420 100175      HPL    0TYPEC  ;
5564 024422 118677      MOVB   2(SP),08TPB ;LOAD CHAR TO BE TYPED INTO DATA REG.
5565 024430 122766      CNFB   *CR,2(SP)   ;IS CHARACTER A CARRIAGE RETURN?
5566 024436 001003      BNE    18        ;BRANCH IF NO
5567 024440 109037      CLR   0CHARCNT   ;YES--CLEAR CHARACTER COUNT
5568 024444 000406      BR     0TYPEX   ;EXIT
5569 024446 122766      CNFB   *LF,2(SP)  ;IS CHARACTER A LINE FEED?
5570 024454 001402      BEQ   0TYPEX   ;BRANCH IF YES
5571 024456 109227      INCB  (PC)+     ;COUNT THE CHARACTER
5572 024460 000000      0CHARCNT,WORD 0 ;CHARACTER COUNT STORAGE
5573 024462 000207      0TYPEX,RTS    PC
5574

```

```

5575      ,SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
5576
5577      ;*****
5578      ;THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
5579      ;UNSIGNED OCTAL ASCII NUMBER.
5580      ;CALL
5581      ;*   MOV   $PNTR,=(SP)      ;POINTER TO LOW WORD OF BINARY NUMBER
5582      ;*   JSR   PC,000DB20      ;CALL THE ROUTINE
5583      ;*   RETURN                 ;THE ADDRESS OF THE FIRST ASCII CHAR, IS ON THE STACK
5584
5585
5586 024464 104414      0DB20: SAVREG      ;SAVE ALL REGISTERS
5587 024466 016601      MOV   2(SP),R1     ;PICKUP THE POINTER TO LOW WORD
5588 024472 012705      MOV   00OCTLV+13,R5 ;POINTER TO DATA TABLE
5589 024476 012704      MOV   012,R4       ;DO ELEVEN CHARACTERS
5590 024502 012703      MOV   *C7,R3      ;MASK
5591 024506 012100      MOV   (R1)+,R0     ;LOWER WORD
5592 024510 012101      MOV   (R1)+,R1     ;HIGH WORD
5593 024512 005002      CLR   R2           ;TERMINATOR
5594 024514 110245      18:  MOVB   R2,=(R5)   ;PUT CHARACTER IN DATA TABLE
5595 024516 010002      MOV   R0,R2        ;GET THIS DIGIT
5596 024520 005304      DEC   R4           ;COUNT THIS CHARACTER
5597 024522 003007      BGT   38           ;BR IF NOT THE LAST DIGIT
5598 024524 001405      BEQ   28           ;BR IF IT IS THE LAST DIGIT
5599 024526 005205      INC   R5           ;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
5600 024530 010566      MOV   RS,2(SP)     ;ASCII CHAR, & PUT IT ON THE STACK
5601 024534 104415      RESREG             ;RESTORE ALL REGISTERS
5602 024536 000207      RTS   PC           ;RETURN TO USER
5603 024540 006203      28:  ABR   R3        ;POSITION THE MASK FOR THE LAST DIGIT
5604 024542 006001      38:  ROR   R1        ;POSITION THE BINARY NUMBER FOR
5605 024544 006000      ROR   R0           ; THE NEXT OCTAL DIGIT
5606 024546 006001      ROR   R1
5607 024550 006000      ROR   R0
5608 024552 006001      ROR   R1
5609 024554 006000      ROR   R0
5610 024556 040302      BIC   R3,R2        ;MASK OUT ALL JUNK
5611 024560 062702      ADD   #'0,R2       ;MAKE THIS CHAR, ASCII
5612 024564 000753      BR    18          ;GO PUT IT IN THE DATA TABLE
5613 024566 000016      0OCTLV: ,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     @PTR,=(SP)    ;POINTER TO LOW WORD OF BINARY NUMBER
5622          ;*   JBR    PC,@#DB2D    ;
5623          ;*   RETURN                ;THE FIRST ADDRESS OF ASCII
5624          ;*                           ;IS ON THE STACK
5625
5626
5627 024404 104414          SDB2D:  SAVREG                ;SAVE REGISTERS
5628 024406 016602 000002  MOV     2(SP),R2          ;PICKUP THE DATA POINTER
5629 024412 012700 024764  MOV     @#DECVL,R0       ;GET ADDRESS OF "#DECVL" STRING
5630 024416 010066 000002  MOV     R0,2(SP)        ;PUT ADDRESS OF ASCII STRING ON STACK
5631 024422 012201          MOV     (R2)+,R1        ;PICKUP THE BINARY NUMBER
5632 024424 012202          MOV     (R2)+,R2
5633 024426 012737 000012 024702  MOV     #10,,#48        ;SET UP TO DO 10 CONVERSIONS
5634 024434 012704 024714  MOV     @#TNPWR,R4       ;ADDRESS OF TEN POWER
5635 024440 012705 024716  MOV     @#TNPWR+2,R5
5636 024444 008003          16:   CLR     R3                ;CLEAR PARTIAL
5637 024446 161401          20:   SUB     (R4),R1          ;SUBTRACT TEN POWER
5638 024450 008402          SUB     R2
5639 024452 161502          SUB     (R5),R2
5640 024454 002402          BLT     30                ;BR IF TEN POWER TO LARGE
5641 024456 008203          INC     R3                ;ADD 1 TO PARTIAL
5642 024660 000772          BR      28                ;LOOP
5643 024462 062401          38:   ADD     (R4)+,R1        ;RESTORE SUBTRACTED VALUE
5644 024464 005502          ADC     R2
5645 024466 062402          ADD     (R4)+,R2
5646 024470 022523          CMP     (R5)+,(R5)+      ;MOVE TO NEXT TEN POWER
5647 024472 052703 000060  BIS     #0,R3            ;CHANGE PARTIAL TO ASCII
5648 024476 110320          MOVB   R3,(R0)+          ;SAVE IT
5649 024700 005327          DEC     (PC)+            ;DONE?
5650 024702 000000          48:   .WORD 0
5651 024704 001357          BNE     16                ;BR IF NO
5652 024706 105020          CLRB   (R0)+            ;TERMINATOR
5653 024710 104415          RESREG                ;RESTORE REGISTERS
5654 024712 000207          RTS     PC                ;RETURN
5655 024714 145000          #TNPWR: 145000            ;1.0E09
5656 024716 035632          35632                    ;1.0E08
5657 024720 160400          160400                    ;1.0E07
5658 024722 002765          2765                      ;1.0E06
5659 024724 113200          113200                    ;1.0E05
5660 024726 000230          230                        ;1.0E04
5661 024730 041100          041100                    ;1.0E03
5662 024732 000017          17                          ;1.0E02
5663 024734 103240          103240                    ;1.0E01
5664 024736 000001          1                            ;1.0E00
5665 024740 023420          23420                      ;1.0E00
5666 024742 000000          0
5667 024744 001750          1750                        ;1.0E00
5668 024746 000000          0
5669 024750 000144          144                        ;1.0E02
    
```

```

5670 024752 000000          0
5671 024754 000012          12                    ;1.0E01
5672 024756 000000          0
5673 024760 000001          1                    ;1.0E00
5674 024762 000000          0
5675 024764 000014          #DECVL: #BLKS 12,      ;RESERVE STORAGE FOR ASCII STRING
    
```

```

5676          ,SBTTL SUPRS = TYPE NUMERICAL ASCIZ STRING, REPLACE LEADING 0'S BY BLANKS
5677          ,SBTTL SUPRSL = TYPE NUMERICAL ASCIZ STRING, LEFT JUSTIFY
5678          ;NOT FROM SYBMAC
5679
5680 025A00 010046          SUPRSL: MOV    R0,=(SP)          ;SAVE R0
5681 025A02 005037 025070 CLR    SUP2
5682 025A06 016600 000004 MOV    4(SP),R0
5683 025A12 010405          BR     SUP1
5684
5685 025A14 010046          SUPRS: MOV    R0,=(SP)          ;SAVE R0
5686 025A16 016600 000004 MOV    4(SP),R0          ;PICKUP THE POINTER
5687 025A22 010037 025070 MOV    R0,SUP2          ;SAVE FOR TYPING
5688 025A26
5689 025A26 105710          SUP1: 18:  TSTB   (R0)          ;TERMINATOR?
5690 025A30 001406          BEQ    28
5691 025A32 122710 000060 CMBB   #'0,(R0)          ;IS THIS AN ASCII '0'?
5692 025A36 001006          BNE    48
5693 025A40 112720 000040 MOVB   #40,(R0)+
5694 025A44 000770          BR     18
5695 025A46 003300          28:  DEC    R0          ;BACKUP BY 1
5696 025A80 112710 000060 MOVB   #'0,(R0)          ;ASCII '0'
5697 025A54 005737 025070 48:  TST   SUP2          ;LEFT JUSTIFY?
5698 025A60 001002          BNE    58
5699 025A62 010037 025070 MOV    R0,SUP2          ;YES
5700 025A66 104401          58:  TYPE   ;GO TYPE
5701 025A70 000000          SUP2: ,WORD  0
5702 025A72 012800          MOV    (SP)+,R0          ;RESTORE R0
5703 025A74 012816          MOV    (SP)+,(SP)       ;RESTORE THE STACK
5704 025A76 000207          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 025100          #MULT:
5720 025100 010046          MOV    R0,=(SP)          ;;PUSH R0 ON STACK
5721 025102 010146          MOV    R1,=(SP)          ;;PUSH R1 ON STACK
5722 025104 010246          MOV    R2,=(SP)          ;;PUSH R2 ON STACK
5723 025106 005046          CLR    -(SP)            ;;CLEAR THE SIGN KEY
5724 025110 016601 000012 MOV    12(SP),R1        ;;GET THE MULTIPLICAND
5725 025114 100002          BPL    18
5726 025116 005216          INC    (SP)            ;;SET THE SIGN KEY
5727 025120 005401          NEG    R1              ;;MAKE THE MULTIPLICAND POSITIVE
5728 025122 016602 000014 18:  MOV    14(SP),R2        ;;GET THE MULTIPLIER
5729 025126 100002          BPL    28
5730 025130 005316          DEC    (SP)            ;;UPDATE THE SIGN KEY
5731 025132 005402          NEG    R2              ;;MAKE THE MULTIPLIER POSITIVE
5732 025134 012746 000021 28:  MOV    #17,=(SP)       ;;SET THE LOOP COUNT
5733 025140 005000          CLR    R0              ;;SETUP FOR THE MULTIPLY LOOP
5734 025142 103001          BCC    48
5735 025144 060200          ADD    R2,R0           ;;DON'T ADD IF MULTIPLICAND = 0
5736 025146 006000          48:  ROR    R0              ;;POSITION THE PARITIAL PRODUCT AND
5737 025150 006001          ROR    R1              ;;THE MULTIPLICAND
5738 025152 005316          DEC    (SP)            ;;HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
5739 025154 001372          BNE    38
5740 025156 022616          CMP    (SP)+,(SP)      ;;SHOULD PRODUCT BE NEGATIVE?
5741 025160 001403          BEQ    58
5742 025162 005400          NEG    R0              ;;GO TO EXIT IF NO
5743 025164 005401          NEG    R1              ;;YES--SO MAKE IT SO
5744 025166 005600          SBC    R0
5745 025170 005726          38:  TST   (SP)+          ;;CLEAR SIGN INFO. OFF OF STACK
5746 025172 010066 000012 MOV    R0,12(SP)        ;;PUT THE PRODUCT ON THE STACK (MSB'S)
5747 025176 010166 000010 MOV    R1,10(SP)        ;;LSB'S
5748 025202 012602          MOV    (SP)+,R2        ;;POP STACK INTO R2
5749 025204 012601          MOV    (SP)+,R1        ;;POP STACK INTO R1
5750 025206 012600          MOV    (SP)+,R0        ;;POP STACK INTO R0
5751 025210 000207          RTS   PC
    
```

```

5752          ,SBTTL INTEGER DIVIDE ROUTINE
5753
5754          ;*CALL:
5755          ;*   MOV     LOW DIVIDEND,-(SP)      ;THE HIGH DIVIDEND MUST BE < 1/2
5756          ;*   MOV     HIGH DIVIDEND,-(SP)    ; AS LARGE AS THE DIVISOR
5757          ;*   MOV     DIVISOR,-(SP)
5758          ;*   JSR     PC,SDIV
5759          ;*   RETURN
5760          ;*           IMPLIES NO ERROR          ;QUOTIENT & REMAINDER ARE ON THE STACK
5761          ;*           "Y"=0
5762          ;*           "Y"=1 IMPLIES ERROR OCCURRED
5763          ;*           "C"=0 DIVIDE OVERFLOW OCCURRED
5764          ;*           "C"=1 ATTEMPTED TO DIVIDE BY ZERO
5765
5766          ;*
5767          ;*   STACK NO ERROR OVERFLOW DIVIDE BY ZERO
5768          ;*   -----
5769          ;*   TOP REMAINDER ALL ZEROS ALL ONES
5770          ;*   +2 QUOTIENT ALL ZEROS ALL ONES
5771
5771 025712 013748 000034 SDIV: MOV 34,-(SP) ;SAVE CURRENT TRAP VECTOR
5772 025716 012737 028226 000034 MOV #18,34 ;SET UP TRAP VECTOR
5773 025724 104400 TRAP
5774 025726 012716 028280 18: MOV #28,(SP) ;REPLACE NEW PC
5775 025732 018637 000004 000034 MOV 4(SP),34 ;RESTORE OLD TRAP VECT
5776 025740 018686 000002 000004 MOV 2(SP),4(SP) ;SAVE PSW
5777 025746 000002 RTI ;RESTORE PSW
5778
5779 025750 042716 000017 28: BIC #17,(SP) ;STRIP AWAY CONDITION CODES
5780 025754 010046 MOV R0,-(SP) ;PUSH R0 ON STACK
5781 025756 010146 MOV R1,-(SP) ;PUSH R1 ON STACK
5782 025760 010246 MOV R2,-(SP) ;PUSH R2 ON STACK
5783 025762 010346 MOV R3,-(SP) ;PUSH R3 ON STACK
5784 025764 008046 CLR -(SP) ;SAVE A PLACE FOR SIGNS
5785 025766 012746 000021 MOV #17,-(SP) ;SETUP THE ITERATION COUNTER
5786 025772 018601 000024 MOV 24(SP),R1 ;PICKUP THE DIVIDEND
5787 025776 018600 000022 MOV 22(SP),R0
5788 025780 100008 BPL 38 ;CHECK THE SIGN
5789 025784 105386 000003 38: DECB 3(SP) ;KEEP TRACK OF THE SIGN
5790 025788 005400 NEG R0 ;AND NEGATE THE ORIGINAL
5791 025792 005401 NEG R1 ;NUMBER
5792 025794 005800 SBC R0
5793 025796 016602 000020 38: MOV 20(SP),R2 ;PICKUP THE DIVISOR
5794 025798 022702 000001 38: CMP #1,R2 ;IF THE DIVISOR IS 1 SKIP THE REST
5795 025800 001443 BEQ 138 ;YES
5796 025802 008702 TST R2
5797 025804 002407 BLT 48 ;CHECK THE SIGN
5798 025806 003011 BGT 58 ;DIVISOR OF 0 IS A NO-NO
5799 025808 032766 000003 000014 BIF #3,14(SP) ;SET "Y" & "C"
5800 025810 012700 177777 MOV #1,R0 ;SET REMAINDER TO ALL ONES
5801 025812 000424 BR 98 ;EXIT
5802 025814 005266 000002 48: INC 2(SP) ;KEEP TRACK OF DIVISOR SIGN
5803 025816 000401 BR 68
5804 025818 005402 58: NEG R2 ;NEGATE THE ORIGINAL NUMBER
5805 025820 000241 68: CLC ;CLEAR "C"
5806 025822 000405 BR 88 ;START FORMING QUOTIENT
5807 025824 006100 78: ROL R0 ;POSITION MSB'S
    
```

```

5808 025826 010003 MOV R0,R3 ;COPY
5809 025828 060203 ADD R2,R3 ;COMPARE DIVIDEND & DIVISOR
5810 025830 103001 SBC 88 ;BR IF DIVIDEND > DIVISOR
5811 025832 010300 MOV R3,R0 ;REMAINDER AFTER THIS LOOP
5812 025834 006101 88: ROL R1 ;QUOTIENT BIT ENTERS HERE
5813 025836 005316 DEC (SP) ;DONE?
5814 025838 001370 BNE 76 ;BR IF NO
5815 025840 005701 TST R1 ;OVERFLOW?
5816 025842 100005 BPL 108 ;BR IF NO
5817 025844 032766 000002 000014 BIF #2,14(SP) ;SET "Y" IN RETURN STATUS WORD
5818 025846 005000 CLR R0 ;SET REMAINDER TO ALL ZEROS
5819 025848 010001 98: MOV R0,R1 ;COPY REMAINDER INTO QUOTIENT
5820 025850 005726 108: TST (SP)+ ;CLEAR COUNTER FROM STACK
5821 025852 005716 TST (SP) ;REMAINDER SIGN CORRECTION NEEDED?
5822 025854 002004 BGE 118 ;BR IF NO
5823 025856 005400 NEG R0 ;NEGATE REMAINDER
5824 025858 105066 000001 118: CLRB 1(SP) ;CLEAR SIGN
5825 025860 005316 DEC (SP) ;BUT DON'T FORGET QUOTIENT
5826 025862 005726 118: TST (SP)+ ;QUOTIENT SIGN CORRECTION NEEDED?
5827 025864 001401 BEQ 128 ;BR IF NO
5828 025866 005401 NEG R1 ;NEGATE QUOTIENT
5829 025868 010166 000020 128: MOV R1,20(SP) ;RETURN QUOTIENT AND
5830 025870 010066 000016 MOV R0,16(SP) ;REMAINDER TO USER
5831 025872 012603 MOV (SP)+,R3 ;POP STACK INTO R3
5832 025874 012602 MOV (SP)+,R2 ;POP STACK INTO R2
5833 025876 012601 MOV (SP)+,R1 ;POP STACK INTO R1
5834 025878 012600 MOV (SP)+,R0 ;POP STACK INTO R0
5835 025880 012666 000002 MOV (SP)+,2(SP) ;SETUP TO RETURN CONDITION CODES
5836 025882 000002 RTI ;RETURN
5837 025884 022626 138: CMP (SP)+,(SP)+ ;POP THE STACK
5838 025886 000763 BR 128
    
```



```

5839 ;SBTTL SAVE AND RESTORE R0-R5 ROUTINES
5840 ;*CALL:
5841 ;*****
5842 ;SAVE R0-R5
5843 ;*CALL:
5844 ;* SAVREG
5845 ;UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
5846 ;*
5847 ;*TOP==(+16)
5848 ;* +2==(+18)
5849 ;* +4==R5
5850 ;* +6==R4
5851 ;* +8==R3
5852 ;* +10==R2
5853 ;* +12==R1
5854 ;* +14==R0
5855
5856 025402 $SAVREG:
5857 025402 010046 MOV R0,=(SP) ;PUSH R0 ON STACK
5858 025404 010146 MOV R1,=(SP) ;PUSH R1 ON STACK
5859 025406 010246 MOV R2,=(SP) ;PUSH R2 ON STACK
5860 025410 010346 MOV R3,=(SP) ;PUSH R3 ON STACK
5861 025412 010446 MOV R4,=(SP) ;PUSH R4 ON STACK
5862 025414 010546 MOV R5,=(SP) ;PUSH R5 ON STACK
5863 025416 018646 000022 MOV 22(SP),=(SP) ;SAVE P6 OF MAIN FLOW
5864 025422 018646 000022 MOV 22(SP),=(SP) ;SAVE PC OF MAIN FLOW
5865 025426 018646 000022 MOV 22(SP),=(SP) ;SAVE P6 OF CALL
5866 025432 018646 000022 MOV 22(SP),=(SP) ;SAVE PC OF CALL
5867 025436 000002 RTI
5868
5869 ;*RESTORE R0-R5
5870 ;*CALL:
5871 ;* RESREG
5872 025440 $RESREG:
5873 025440 012666 000022 MOV (SP)+,22(SP) ;RESTORE PC OF CALL
5874 025444 012666 000022 MOV (SP)+,22(SP) ;RESTORE P6 OF CALL
5875 025450 012666 000022 MOV (SP)+,22(SP) ;RESTORE PC OF MAIN FLOW
5876 025454 012666 000022 MOV (SP)+,22(SP) ;RESTORE P6 OF MAIN FLOW
5877 025460 012605 MOV (SP)+,R5 ;POP STACK INTO R5
5878 025462 012604 MOV (SP)+,R4 ;POP STACK INTO R4
5879 025464 012603 MOV (SP)+,R3 ;POP STACK INTO R3
5880 025466 012602 MOV (SP)+,R2 ;POP STACK INTO R2
5881 025470 012601 MOV (SP)+,R1 ;POP STACK INTO R1
5882 025472 012600 MOV (SP)+,R0 ;POP STACK INTO R0
5883 025474 000002 RTI
    
```

```

5884 ;SBTTL RANDOM NUMBER GENERATOR ROUTINE
5885 ;*CALL:
5886 ;* JSR PC,$RAND ;CALL THE ROUTINE
5887 ;* RETURN ;RETURN HERE THE RANDOM
5888 ;* ;NUMBER WILL BE IN
5889 ;* ;$HINUM,$LONUM
5890 025476 $RAND:
5891 025476 010046 MOV R0,=(SP) ;PUSH R0 ON STACK
5892 025400 010146 MOV R1,=(SP) ;PUSH R1 ON STACK
5893 025402 010246 MOV R2,=(SP) ;PUSH R2 ON STACK
5894 025404 010346 MOV R3,=(SP) ;PUSH R3 ON STACK
5895 025406 010446 MOV R4,=(SP) ;PUSH R4 ON STACK
5896 025410 017604 000012 MOV @12(SP),R4 ;GET POINTER TO THE SAVED SEEDS
5897 ;FOR GENERATING THIS RANDOM NUMBER
5898 025414 011400 MOV (R4),R0 ;GET LO NUMBER SEED
5899 025416 016401 000002 MOV 2(R4),R1 ;GET HIGH NUMBER SEED
5900 025422 012703 177771 MOV #-7,R3 ;SET SHIFT COUNT
5901 025426 005002 CLR R2 ;ZERO R2
5902 025430 006300 181 ASL R0 ;SHIFT R0 LEFT AND
5903 025432 006101 ROL R1 ;ROTATE CARRY INTO R1 AND
5904 025434 006102 ROL R2 ;ROTATE CARRY INTO R2
5905 025436 005203 INC R3 ;CHECK FOR DONE
5906 025440 001373 BNE 18 ;CONTINUE SHIFT LOOP
5907 025442 0061400 ADD (R4),R0 ;ADD NUMBER TO MAKE X 129
5908 025444 005501 ADC R1 ;PROPOGATE CARRY
5909 025446 006401 000002 ADD 2(R4),R1 ;ADD NUMBER TO MAKE X 129
5910 025452 005502 ADC R2 ;PROPOGATE CARRY
5911 025454 0062700 001057 ADD #1057,R0 ;ADD LOW CONSTANT
5912 025460 005501 ADC R1 ;PROPOGATE CARRY
5913 025662 005502 ADC R2 ;PROPOGATE CARRY
5914 025464 0062701 047401 ADD #47401,R1 ;ADD HIGH CONSTANT
5915 025470 005502 ADC R2 ;PROPOGATE CARRY
5916 025472 0062702 000006 ADD #6,R2 ;ADD HIGHEST CONSTANT
5917 025676 0060200 ADD R2,R0 ;REPRIME R0 WITH HIGHEST DIGIT
5918 025700 005501 ADC R1 ;PROPOGATE CARRY
5919 025702 010014 MOV R0,(R4) ;SAVE R0=$LONUM (FOR USE NXT TIME)
5920 025704 010164 000002 MOV R1,2(R4) ;SAVE R1=$HINUM (FOR USE NXT TIME)
5921 025710 012604 MOV (SP)+,R4
5922 025712 012603 MOV (SP)+,R3 ;POP STACK INTO R3
5923 025714 012602 MOV (SP)+,R2 ;POP STACK INTO R2
5924 025716 012601 MOV (SP)+,R1 ;POP STACK INTO R1
5925 025720 012600 MOV (SP)+,R0 ;POP STACK INTO R0
5926 025722 0062716 000002 ADD #2,(SP) ;ADJUST SP FOR CORRECT RETURN
5927 025726 000207 RTS ;RETURN
5928 025730 123456 RSDRVL: 123456 ;RANDOM SEED FOR DRIVE SELECTION (LO)
5929 025732 176543 RSDRVH: 176543 ; " " (HI)
5930 025734 001201 R$FUNL: 1201 ;RANDOM SEED FOR FUNCTION
5931 025736 062465 R$FUNH: 62465 ; " " (HI)
5932 025740 176105 R$CYLL: 176105 ;RANDOM SEED FOR CYLINDER (LO)
5933 025742 174532 R$CYLH: 174532 ; " " (HI)
5934 025744 157650 R$BAL: 157650 ;RANDOM SEED FOR BUS ADDRESS (LO)
5935 025746 030753 R$BAH: 30753 ; " " (HI)
5936 025750 131547 R$WCL: 131547 ;RANDOM SEED FOR WORD COUNT (LO)
5937 025752 032070 R$WCH: 32070 ; " " (HI)
5938 025754 123456 R$DTL: 123456 ;RANDOM SEED FOR DATA (LO)
5939 025756 176543 R$DTH: 176543 ; " " (HI)
    
```

```

5940          ,SBTTL BINARY TO OCTAL (ASCII) AND TYPE
5941
5942          ;*****
5943          ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
5944          ;OCTAL (ASCII) NUMBER AND TYPE IT.
5945          ;#TYP0S---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
5946          ;CALLI
5947          ;# MOV     NUM,=(SP)      ;NUMBER TO BE TYPED
5948          ;# TYP0S      ;CALL FOR TYPEOUT
5949          ;# ,BYTE N      ;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
5950          ;# ,BYTE M      ;M=1 OR 0
5951          ;# ;#TYPE LEADING ZEROS
5952          ;# ;#0=SUPPRESS LEADING ZEROS
5953          ;#
5954          ;#TYP0N---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
5955          ;#TYP0S OR #TYP0C
5956          ;CALLI
5957          ;# MOV     NUM,=(SP)      ;NUMBER TO BE TYPED
5958          ;# TYP0N      ;CALL FOR TYPEOUT
5959          ;#
5960          ;#TYP0C---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
5961          ;CALLI
5962          ;# MOV     NUM,=(SP)      ;NUMBER TO BE TYPED
5963          ;# TYP0C      ;CALL FOR TYPEOUT
5964
5965          025760 017646 000000 026203 #TYP0S: MOV     0(SP),=(SP)      ;PICKUP THE MODE
5966          025764 116637 000001 026203 MOV     1(SP),#0FILL      ;LOAD ZERO FILL SWITCH
5967          025772 112637 026208 MOV     0(SP)+,#0MODE+1 ;NUMBER OF DIGITS TO TYPE
5968          025774 062716 000002 ADD     #2,(SP)          ;ADJUST RETURN ADDRESS
5969          026002 000406 BR      #TYP0N
5970          026004 112737 000001 026203 #TYP0C: MOV     #1,#0FILL      ;SET THE ZERO FILL SWITCH
5971          026012 112737 000008 026208 MOV     #6,#0MODE+1 ;SET FOR SIX(6) DIGITS
5972          026020 112737 000008 026202 #TYP0N: MOV     #6,#0CNT      ;SET THE ITERATION COUNT
5973          026026 010346 MOV     R3,=(SP)        ;SAVE R3
5974          026030 010446 MOV     R4,=(SP)        ;SAVE R4
5975          026032 010546 MOV     R6,=(SP)        ;SAVE R6
5976          026034 113704 026208 MOV     #0MODE+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 MOV     R4,#0MODE      ;SAVE IT FOR USE
5980          026052 113704 026203 MOV     #0FILL,R4 ;GET THE ZERO FILL SWITCH
5981          026056 016608 000012 MOV     12(SP),R6 ;PICKUP THE INPUT NUMBER
5982          026062 005003 CLR     R3            ;CLEAR THE OUTPUT WORD
5983          026064 026105 10:  ROL     R6            ;ROTATE HBB INTO "C"
5984          026066 000404 BR      #
5985          026070 006105 20:  ROL     R6            ;FORM THIS DIGIT
5986          026072 006105 ROL     R6
5987          026074 006105 ROL     R6
5988          026076 010803 MOV     R6,R3
5989          026100 006103 30:  ROL     R3            ;GET LSB OF THIS DIGIT
5990          026102 105337 026204 DECB   #0MODE ;TYPE THIS DIGIT
5991          026106 100016 BPL    #
5992          026110 042703 177770 BIC    #177770,R3 ;BR IF NO
5993          026114 001002 SNE    #
5994          026116 005704 TST    R4            ;GET RID OF JUNK
5995          026120 001403 BEQ    #            ;TEST FOR 0
                    ;SUPPRESS THIS 0?
                    ;BR IF YES

```

```

5996          026122 005204 40:  INC     R4            ;DON'T SUPPRESS ANYMORE 0'S
5997          026124 052703 000060 BIS    #0,R3          ;MAKE THIS DIGIT ASCII
5998          026130 052703 000040 50:  BIS    #*,R3          ;MAKE ASCII IF NOT ALREADY
5999          026134 110337 026200 MOV     R3,#0        ;SAVE FOR TYPING
6000          026140 104401 026200 TYPE   #0           ;GO TYPE THIS DIGIT
6001          026144 105337 026202 70:  DECB   #0CNT        ;COUNT BY 1
6002          026150 005347 BGT    #
6003          026152 002402 BLT    #
6004          026154 005204 INC     R4            ;BR IF MORE TO DO
6005          026156 000744 BR      #
6006          026160 012605 60:  MOV     (SP)+,R6 ;BR IF DONE
6007          026162 012604 MOV     (SP)+,R4 ;INSURE LAST DIGIT ISN'T A BLANK
6008          026164 012603 MOV     (SP)+,R3 ;GO DO THE LAST DIGIT
6009          026166 016666 000002 000004 MOV     2(SP),4(SP) ;RESTORE R6
6010          026174 012616 MOV     (SP)+,(SP) ;RESTORE R4
6011          026176 000002 RTI ;RESTORE R3
6012          026000 000 ;SET THE STACK FOR RETURNING
6013          026001 000 80:  ,BYTE 0 ;RETURN
6014          026002 000 ;STORAGE FOR ASCII DIGIT
6015          026003 000 #CNT: ,BYTE 0 ;TERMINATOR FOR TYPE ROUTINE
6016          026004 000000 #0FILL: ,BYTE 0 ;OCTAL DIGIT COUNTER
6017          #0MODE: ,WORD 0 ;ZERO FILL SWITCH
                    ;NUMBER OF DIGITS TO TYPE

```

```

,ERRTL ERROR HANDLER ROUTINE
6018
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 OCCURRING 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 ERRORS ON THIS DRIVE
6032 ;*SW01=1 TYPE OUT THE SERIAL NUMBER OF THE ERORING DRIVE
6033 ;GO TO ERRTP ON ERROR
6034
6035
6036 026906 104407 ;ERROR: CK&WR ;LOOK FOR A 'CONTROL G'
6037 026910 108237 001103 INCB ;ERRFLG ;SET THE ERROR FLAG
6038 026914 001774 BEQ ;ERRR ;DON'T LET THE FLAG GO TO ZERO
6039 026916 013777 001102 152716 MOV ;STNM,#DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
6040 026924 032777 002000 182706 BIT ;SW10,#SWR ;BELL ON ERROR?
6041 026932 001402 BEQ ; ;NO = SKIP
6042 026934 104401 001206 TYPE ;SBELL ;RING BELL
6043 026940 005237 001112 18: INC ;ERTTL ;COUNT THE NUMBER OF ERRORS
6044
6045 026944 032777 000004 152666 BIT ;SW2,#SWR ;COUNT # OF ERRORS & DROP DRIVE?
6046 026952 001415 BEQ ; ;NO
6047 ; ;YES
6048 026954 010146 MOV R1,=(SP) ;SAVE R1
6049 026956 013701 001250 MOV ;SRDRV,R1 ;GET ERRORING DRIVE #
6050 026962 100410 BMI ; ;IF (SRDRV)=-1, SKIP(BECAUSE THE
;EROR WAS NOT ATTRIBUTABLE TO ANY
;SPECIFIC DRIVE)
6051
6052
6053 026964 105261 001542 INCB ERDRV(R1) ;COUNT # OF ERRORS ON THIS DRIVE
6054 026970 126127 001542 000003 CMPB ERDRV(R1),#3 ;# OF ERRORS GREATER THAN ALLOWABLE?
6055 026976 101402 BLOS ; ;NO
6056 026980 000137 016312 JMP ;DSELCT ;DROP THE DRIVE
6057
6058 026984 012601 28: MOV (SP)+,R1 ;RESTORE R1
6059
6060 026986 011637 001116 38: MOV (SP),ERRPC ;GET ADDRESS OF ERROR INSTRUCTION
6061 026912 152737 000002 001116 SUB ;#2,ERRPC
6062 026920 005046 CLR ;-(SP)
6063 026922 117716 152570 MOV ;ERRPC,(SP) ;STRIP AND SAVE THE ERROR ITEM CODE
6064 026926 121627 000100 CMPB ;(SP),#100 ;FORM THE CO&ECT ITEM# IF THIS IS AN
6065 026932 002402 BIT ; ;EROR MESSAGE EQUAL OR ABOVE 100,
6066 026934 162716 000040 SUB ;#40,(SP) ;NOTE THERE R 2 CLASSES OF ERRORS:
;1) ITEM#S BELOW 100
;2) ITEM#S ABOVE 100
;SUBTRACTION FACTOR HAS TOBE SUCH THAT
;THE CO&ECT OFFSET IS SELECTED, THIS
;FACTOR WILL CHANGE IF THE TOTAL # OF
;ERROR MESSAGES IN CLASS 1 CHANGES,
;#100 -LAST ITEM IN # CLASS 1 - 1
6067
6068
6069
6070
6071
6072
6073

```

```

6074 026940 012637 001114 48: MOV (SP)+,ITEMB ;SKIP TYPEOUT IF SET
6075 026944 032777 020000 152566 BIT ;SW13,#SWR ;SKIP TYPEOUTS
6076 026952 001012 BNE ; ;SKIP TYPEOUTS
6077 026954 004737 026472 JSR ;PC,##ERRTP ;GO TO USER ERROR ROUTINE
6078 026960 104401 001213 TYPE ; ;SCRLF
6079
6080 026964 032777 010000 152546 BIT ;SW12,#SWR ;TYPE ERROR HISTORY?
6081 026972 001402 BEQ ; ;NO
6082 026974 004737 020426 JSR ;PC,HISTRY ;YES
6083
6084 026400 023737 000042 000046 58: 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 ; ;SKIP IF CONTINUE
6088 026416 000000 HALT ; ;HALT ON ERROR
6089 026420 104407 CK&WR ;LOOK FOR A 'CONTROL G'
6090 026422 032777 001000 152510 68: BIT ;SW09,#SWR ;LOOP ON ERROR SWITCH SET?
6091 026430 001411 BEQ ; ;BR IF NO
6092 026432 123727 001114 000040 CMPB ;ITEMB,#40 ;THERE R 37 ERROR MESSAGES IN CLASS 1
6093 026440 103011 BHIS ; ;
6094 026442 013746 001244 MOV ;PPRLVL,=(SP) ;LOCK OUT ALL INTERRUPTS ON RETURN
;FROM THIS EROR HANDLER, IF THE EROR
;IS IN EXERCISER & LOOPING IS TO
;BE DONE
6095
6096
6097
6098 026446 005726 TST (SP)+
6099 026450 012716 015726 MOV ;EXCLUP,(SP) ;IF THIS ERROR CALL WAS FROM EXERCISER
;PART OF THE PROGRAM, GO TO 'EXCLUP'
;OTHERWISE RETURN THRU '#LUPERR'
6100
6101
6102
6103 026454 012737 177777 001250 78: MOV #-1,SRDRV ;RESET SERIAL NO FLAG
6104 ; ;IF (SRDRV)=-1, THEN THE SERIAL
6105 ;NO OF THE DRIVE WILL NOT BE
6106 ;TYPED OUT.
6107 ;OTHERWISE, SERIAL NO FOR THE DRIVE
6108 ; IN 'SRDRV' WILL BE TYPED.
6109 026462 000002 RTI
6110 026464 013716 001110 88: MOV ;LPERR,(SP) ;FUDGE RETURN FOR LOOPING
6111 026470 000771 BR ; ;RETURN

```

```

6112          ,SBTIL ERROR MESSAGE TYPEOUT ROUTINE
6113
6114          ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" (ITEMB) TO DETERMINE WHICH
6115          ;*ERROR IS TO BE REPORTED, IT THEN OBTAINS, FROM THE "ERROR TABLE" (ERRTB),
6116          ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
6117
6118
6119 026472 104401 001213      SERRTY:TYPE      ,SCLRF          ;"CARRIAGE RETURN" & "LINE FEED"
6120 026476 010046          MOV          RO,-(SP)      ;SAVE RO
6121 026400 005000          CLR          RO            ;PICKUP THE ITEM INDEX
6122 026402 193700          B1SB        @#ITEMB,RO     ;
6123 026406 001004          BNE         18            ;IF ITEM NUMBER IS ZERO, JUST
6124 026410 013746          MOV          $ERRPC,-(SP)   ;TYPE THE PC OF THE ERROR
6125 026414 104402          TYPOC
6126 026416 000428          BR          88            ;GET OUT
6127 026420 005300          18: DEC          RO        ;ADJUST THE INDEX SO THAT IT WILL
6128 026422 006300          ABL         RO            ;WORK FOR THE ERROR TABLE
6129 026424 006300          ABL         RO
6130 026426 006300          ABL         RO
6131 026430 062700          ADD          $ERRTB,RO     ;FORM TABLE POINTER
6132 026434 012037          MOV          (RO)+,28     ;PICKUP "ERROR MESSAGE" POINTER
6133 026440 001404          BEQ         48            ;SKIP TYPEOUT IF NO POINTER
6134 026442 104401          TYPE
6135 026444 000000          ,WORD         0          ;TYPE THE "ERROR MESSAGE"
6136 026446 104401          28: TYPE      ,SCLRF      ;"ERROR MESSAGE" POINTER GOES HERE
6137 026452 032777          38:          ,SW11,$SWR     ;"CARRIAGE RETURN" & "LINE FEED"
6138 026460 001404          BEQ         88            ;DUMP OUT RK REGISTERS?
6139 026462 004737          JSR         PC,DMPREG      ;GO TYPE OUT RK REGISTERS
6140 026466 104401          TYPE      ,SCLRF
6141
6142 026472 012037          88: MOV          (RO)+,88     ;PICKUP "DATA HEADER" POINTER
6143 026476 001404          BEQ         78            ;SKIP TYPEOUT IF 0
6144 026400 104401          TYPE
6145 026402 000000          ,WORD         0          ;TYPE THE "DATA HEADER"
6146 026404 104401          68: TYPE      ,SCLRF      ;"DATA HEADER" POINTER GOES HERE
6147 026410 011000          78: MOV          (RO),RO    ;"CARRIAGE RETURN" & "LINE FEED"
6148 026412 001406          BEQ         98            ;PICKUP "DATA TABLE" POINTER
6149
6150
6151 026414 013048          88: MOV          @ (RO)+,-(SP) ;TYPE AN OCTAL NUMBER
6152 026416 104402          TYPOC          ;SAVE @ (RO)+ FOR TYPEOUT
6153 026420 104401          002663        TYPE      ,BLNK$2        ;GO TYPE=OCTAL ASCII(ALL DIGITS)
6154 026424 008710          TST         (RO)          ;TYPE TWO(2) SPACES
6155 026426 001372          BNE         88            ;IS THERE ANOTHER NUMBER?
6156
6157 026430 012600          98: MOV          (SP)+,RO    ;RESTORE RO
6158 026432 104401          TYPE      ,SCLRF      ;"CARRIAGE RETURN" & "LINE FEED"
6159 026436 123727          001114 000040  CMPB        $ITEMB,$40    ;SKIP TIME & SERIAL $ FOR
6160
6161 026444 103004          BHS         108          ;NON-EXERCISER ERRORS
6162 026446 004737          026660        JSR         PC,TIMTYP     ;TYPE OUT TIME IF SW 3 IS SET
6163 026452 004737          026766        JSR         PC,$NOTYP     ;GO TYPE OUT THE SERIAL $ OF THE
6164
6165 026456 000207          108: RTS         PC       ;ERRORING DRIVE, IF SW 1 IS SET.
6166
6167
6168
6169

```

```

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 KW11 IS NOT PRESENT.
6169
6170 026460 032777 000010 152252 TIMTYP: BIT      $SW3,$SWR      ;IS SW 3 SET?
6171 026466 001434          BEQ         48            ;IF NOT SKIP TYPING TIME
6172 026470 104401 002652          TYPE      ,M$C29        ;"TIME"
6173 026474 104401 002662          TYPE      ,BLNK$3
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          28            ;
6178 026712 012001          18: MOV          (RO)+,R1   ;TYPE OUT
6179 026714 001402          BEQ         28            ;
6180 026716 062701 000074          ADD          $60,,R1      ;
6181 026722 010137 026762          28: MOV          R1,$8       ;
6182 026726 012746 026762          MOV          $54,-(SP)    ;HOURS:MIN:SECS
6183 026732 004737 024604          JSR         PC,$$SDB2D   ;CONVERT TO ASCII STRING
6184 026736 004737 025000          JSR         PC,$$SUPR$L  ;GO TYPE
6185 026742 020027 001560          CMP         RO,$KN$SEC*2 ;ALL DONE?
6186 026746 001403          BEQ         38            ;
6187 026750 104401 002334          TYPE      ,M$G18
6188 026754 000756          BR          18            ;
6189 026756 104415          38: RESREG
6190 026760 000207          48: RTS         PC       ;RESTORE RO=R4
6191 026762 000000          58:          ,WORD         0,0 ;RETURN

```

```

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 28 ;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 18 ;THE ERROR WAS NOT ATTRIBUTABLE
6205 ;TO A SPECIFIC DRIVE)
6206 027010 104401 001213 TYPE ,%CRLF ;
6207 027014 104401 002326 TYPE ,MSG17 ;TYPE "SR. NO: "
6208 027020 016146 001266 MOV BRNO(R1),-(SP) ;GET THE SERIAL #
6209 027024 104405 TYPDS ;TYPE IT OUT (DECIMAL)
6210
6211 027026 012601 18: MOV (SP)+,R1 ;RESTORE R1
6212 027030 000207 28: 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 ASCII STRING
6221 027036 000441 BR 648 ;GET OVER THE ASCII
6222 ;55$! ,ASCII <15><12>/ PC RKDS RKER RKC5 RKC6 RKC7 RKC8 RKC9 RKC0
6223 027142 648: MOV $ERRPC,=(SP)
6224 027142 013746 001116 TYPDC
6225 027146 104402 TYPE ,BLNK$2
6226 027150 104401 002663 MOV RO,=(SP)
6227 027154 010046 MOV $RKDS,RO
6228 027156 012700 001216 18: MOV $(RO)+,=(SP)
6229 027162 013046 TYPDC
6230 027164 104402 TYPE ,BLNK$2
6231 027166 104401 002663 CMP RO,$RKDB
6232 027172 020027 001232 BLE 18
6233 027176 003771 MOV (SP)+,RO
6234 027000 012600 RTS PC
6235 027002 000207
    
```

```

6236 ;SBTTL SCOPE HANDLER ROUTINE
6237
6238 ;*****
6239 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS, IT WILL INCREMENT
6240 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7;0>)
6241 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15;00>
6242 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
6243 ;*SW14=1 LOOP ON TEST
6244 ;*SW09=1 LOOP ON ERROR
6245 ;*CALL
6246 ;* SCOPE ;SCOPE=IOT
6247
6248 027004 $SCOPE:
6249 027004 104407 BIT CKSWR ;TEST FOR CHANGE IN SOFT-SWR
6250 027006 032777 040000 151724 18: BIT $BIT14,@SWR ;LOOP ON PRESENT TEST?
6251 027014 001047 SNE $OVER ;YES IF SW14=1
6252 ;###START OF CODE FOR THE XOR TESTER###
6253 027016 000416 $XTSTR: BR 68 ;IF RUNNING ON THE "XOR" TESTER CHANGE
6254 ;THIS INSTRUCTION TO A "NOP" (NOP=240)
6255 027220 013746 000004 MOV $ERRVEC,=(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
6256 027224 012737 027244 000004 MOV $177060 ;SET FOR TIMEOUT
6257 027232 005737 177060 TST $177060 ;TIME OUT ON XOR?
6258 027236 012637 000004 MOV (SP)+,$ERRVEC ;RESTORE THE ERROR VECTOR
6259 027242 000421 BR $SVLAD ;GO TO THE NEXT TEST
6260 027244 022826 88: CMP (SP)+,(SP)+ ;CLEAR THE STACK AFTER A TIME OUT
6261 027246 012637 000004 MOV (SP)+,$ERRVEC ;RESTORE THE ERROR VECTOR
6262 027252 000407 BR 78 ;LOOP ON THE PRESENT TEST
6263
6264 68: ;###END OF CODE FOR THE XOR TESTER###
6265 027254 105737 28: TSTB $ERFLG ;HAS AN ERROR OCCURRED?
6266 027260 001412 BEQ $SVLAD ;BR IF NO
6267 027262 032777 001000 151650 BIT $BIT09,@SWR ;LOOP ON ERROR?
6268 027270 001404 BEQ 48 ;BR IF NO
6269 027272 013737 001110 001106 78: MOV $LPERR,$LPADR ;SET LOOP ADDRESS TO LAST SCOPE
6270 027302 105037 BR $OVER
6271 027306 105237 001102 48: CLRB $ERFLG ;ZERO THE ERROR FLAG
6272 027312 011637 001106 $SVLAD: INCB $TSTNM ;COUNT TEST NUMBERS
6273 027316 011637 001110 MOV (SP), $LPADR ;SAVE SCOPE LOOP ADDRESS
6274 027322 005037 001204 MOV (SP), $LPERR ;SAVE ERROR LOOP ADDRESS
6275 027326 112737 000001 001115 CLR $ESCAPE ;CLEAR THE ESCAPE FROM ERROR ADDRESS
6276 027334 013777 001102 151600 $OVER: MOV #1,$ERRMAX ;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
6277 027342 013716 001106 MOV $TSTNM,$DISPLAY ;DISPLAY TEST NUMBER
6278 027346 000002 MOV $LPADR,(SP) ;FUJGE RETURN ADDRESS
        RTI ;FIXES P8
    
```

```

6279          ,SBTTL  TRAP DECODER
6280
6281          ;*****
6282          ;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
6283          ;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
6284          ;OF THE DESIRED ROUTINE, THEN USING THE ADDRESS OBTAINED IT WILL
6285          ;GO TO THAT ROUTINE.
6286
6287 077450 010046          @TRAP:  MOV  R0,=(SP)          ;;SAVE R0
6288 077452 016600 000002  MOV  2(SP),R0          ;;GET TRAP ADDRESS
6289 077456 005740          TST  =(R0)              ;;BACKUP BY 2
6290 077460 111000          MOVB (R0),R0          ;;GET RIGHT BYTE OF TRAP
6291 077462 006300          ASL  R0                ;;POSITION FOR INDEXING
6292 077464 016000 027404  MOV  @TRPAD(R0),R0     ;;INDEX TO TABLE
6293 077470 000200          RTS   R0              ;;GO TO ROUTINE
6294
6295
6296          ;THIS IS USE TO HANDLE THE "GETPRI" MACRO
6297
6298 077472 011646          @TRAP2: MOV  (SP),=(SP)          ;;MOVE THE PC DOWN
6299 077474 016666 000004 000002  MOV  4(SP),2(SP)       ;;MOVE THE PSW DOWN
6300 077402 000002          RTI                    ;;RESTORE THE PSW
6301
6302          ,SBTTL  TRAP TABLE
6303
6304          ;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
6305          ;BY THE "TRAP" INSTRUCTION.
6306
6307          ;
6308          ; ROUTINE
6309          ;-----
6309 027404 027372          @TRPAD:  ,WORD  @TRAP2
6310 027406 024244          $TYPE  ;;CALL=TYPE          TRAP+1(104401)  TTY TYPEOUT ROUTINE
6311 027410 026004          $TYPOC ;;CALL=TYPOC        TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
6312 027412 025760          $TYPOS ;;CALL=TYPOS        TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
6313 027414 026020          $TYPON ;;CALL=TYPON        TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
6314 027416 024020          $TYPDS ;;CALL=TYPDS        TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
6315
6316 027420 023034          @GTSWR  ;;CALL=GTSWR        TRAP+6(104406)  GET SOFT-SWR SETTING
6317
6318 027422 022764          @CKSWR  ;;CALL=CKSWR        TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
6319 027424 023246          @RDCHR  ;;CALL=RDCHR        TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
6320 027426 023366          @RDLIN  ;;CALL=RDLIN        TRAP+11(104411) TTY TYPEIN STRING ROUTINE
6321 027430 023540          @RDCT   ;;CALL=RDCT        TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
6322 027432 023642          @RDDEC  ;;CALL=RDDEC        TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY
6323 027434 025502          @SAVREG ;;CALL=SAVREG        TRAP+14(104414) SAVE R0-R5 ROUTINE
6324 027436 028540          @RESREG ;;CALL=RESREG        TRAP+15(104415) RESTORE R0-R5 ROUTINE
6325 027440 022386          CN,RST  ;;CALL=CON,RESET    TRAP+16(104416) CONTROL RESET ROUTINE
6326 027442 023364          CN,RDY  ;;CALL=CON,RDY      TRAP+17(104417) WAIT FOR CONTROL READY
6327 027444 022744          DR,RST  ;;CALL=DRV,RESET    TRAP+20(104420) DRIVE RESET ROUTINE
6328 027446 022522          TY,M&G ;;CALL=TYPM&G         TRAP+21(104421) TYPE MESSAGE ROUTINE, SW13
6329
    
```

```

6330          ,SBTTL  POWER DOWN AND UP ROUTINES
6331
6332          ;*****
6333          ;POWER DOWN ROUTINE
6334 027480 012737 027614 000024 @PWRDN:  MOV  @ILLUP,@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 027400 017746 151434          MOV  @SWR,=(SP)        ;;PUSH @SWR ON STACK
6343 027404 010637 027620          MOV  SP,@SAVR6         ;;SAVE SP
6344 027410 012737 027522 000024  MOV  @PWRUP,@PWRVEC   ;;SET UP VECTOR
6345 027416 000000          HALT
6346 027420 000776          BR   ,=2              ;;HANG UP
6347
6348          ;*****
6349          ;POWER UP ROUTINE
6350 027422 012737 027614 000024 @PWRUP:  MOV  @ILLUP,@PWRVEC ;;SET FOR FAST DOWN
6351 027430 013706 027620          MOV  @SAVR6,SP        ;;GET SP
6352 027434 005037 027620          CLR  @SAVR6           ;;WAIT LOOP FOR THE TTY
6353 027440 005237 027620          16:  INC  @SAVR6           ;;WAIT FOR THE INC
6354 027444 001375          SNE  16              ;;OF WORD
6355 027446 012677 151366          MOV  (SP)+,@SWR       ;;POP STACK INTO @SWR
6356 027452 012605          MOV  (SP)+,R5         ;;POP STACK INTO R5
6357 027454 012604          MOV  (SP)+,R4         ;;POP STACK INTO R4
6358 027456 012603          MOV  (SP)+,R3         ;;POP STACK INTO R3
6359 027460 012602          MOV  (SP)+,R2         ;;POP STACK INTO R2
6360 027462 012601          MOV  (SP)+,R1         ;;POP STACK INTO R1
6361 027464 012600          MOV  (SP)+,R0         ;;POP STACK INTO R0
6362 027466 012737 027450 000024  MOV  @PWRDN,@PWRVEC   ;;SET UP THE POWER DOWN VECTOR
6363 027474 012737 000340 000026  MOV  @340,@PWRVEC*2    ;;PRIO:7
6364 027602 104401          TYPE  ;;REPORT THE POWER FAILURE
6365 027404 027622          @PWRNG: ,WORD  @PWRNG   ;;POWER FAIL MESSAGE POINTER
6366 027406 012716          MOV  (PC)+,(SP)       ;;RESTART AT PFBTRT
6367 027410 003366          @PWRAD: ,WORD  PFBTRT  ;;RESTART ADDRESS
6368 027412 000002          RTI
6369 027414 000000          @ILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
6370 027416 000776          BR   ,=2              ;; BEFORE THE POWER DOWN WAS COMPLETE
6371 027420 000000          @SAVR6: 0            ;;PUT THE SP HERE
6372 027422 005015 047820 042527  @PWRNG:  ,ASCIZ <15><13>"POWER"
6373 027430 000122
6374          ,EVEN
    
```

						ERROR MESSAGES
6375						
6376						
6377	027432	051105	051117	047440	EM11	.ASCIZ /ERROR ON WRITE/
6378	027440	020116	051127	052111		
6379	027446	000105				
6380	027450	052101	048503	052120	EM21	.ASCIZ /ATTEMPT TO INITIATE FUNCTION ON 'BUSY' DRIVE/
6381	027456	032040	020117	047111		
6382	027464	052111	040511	042824		
6383	027472	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	EM31	.ASCIZ /CONTROL RDY NOT SET/
6389	027730	020114	042122	020131		
6390	027736	047516	020124	042523		
6391	027744	000124				
6392	027746	051057	053457	051457	EM41	.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	EM51	.ASCIZ /CONTROL RDY NOT SET AFTER 1ST INTRUPT ON ISSUING SEEK/
6397	027776	020114	042122	020131		
6398	030104	047516	020124	042523		
6399	030112	020124	043101	042524		
6400	030120	020122	051461	020124		
6401	030126	047111	051124	050125		
6402	030134	020124	047117	044440		
6403	030142	051523	044525	043516		
6404	030150	051440	042505	000113		
6405	030156	051127	047117	020107	EM61	.ASCIZ /WRONG BITS IN RKCS, EXPCT SEEK/
6406	030164	044502	051524	044440		
6407	030172	020116	045522	051503		
6408	030100	020054	054105	041520		
6409	030106	020124	042523	045505		
6410	030114	000				
6411	030115	047	052502	054523	EM71	.ASCIZ /'BUSY' FLAG CLEAR ON INTERRUPTING 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	EM101	.ASCIZ /'POSITIONING' FLAG FOR INTERRUPTING 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	EM111	.ASCIZ /'ERROR 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	030202	051511	052523	047111		
6433	030210	020107	042523	045505		
6434	030216	000				
6435	030217	123	050103	051440	EM121	.ASCIZ /SCP SET AFTER 1ST INTRUPT ON ISSUING SEEK/
6436	030224	052105	040440	052106		
6437	030232	051105	030440	052123		
6438	030240	044440	052116	052522		
6439	030246	052120	047440	020116		
6440	030254	051511	052523	047111		
6441	030262	020107	042523	045505		
6442	030270	000				
6443	030271	103	052116	047522	EM131	.ASCIZ /CONTROL RDY NOT SET AFTER SEEK DONE INTRUPT/
6444	030276	020114	042122	020131		
6445	030204	047516	020124	042523		
6446	030212	020124	043101	042524		
6447	030220	020122	042523	045505		
6448	030226	042040	047117	020105		
6449	030234	047111	051124	050125		
6450	030242	000124				
6451	030244	047111	051124	050125	EM141	.ASCIZ /INTRUPTING DRIVE (SEEK DONE) WAS NOT 'BUSY'/
6452	030252	044524	043516	042040		
6453	030260	053122	020105	051450		
6454	030266	042505	020113	047504		
6455	030274	042516	020051	040527		
6456	030202	020123	047516	020124		
6457	030210	041047	051525	023531		
6458	030216	000				
6459	030217	122	053457	051457	EM151	.ASCIZ "/R/W/S READY NOT SET FOR INTERRUPTING DRIVE (SEEK DONE)"
6460	030224	051040	040505	054504		
6461	030232	047040	052117	051440		
6462	020240	052105	043040	051117		
6463	030246	044440	052116	052522		
6464	030254	052120	047111	020107		
6465	030262	051104	042526	024040		
6466	030270	042523	045505	042040		
6467	030276	047117	024505	000		
6468	030203	123	047111	042440	EM161	.ASCIZ /SIN ERROR/
6469	030210	047522	000122			
6470	030214	042447	051122	047447	EM171	.ASCIZ /'ERROR ON DOING SEEK/
6471	030222	020122	047117	042040		
6472	030230	044517	043516	051440		
6473	030236	042505	000113			
6474	030242	041523	020120	044504	EM201	.ASCIZ /SCP DID NOT SET AFTER SEEK WAS DONE/
6475	030250	020104	047516	020124		
6476	030256	042523	020124	043101		
6477	030264	042524	020122	042523		
6478	030272	045505	053440	051501		
6479	030700	042040	047117	000105		
6480	030706	047523	052106	042440	EM211	.ASCIZ /SOFT ERROR/
6481	030714	047522	000122			
6482	030720	040504	040524	024040	EM231	.ASCIZ /DATA (COMPARISON) ERROR/
6483	030726	047503	050115	051101		
6484	030734	051511	047117	020051		
6485	030742	051105	051117	000		
6486	030747	103	052116	047522	EM241	.ASCIZ /CONTROL RDY CLR ON INTRUPT AFTER RK FUNCTION/

6487	030754	020114	042122	020131			
6488	030762	046103	020122	047117			
6489	030770	044440	052116	052522			
6490	030776	057120	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 SHLDSB 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 DC WRITE BEFORE WRT CNK/
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	054108	041508	052125			
6511	031160	020105	047503	046515			
6512	031166	047101	026504	047111			
6513	031174	030040	047522	051107			
6514	031902	051505	020123	051117			
6515	031910	040440	051114	040505			
6516	031916	054504	043040	047111			
6517	031924	051511	042510	000104			
6518	031932	043047	047125	052103	EM31:	.ASCIZ	/FUNCTION IN PROGRES' FLG FOR INTRUPTING DRIVE ISN'T SET/
6519	031940	047511	020116	047111			
6520	031946	050040	047522	051107			
6521	031954	051505	020047	046106			
6522	031962	020107	047506	020122			
6523	031970	047111	051124	050125			
6524	031976	044524	043516	042040			
6525	031984	044522	042526	044440			
6526	031912	047123	052047	051440			
6527	031920	052105	000				
6528	031923	125	042516	050130	EM32:	.ASCIZ	/UNEXPCED DRIVE INTRUPTED/
6529	031930	052103	042105	042040			
6530	031936	044522	042526	044440			
6531	031944	052116	052522	052120			
6532	031952	042105	000				
6533	031955	125	042516	050130	EM33:	.ASCIZ	/UNEXPCD FUNCTION CODE IN RK05 AFTER INTRUPT/
6534	031962	052103	020104	052506			
6535	031970	041516	044524	047117			
6536	031976	041440	042117	020105			
6537	031984	047111	051040	041513			
6538	031912	020123	043101	042524			
6539	031920	020122	047111	051124			
6540	031926	050125	000124				
6541	031932	051104	042526	051040	EM34:	.ASCIZ	/DRVE RDY CLEAR/
6542	031940	054504	041440	042514			

6543	031466	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	031402	000					
6550	031403	127	051520	051440	EM37:	.ASCIZ	/WPS SET/
6551	031410	052105	000				
6552	031413	111	052116	051105	EM101:	.ASCIZ	/INTERUPT DIDN'T OCUR AFTER WRTE/
6553	031420	050125	020124	044504			
6554	031426	047104	052047	047440			
6555	031434	052503	020122	043101			
6556	031442	042524	020122	051127			
6557	031450	042524	000				
6558	031453	047	051105	023522	EM102:	.ASCIZ	/ERR'OR SET/
6559	031460	051117	051440	052105			
6560	031466	000					
6561	031467	122	042113	020101	EM103:	.ASCIZ	/RKDA INCRMENTED WRONG/
6562	031474	047111	051103	042515			
6563	031402	052116	042105	053440			
6564	031410	047522	043516	000			
6565	031415	122	041113	020101	EM104:	.ASCIZ	/RKBA INCRMENTED WRONG/
6566	031422	047111	051103	042515			
6567	031430	052116	042105	053440			
6568	031436	047522	043516	000			
6569	031443	122	053513	020103	EM105:	.ASCIZ	/RKWC DIDN'T OVRFLD TO 0/
6570	031450	044504	047104	052047			
6571	031456	047440	051126	046106			
6572	031464	020117	047524	030040			
6573	031472	000					
6574	031473	119	054105	041040	EM106:	.ASCIZ	/MEX BITS WRONG/
6575	031700	052111	020123	051127			
6576	031706	047117	000107				
6577	031712	051127	042524	041440	EM110:	.ASCIZ	/WRTE CNK EROR/
6578	031720	045510	042440	047522			
6579	031726	000122					


```

6580 ;DATA HEADERS
6581 031730 020040 041520 020040 DH1: ,ASCIZ / PC RKCS RKER RKDS RKDA/
6582 031736 020040 051040 041513
6583 031744 020123 020040 051040
6584 031752 042513 020122 020040
6585 031760 051040 042113 020123
6586 031766 020040 051040 042113
6587 031774 000101
6588
6589 031776 020040 041520 020040 DH2: ,ASCIZ / PC DRV#/
6590 032004 020040 051104 021526
6591 032012 000
6592
6593 032013 040 050040 020103 DH21: ,ASCIZ / PC RKCS RKER RKDS DRIVE CYL SUR SEC/
6594 032020 020040 020040 045522
6595 032026 051503 020040 020040
6596 032034 045522 051105 020040
6597 032042 020040 045522 051504
6598 032050 020040 042040 044522
6599 032056 042526 020040 020040
6600 032064 054503 020114 020040
6601 032072 020040 052523 020122
6602 032100 020040 020040 042523
6603 032106 000103
6604
6605 032110 020040 041520 020040 DH23: ,ASCIZ / PC RKBA EXPCT RECVD RKDA/
6606 032116 020040 045522 040502
6607 032124 020040 020040 054105
6608 032132 041520 020124 020040
6609 032140 042522 053103 020104
6610 032146 020040 045522 040504
6611 032154 000
6612 032155 040 050040 020103 DH25: ,ASCIZ / PC RKCS RKER RKDS RKDA DRIVE#/
6613 032162 020040 020040 045522
6614 032170 051503 020040 051040
6615 032176 042513 020122 020040
6616 032004 051040 042113 020123
6617 032012 020040 051040 042113
6618 032020 020101 020040 051104
6619 032026 042526 000043
6620
6621 032032 020040 041520 020040 DH27: ,ASCIZ / PC KEY FNCIN CODE/
6622 032040 020040 045440 054505
6623 032046 020040 020040 047106
6624 032054 052103 020116 047503
6625 032062 042504 000
6626 032065 040 050040 020103 DH103: ,ASCIZ / PC EXPCT RECVD/
6627 032072 020040 042440 050130
6628 032000 052103 020040 051040
6629 032006 041505 042126 000
6630
6631 032113 040 050040 020103 DH30: ,ASCIZ / PC KEY/
6632 032120 020040 045440 054505
6633 032126 000
6634 032127 040 050040 020103 DH105: ,ASCIZ / PC RKDA RKNC/
6635 032134 020040 020040 045522
    
```

```

6636 032142 040504 020040 051040
6637 032150 053513 000103
6638 032154 020040 041520 020040 DH110: ,ASCIZ / PC RKCS RKER RKBA RKDA/
6639 032162 020040 051040 041513
6640 032170 020123 020040 051040
6641 032176 042513 020122 020040
6642 032104 051040 041113 020101
6643 032112 020040 051040 042113
6644 032120 000101
6645
6646
6647 ,EVEN
6648
6649 032122 001116 001162 001164 DT1: ,WORD $ERRPC,$REG0,$REG1,$REG2,$REG3,0
6650 032130 001166 001170 000000
6651
6652 032136 001116 001162 000000 DT2: ,WORD $ERRPC,$REG0,0
6653
6654 032144 001116 001162 001164 DT21: ,WORD $ERRPC,$REG0,$REG1,$REG2,$REG3,$REG4,$REG5,$REG6,0
6655 032152 001166 001170 001172
6656 032160 001174 001176 000000
6657 032166 001116 001162 001164 DT25: ,WORD $ERRPC,$REG0,$REG1,$REG2,$REG3,$REG4,0
6658 032174 001166 001170 001172
6659 032102 000000
6660 032104 001116 001162 001164 DT103: ,WORD $ERRPC,$REG0,$REG1,0
6661 032112 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/% WORDS LONG.
6666
6667 032114 DBUF:
6668 032116 PGEND: NOP
6669 000001 ,END
    
```

ABORT 001672	CICNT1 001470	DSELC7 016312	EXRC8W 007772	KIPDR5 172312
ABRT 012046	CKSWR 104407	DSWR 177570	FNCHND 020372	KIPDR6 172314
ABRT1 012060	CLNAPR 006434	D11 032422	FNMAP 001526	KIPDR7 172316
BAMAP 001530	CLREPP 016006	DT103 032504	FRDIRT 001253	KWCOUN 001500
BASEBA 002052	CLRFLG 015770	DT2 032436	FTITLF 001252	KWHR 001952
BC787 005352	CLRSIG 016192	DT21 032444	GENMUF 016704	KWLS 001234
BEGEX1 010612	CMNART 014216	DT25 032466	GEN1 014724	KWLVEG 000100
BEGNEX 010630	CMND 001326	ECOUNT 001540	GENBRQ 014500	KWMTN 001544
BITO 000001	CMNERR 014160	EMTVECC 000030	GETIMF 022064	KWPLVL 001246
BIT00 000001	CNOBBY 021712	EM1 027632	GTSDRV 022226	KWDFC 001856
BIT01 000002	CN,PDY 022364	EM10 030162	GTWR 104406	KWBRVE 022552
BIT02 000004	CN,RST 022396	EM101 031813	GT3RG 022040	LF 000012
BIT03 000010	COMRET 020306	EM102 031853	GT4RG 022032	MAXBA 002054
BIT04 000020	CON,RD 104417	EM103 031867	HE 040000	MH1 020746
BIT05 000040	CON,RE 104416	EM104 031615	HECN 001562	MH2 020762
BIT06 000100	CR 000015	EM105 031643	HISTRY 020426	MH3 020772
BIT07 000200	CRCMND 020340	EM106 031673	HPDERR 014016	MH4 020775
BIT08 000400	CRLF 000200	EM11 030237	HT 000011	MNVEC 000250
BIT09 000800	CROTFL 022170	EM110 031712	INTFLG 001534	M8G1 022062
BIT1 000002	CSE 000002	EM12 030317	INTHND 013256	M8G10 002165
BIT10 000200	CSECN 001652	EM13 030371	INTIFL 001535	M8G11 002201
BIT11 000400	CYLMAP 001522	EM14 030444	INT18K 012376	M8G12 002206
BIT12 010000	DATCHK 017110	EM15 030517	IOTVEC 000020	M8G13 002214
BIT13 020000	DATER 001712	EM16 030603	KDPAR0 172360	M8G14 002225
BIT14 040000	DBUF 032514	EM17 030614	KDPAR1 172362	M8G15 002245
BIT15 100000	DDISP 177570	EM2 027650	KDPAR2 172364	M8G16 002305
BIT2 000004	DH1 031730	EM20 030642	KDPAR3 172366	M8G17 002326
BIT3 000010	DH103 032265	EM21 030706	KDPAR4 172370	M8G18 002334
BIT4 000020	DH105 032327	EM23 030720	KDPAR5 172372	M8G19 002336
BIT5 000040	DH110 032354	EM24 030747	KDPAR6 172374	M8G2 002070
BIT6 000100	DH2 031776	EM26 031023	KDPAR7 172376	M8G20 002362
BIT7 000200	DH21 032013	EM27 031076	KDPR0 172320	M8G24 002372
BIT8 000400	DH23 032110	EM3 027723	KDPR1 172322	M8G25 002375
BIT9 001000	DH25 032155	EM30 031137	KDPR2 172324	M8G26 002377
BLNKS1 002664	DH27 032232	EM31 031232	KDPR3 172326	M8G26A 002474
BLNKS2 002663	DH30 032313	EM32 031323	KDPR4 172330	M8G27 002532
BLNKS3 002662	DJSPLA 001142	EM33 031355	KDPR5 172332	M8G28 002604
BPTVEC 000014	DISPRE 000174	EM34 031432	KDPR6 172334	M8G29 002652
BUSY 001426	DMPREG 027032	EM35 031481	KDPR7 172336	M8G3 002076
CB5Y 021474	DORFAD 006424	EM36 031467	KEY 001306	M8G4 002104
CHDPRS 022456	DOWRT 006304	EM37 031503	KIPAR0 172340	M8G5 002120
CHFAFN 011154	DOXFER 006312	EM4 027746	KIPAR1 172342	M8G6 002131
CHKBA 020150	DPL 010000	EM5 027771	KIPAR2 172344	M8G7 002141
CHKCRD 020272	DRCMND 020314	EM6 030056	KIPAR3 172346	M8G8 002146
CHKCS 020112	DRMAP 001520	EM7 030115	KIPAR4 172350	M8G9 002156
CHKDA 020126	DRU 002000	ERCODE 001474	KIPAR5 172352	NIEROR 021516
CHKDPV 016440	DRVABT 014426	ERDRV 001542	KIPAR6 172354	NOEROR 014300
CHKMEX 020204	DRVANT 001500	ERINF1 005720	KIPAR7 172356	NRDH 001774
CHKRWS 020254	DRVPRS 001264	ERR 100000	KIPDR0 172300	NRDL 001772
CHKWC 020230	DRVPR 001476	ERRVEC 000004	KIPDR1 172302	NWFNC 011766
CH1 011210	DRV,RE 104420	EXCRU 015726	KIPDR2 172304	NWRTH 001734
CH2 011210	DRY 000200	EXCUT 011612	KIPDR3 172306	NWRTL 001732
CICNT 001466	DR,RST 022244	EXNSK 011470	KIPDR4 172308	NXTDRV 005366

ODDEVN 015724	RG48R 022222	ST3 005322	TYPM8G 104421	GGDAT 001124
PCMND 002032	RRA 001226	ST4 004752	TYPC 104402	GGT42 022740
PCNTP 001236	RKCS 001222	SUPR 025014	TYPN 104404	GGTSWR 023034
PDR 001254	RKDA 001230	SUPRSL 025000	TYPS 104403	SHD 000000
PFSTHT 003366	RKDB 001232	SUP1 025026	TY,MSG 022522	SHOCT 023640
PFTERR 013714	RKDS 001216	SUP2 025070	WATIME 022630	SICNT 001104
PGEND 032414	RKER 001220	SWR 001140	WCE 000001	SILLUP 027614
PIRO 177772	RKSTAT 001242	SWREG 000176	WCECN 001532	SINTAG 001135
PIRGVE 000240	RKVEC 001240	SWO 000001	WCFLG 001456	SITEMB 001114
POKREP 013222	RKWC 001224	SWO0 000001	WCMAP 001532	SKTNE 017666
POS 001436	RNUM 017106	SWO1 000002	WPS 000040	SKTOU 017656
POSCMN 020150	RSBAH 025746	SWO2 000004	WRD6K 010266	SKT11 017524
POSK 012164	RSBAL 025744	SWO3 000010	WRFNC 012064	SLF 001214
POSTIO 011414	RSCYLH 025742	SWO4 000020	XFR 007514	SLPADR 001106
PPRLVL 001744	RSCYLL 025740	SWO5 000040	XFR16K 007504	SLPERR 001110
PRICMN 011410	RSDRVH 025732	SWO6 000100	XXDPMD 002060	SLSTAD 017770
PRSFNC 001462	RSDRVL 025730	SWO7 000200	XAUTOB 001134	SLTBK 017772
PR0 000000	RSDTH 025756	SWO8 000400	XDDADR 001122	SMNEW 023527
PR1 000040	RSDTL 025754	SWO9 000100	XDDDAT 001126	SMBWR 023516
PR2 000100	RSPUNH 025736	SW1 000002	SEELL 001206	SMUL 000003
PR3 000140	RSPUNL 025734	SW10 002000	SCHARC 024460	SMULT 028100
PR4 000200	RSWCH 025752	SW11 004000	CKKWP 022764	SMULL 001154
PR5 000240	RSWCL 025750	SW12 010000	CKMTAC 001100	SMWT8 000001
PR6 000300	RTIPC7 021732	SW13 020000	CKM1 000011	SOCNT 026202
PP7 000340	RWS 000100	SW14 040000	CKM2 000022	SOCTVL 024566
PS 177776	R6 000006	SW15 100000	CKM3 000011	SONODE 026204
PSINER 013070	R7 000007	SW2 000004	CKNTLG 023511	COVER 027334
PSRTRY 013130	SAVKEY 001536	SW3 000010	CKNTLU 023504	SPAS8 001100
PSTFNC 001464	SAVREG 104414	SW4 000020	CKORE 017674	SPOWER 027622
PSW 177776	SCP 020000	SW5 000040	CKRLF 001213	SPWRAD 027610
PMRVEC 000024	SECMAP 001524	SW6 000100	CKROUT 017724	SPWRDN 027450
P1 020176	SEXIT 021720	SW7 000200	CKDLK 024234	SPWRMG 027604
P2 020410	SFTERR 013636	SW8 000400	CKB2D 024604	SPWRUP 027522
OBUSAD 001514	SIN 001000	SW9 001030	CKB2Q 024464	SQUE8 001212
OCYL 001904	SINCN 001622	TBITVE 000014	CKEVL 024764	SRAND 025576
ODRV 001502	SINCNT 016264	TIMER 001472	CKDIV 025212	SRDCH 023246
QERDR 021610	SKCMP 012704	TINTOUT 023354	CKDAGN 022760	SRDEEC 023642
QFNC 001512	SKE 010000	THTYTP 026660	CKDBL 024224	SRDLIN 023366
QMGGER 010550	SKECN 001602	TKVEC 000060	CKENDAD 021750	SRDOCT 023540
QSCNT 001460	SNOTYP 026760	TKVEC 000064	CKENDCT 022134	SRDAX 000010
QSEC 001510	SRDRV 001250	TRAPVE 000034	CKEOP 022704	SRGAD 001160
QSUB 001506	SRND 001266	TRTYVEC 000014	CKOPCT 022726	SRGO 001162
QWRCNT 001516	SRO 177572	TST1 005414	CKRFLG 001103	SRG1 001164
RCNT 017104	SR1 177574	TST2 005526	CKRMAX 001115	SRG10 001202
RDCNR 104410	SR2 177576	TST3 005572	CKERRR 026206	SRG2 001166
RDECC 104413	SR3 172516	TST4 006106	CKERRC 001116	SRG3 001170
RDLIN 104411	STACK 001100	TBT5 006486	CKERRB 002666	SRG4 001172
RDOCT 104412	START 003376	TBT6 007040	CKERTY 026472	SRG5 001174
REPCNT 002056	STARTA 003444	TBT7 007374	CKERTL 001112	SRG6 001176
REPBTA 011116	STATST 021006	TYDBO 017774	CKSCAP 001204	SRG7 001200
RESEGA 104415	STATUS 021436	TYDB 104403	CKFILL 001156	SRG8 025540
RESEVC 000010	STKLMT 177774	TYPE 104401	CKFILL 001153	SRTHAD 027622
RETRY 001446	ST2 004400	TYFFN 021736	CKGADR 001120	SAVRE 025502

\$SAVR6 027A0J	\$SVPC = 000220	\$TPB 001152	\$STNM 001102	\$TYPON 026020
\$SCOPE 027204	\$SWR = 143000	\$TPFLG 001157	\$TYIN 023474	\$TYP08 025760
\$SETUP= 000115	\$SWRMK= 000000	\$TPS 001150	\$TYPDS 024020	\$XTSTR 027216
\$SIZE 017466	\$TKB 001146	\$TRAP 027350	\$TYPE 024244	\$GET4= 000000
\$SIZEX 017730	\$TKS 001144	\$TRAP2 027372	\$TYPEC 024414	\$OFILL 026203
\$STUP = 177777	\$TN = 000010	\$TRP = 000022	\$TYPEX 024462	, = 032516
\$SVLAD 027A06	\$TNPWR 024714	\$TRPAD 027404	\$TYPC 026004	

. ABS. 032A16 000

ERRORS DETECTED: 0

,DSKZ:IDZPKHG/SDL=DSKZ:SYSMAC,SML,DSKM:IDZRRHG,P11
RUN-TIME: 15 22 ,6 SECONDS
RUN-TIME RATIO: 609/38=15.6
CORE USED: 36K (71 PAGES)