IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZRKH-F
PRODUCT NAME:	RK11/RK05 PERFORMANCE EXERCISER
DATE:	DECEMBER, 1976
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) 1974,1976 BY DIGITAL EQUIPMENT CORPORATION

TABLE OF CONTENTS

1.Ø ABSTRACT

THE RK11/RKØ5 PERFORMANCE EXERCISER IS A HIGH LEVEL EXERCISER PROGRAM AIMED AT SIMULATING A RK11/RKØ5 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.Ø 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 RKØ5 OR RKØ5F 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 POSSIABLE 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/RKØ5 DYNAMIC TESTS
C. RKØ5 UTILITY PACKAGE (IF NEEDED)

2.3 EXECUTION TIME

THIS VARIES FROM 3Ø TO 9Ø MINUTES FOR A PASS. IT SHOULD BE NOTED THAT THIS IS AN EXERCISER LEVEL PROGRAM AND SHOULD BE PREFERRABLY RUN FOR A LONG PERIOD OF TIME.

3.Ø STARTING ADDRESS

2ØØ - ALL SWITCHES DOWN. SEE SEC. 6.Ø FOR SWITCHES.

21Ø - 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-F

RK11/RK05 PERFORMANCE EXERCISER
```

THEN IT PROCEEDS TO TEST THE DRIVES.

4.2 RKDP DUMP MODE

4.2.1 THE PROGRAM IS LOADED BY THE RKDP MONITOR.

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

4.2.3 THE PROGRAM IDENTIFIES ITSELF AND PRINTS OUT:

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

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

4.3 RKDP CHAIN MODE

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

'DRIVE 'N' NOT TESTED'

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

4.4 ACT11 MODE

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

5.0 DRIVE SELECTION

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

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

6.0 SWITCH OPTIONS

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' whenever the program enters the scope routine or begins a new test. the 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

```
'SWR = NNNNNN    NEW ='
```

EACH TIME SWITCH SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED., 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY. ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

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

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

6.1 SW<15>

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

6.2 SW<13>

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

6.3 SW<12>

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

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

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

6.4 SW<11>

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

6.5 SW<09>

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

6.6 SW<08>

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

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

THE ERROR STATISTICS GIVE THE NUMBER OF ERRORS THAT HAVE OCCURRED

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

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

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

6.7 SW<Ø6>

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

```
TYPE OCTAL BUS ADDRESS FOR DATA XFER, BETWEEN XXXXXX AND YYYYYYY

    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 OCCURING 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<Ø5>

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'GEN8RQ' 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 2ØØ OR RESTARTED AT 21Ø.

6.9 SW<Ø4>

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 TYPEOUT 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 KW11L 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 PREFORMING OVERLAPPING SEEKS. WHILE SOME OF THE DRIVES ARE BEING POSITIONED, THE LAST AVAILABLE (AND EXECUTABLE) COMMAND IS PERFORMED. THUS WHILE SOME DRIVES ARE BUSY POSITIONING THEIR HEADS, SOME DRIVE IS PERFORMING A FUNCTION (DATA TRANSFER,ETC). AS SOON AS THE CONTROLLER IS FREE , A CHECK IS MADE TO SEE IF THERE IS ANY DRIVE WHICH HAS ALREADY POSITIONED ITS HEAD. IF ONE IS FOUND THE COMMAND IS EXECUTED ON THAT DRIVE AND THE CONTROLLER AGAIN BECOMES BUSY. IF NO POSITIONED COMMAND IS FOUND, A CHECK IS MADE TO SEE IF THERE IS A COMMAND THAT IS TO BE POSITIONED. IF YES, IT IS POSITIONED AND THE LAST AVAILABLE COMMAND IS EXECUTED. IF IT IS FOUND THAT NO DRIVE NEEDS TO BE POSITIONED (THIS COULD HAPPEN IF THERE IS ONLY ONE COMMAND LEFT IN THE QUEUE OR THE REMAINING COMMANDS IN THE QUEUE ARE TO BE PERFORMED ON THE SAME DRIVE), THEN THE COMMANDS IS/ ARE EXECUTED.

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

9.0 TRANSFER DATA LOGGING

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

10.0 ERROR LOGGING

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

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

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

11.0 ERROR REPORTING AND RECOVERY

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

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

IF A FATAL ERROR CONDITION IS DETECTED (LIKE DRIVE UNSAFE, WRITE PROTECT SET, DRIVE READY CLEAR, ETC.) THE DRIVE IS REMOVED FROM THE DRIVE SELECTION TABLE AND DROPPED FROM FURTHER TESTING. A MESSAGE IS GIVEN INDICATING DROPPING OF THAT DRIVE. FOR FURTHER INFORMATION, REFER TO THE 'CHKDRV' AND 'DSELCT' 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. THROGH 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 ERROR CALL. THUS 'ERROR 1' IS 104001, ETC. THE ERROR ROUTINE DECIDES IF ANY ACTION IS TO BE TAKEN DEPENDING ON THE SWITCH SETTING (LIKE, HALT ON ERROR, INHIBIT ERROR TYPEOUT, ETC.).

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

```
                        .TITLE  MAINDEC-11-DZRKH-F
                        ;*COPYRIGHT (C) 1973,1976
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY JIM KAPADIA
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
                        ;*
                        ;*REVISED BY GEORGE GALLANT,TOM SAWYER   -  MARCH 1976
                        ;*REVISED BY CHUCK HESS - AUGUST 1976
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        12            TYPE OUT ERROR HISTORY
                        ;*        10            BELL ON ERROR
                        ;*         9            LOOP ON ERROR
                        ;*         8            TYPE OUT ERROR AND TRANSFER DATA STATISTICS
                        ;*         6            SELECT BUS ADDRESS LIMITS FOR DISK DATA TRANSFERS
                        ;*         5            HALT BEFORE DOING NEXT SET OF COMMANDS(GEN8RQ)
                        ;*         4            DO NOT REWRITE THE DISKS ON RESTART AT 210
                        ;*         3            TYPE OUT ELAPSED TIME AT ERROR
                        ;*         2            DROP DRIVE AFTER MAXM ERORS ON THIS DRIVE
                        ;*         1            TYPE SERIAL NUMBER OF ERRORING DRIVE
                        ;*         0            IF SW8=1, ONLY TYPE ELAPSED TIME
                        ;*        11            DUMP OUT ALL RK11 REGISTERS ON ERROR


                        ;*      YOU ARE ADVISED TO READ THE DOCUMENT FOR THIS PROGRAM.
```

```


                        .SBTTL  BASIC DEFINITIONS

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

                        ;*MISCELLANEOUS DEFINITIONS
        000011          HT=     11              ;;CODE FOR HORIZONTAL TAB
        000012          LF=     12              ;;CODE FOR LINE FEED
        000015          CR=     15              ;;CODE FOR CARRIAGE RETURN
        000200          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
        177776          PS=     177776          ;;PROCESSOR STATUS WORD
                        .EQUIV  PS,PSW
        177774          STKLMT= 177774          ;;STACK LIMIT REGISTER
        177772          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
        177570          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
        177570          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
        000000          R0=     %0              ;;GENERAL REGISTER
        000001          R1=     %1              ;;GENERAL REGISTER
        000002          R2=     %2              ;;GENERAL REGISTER
        000003          R3=     %3              ;;GENERAL REGISTER
        000004          R4=     %4              ;;GENERAL REGISTER
        000005          R5=     %5              ;;GENERAL REGISTER
        000006          R6=     %6              ;;GENERAL REGISTER
        000007          R7=     %7              ;;GENERAL REGISTER
        000006          SP=     %6              ;;STACK POINTER
        000007          PC=     %7              ;;PROGRAM COUNTER

                        ;*PRIORITY LEVEL DEFINITIONS
        000000          PR0=    0               ;;PRIORITY LEVEL 0
        000040          PR1=    40              ;;PRIORITY LEVEL 1
        000100          PR2=    100             ;;PRIORITY LEVEL 2
        000140          PR3=    140             ;;PRIORITY LEVEL 3
        000200          PR4=    200             ;;PRIORITY LEVEL 4
        000240          PR5=    240             ;;PRIORITY LEVEL 5
        000300          PR6=    300             ;;PRIORITY LEVEL 6
        000340          PR7=    340             ;;PRIORITY LEVEL 7

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
        100000          SW15=   100000
        040000          SW14=   40000
        020000          SW13=   20000
        010000          SW12=   10000
        004000          SW11=   4000
        002000          SW10=   2000
        001000          SW09=   1000
        000400          SW08=   400
        000200          SW07=   200
        000100          SW06=   100
        000040          SW05=   40
        000020          SW04=   20
```

```
          000010                      SW03=   10
          000004                      SW02=   4
          000002                      SW01=   2
          000001                      SW00=   1
                                      .EQUIV  SW09,SW9
                                      .EQUIV  SW08,SW8
                                      .EQUIV  SW07,SW7
                                      .EQUIV  SW06,SW6
                                      .EQUIV  SW05,SW5
                                      .EQUIV  SW04,SW4
                                      .EQUIV  SW03,SW3
                                      .EQUIV  SW02,SW2
                                      .EQUIV  SW01,SW1
                                      .EQUIV  SW00,SW0

                                      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
          100000                      BIT15=  100000
          040000                      BIT14=  40000
          020000                      BIT13=  20000
          010000                      BIT12=  10000
          004000                      BIT11=  4000
          002000                      BIT10=  2000
          001000                      BIT09=  1000
          000400                      BIT08=  400
          000200                      BIT07=  200
          000100                      BIT06=  100
          000040                      BIT05=  40
          000020                      BIT04=  20
          000010                      BIT03=  10
          000004                      BIT02=  4
          000002                      BIT01=  2
          000001                      BIT00=  1
                                      .EQUIV  BIT09,BIT9
                                      .EQUIV  BIT08,BIT8
                                      .EQUIV  BIT07,BIT7
                                      .EQUIV  BIT06,BIT6
                                      .EQUIV  BIT05,BIT5
                                      .EQUIV  BIT04,BIT4
                                      .EQUIV  BIT03,BIT3
                                      .EQUIV  BIT02,BIT2
                                      .EQUIV  BIT01,BIT1
                                      .EQUIV  BIT00,BIT0

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

```
          000100                      KWLVEC=100                  ;KW11L CLOCK VECTOR

                                      .EQUIV  BIT15,ERR
                                      .EQUIV  BIT14,HE
                                      .EQUIV  BIT13,SCP


                                      .EQUIV  BIT12,DPL
                                      .EQUIV  BIT10,DRU
                                      .EQUIV  BIT09,SIN
                                      .EQUIV  BIT07,DRY
                                      .EQUIV  BIT06,RWS
                                      .EQUIV  BIT05,WPS



                                      .EQUIV  BIT12,SKE
                                      .EQUIV  BIT01,CSE
                                      .EQUIV  BIT00,WCE

                                      .SBTTL  TRAP CATCHER

          000000                                .=0
                                      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
          000174                                .=174
000174    000000                      DISPREG: .WORD  0                      ;;SOFTWARE DISPLAY REGISTER
000176    000000                      SWREG:   .WORD  0                      ;;SOFTWARE SWITCH REGISTER
                                      .SBTTL  STARTING ADDRESS(ES)
000200    000137    003376                      JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM

          000210                                .=210                   ;RESTART ADDRESS, IF RESTART IS
000210    105237    001253                      INCB    FRSTRT          ;DONE AT 210, THE BOUNDARY CONDITION
000214    000137    003376                      JMP     @#START         ;TESTS (TST1-7) ARE SKIPPED. IF SW 4
                                                                        ;IS SET THEN THE DISKS ARE NOT REWRITTEN
                                                                        ;(WRDSK) WITH RANDOM PATTERNS. NORAMALLY
                                                                        ;ALL THE DISKS PRESENT ARE COMPLETELY
                                                                        ;WRITTEN WITH RANDOM PATTERNS, AT THE
                                                                        ;BEGINING OF THE
                                                                        ;EXERCISER PART OF THE PROGRAM.

                                      .SBTTL  ACT11 HOOKS

                                      ;;************************************************************************
                                      ;HOOKS REQUIRED BY ACT11
          000220                                $SVPC=.                 ;SAVE PC
          000046                                .=46
000046    022656                                $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
          000052                                .=52
000052    000000                                .WORD   0               ;;2)SET LOC.52 TO ZERO
          000220                                .=$SVPC                 ;; RESTORE PC

                                      ;KT11 REGISTER DEFINITIONS
                                      .SBTTL  MEMORY MANAGEMENT DEFINITIONS

```

```
                                        ;*KT11 VECTOR ADDRESS

          000250                        MMVEC=  250

                                        ;*KT11 STATUS REGISTER ADDRESSES

          177572                        SR0=    177572
          177574                        SR1=    177574
          177576                        SR2=    177576
          172516                        SR3=    172516

                                        ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS

          172300                        KIPDR0= 172300
          172302                        KIPDR1= 172302
          172304                        KIPDR2= 172304
          172306                        KIPDR3= 172306
          172310                        KIPDR4= 172310
          172312                        KIPDR5= 172312
          172314                        KIPDR6= 172314
          172316                        KIPDR7= 172316

                                        ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS

          172320                        KDPDR0= 172320
          172322                        KDPDR1= 172322
          172324                        KDPDR2= 172324
          172326                        KDPDR3= 172326
          172330                        KDPDR4= 172330
          172332                        KDPDR5= 172332
          172334                        KDPDR6= 172334
          172336                        KDPDR7= 172336

                                        ;*KERNEL "I" PAGE ADDRESS REGISTERS

          172340                        KIPAR0= 172340
          172342                        KIPAR1= 172342
          172344                        KIPAR2= 172344
          172346                        KIPAR3= 172346
          172350                        KIPAR4= 172350
          172352                        KIPAR5= 172352
          172354                        KIPAR6= 172354
          172356                        KIPAR7= 172356

                                        ;*KERNEL "D" PAGE ADDRESS REGISIERS

          172360                        KDPAR0= 172360
          172362                        KDPAR1= 172362
          172364                        KDPAR2= 172364
          172366                        KDPAR3= 172366
          172370                        KDPAR4= 172370
          172372                        KDPAR5= 172372
          172374                        KDPAR6= 172374
          172376                        KDPAR7= 172376


```

```
                                        .SBTTL  COMMON TAGS

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

          001100                                .=1100
001100                                  $CMTAG:                         ;;START OF COMMON TAGS
001100    000000                        $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102       000                        $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103       000                        $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104    000000                        $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106    000000                        $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110    000000                        $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112    000000                        $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114       000                        $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115       001                        $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116    000000                        $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120    000000                        $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122    000000                        $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124    000000                        $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126    000000                        $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130    000000                                .WORD   0               ;;RESERVED--NOT TO BE USED
001132    000000                                .WORD   0
001134       000                        $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135       000                        $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136    000000                                .WORD   0
001140    177570                        SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142    177570                        DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144    177560                        $TKS:   177560                  ;;TTY KBD STATUS
001146    177562                        $TKB:   177562                  ;;TTY KBD BUFFER
001150    177564                        $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152    177566                        $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154       000                        $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155       002                        $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156       012                        $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157       000                        $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160    000000                        $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                                        ;;WHICH  ($REG0) WAS OBTAINED
001162    000000                        $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164    000000                        $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001166    000000                        $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001170    000000                        $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001172    000000                        $REG4:  .WORD   0               ;;CONTAINS (($REGAD)+10)
001174    000000                        $REG5:  .WORD   0               ;;CONTAINS (($REGAD)+12)
001176    000000                        $REG6:  .WORD   0               ;;CONTAINS (($REGAD)+14)
001200    000000                        $REG7:  .WORD   0               ;;CONTAINS (($REGAD)+16)
001202    000000                        $REG10: .WORD   0               ;;CONTAINS (($REGAD)+20)
001204    000000                        $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001206    177607  000377                $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001212       077                        $QUES:  .ASCII  /?/             ;;QUESTION MARK
001213       015                        $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001214    000012                        $LF:    .ASCIZ  <12>            ;;LINE FEED
                                        ;;*************************************************************
001216    177400                        RKDS:   .WORD   177400
001220    177402                        RKER:   .WORD   177402
```

```
001222  177404                  RKCS:   .WORD   177404
001224  177406                  RKWC:   .WORD   177406
001226  177410                  RKBA:   .WORD   177410
001230  177412                  RKDA:   .WORD   177412
001232  177416                  RKDB:   .WORD   177416
001234  177546                  KWLS:   .WORD   177546  ;STATUS REGISTER FOR KW11L

001236  000372                  PCNTR:  .WORD   250.

001240  000220                  RKVEC:  .WORD   220     ;NORMAL RK11 INTERRUPT VECTOR ADDRESS
001242  000222                  RKSTAT: .WORD   222     ;PSW TO BE USED ON INTERRUPT

001244  000240                  PPRLVL: .WORD   240     ;PROGRAM PRIORITY LEVEL=5, PRIORITY LEVEL
                                                        ;AT WHICH THE PROGRAM OPERATES CAN BE CHANGED
                                                        ;BY ALTERING THIS LOCATION.
001246  000340                  KWPLVL: .WORD   340     ;PRIORITY LEVEL OF THE KW11L CLOCK SERVICE
                                                        ;ROUTINE.

001250  177777                  SRDRV:  .WORD   177777  ;'SRDRV' CONTAINS THE DRIVE NO WHOOSE SERIAL
                                                        ;NO IS TO BE TYPED OUT WHEN AN ERROR OCCURS,
                                                        ;IF SW 1 IS SET. WHEN (SRDRV)=-1 SERIAL NO
                                                        ;IS NOT TYPPED OUT, BECAUSE THE ERROR WAS NOT
                                                        ;POSITIVELY ATTRIBUTABLE TO A SPECIFIC DRIVE.


001252     000                  FTITLE: .BYTE   0
001253     000                  FRSTRT: .BYTE   0       ;FLAG FOR RESTART AT 210
```

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


001254  000010                  PDR:    .BLKB   10

001264  000000                  DRVPRS: .WORD   0       ;CONTAINS TOTAL NUMBER OF DRIVES PRESENT

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

001266  000010                  SRNO:   .BLKW   10      ;SERIAL NO'S FOR DRIVES 0-7


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

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


001306  000010                  KEY:    .BLKW   10      ;KEY FOR THE  COMMANDS IN QUEUE


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


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

```
001326  000040                 CMND:   .BLKW   40      ;STORAGE TABLE

                               ;THESE ARE BUSY FLAGS FOR THE DRIVES. IF A DRIVE IS BUSY PERFORMING
                               ;ANY FUNCTION (INCLUDING POSITIONING) THEN BIT 7 OF THE FLAG FOR THAT
                               ;DRIVE IS SET. BITS 0-3 CONTAIN THE OFFSET TO KEY # WHICH MADE THE DRIVE
                               ;BUSY. EX: DRIVE #3 WAS MADE TO DO A WRITE BY COMMAND
                               ;KEY5, HENCE 'BUSY3' WILL CONTAIN 210. NOTE THAT 10 IS THE
                               ;OFFSET FOR KEY5 (TAKING KEY AS BASE).  KEY #= OFFSET<0-3>/2 + 1

001426  000010                 BUSY:   .BLKB   10      ;BUSY FLAGS FOR DRIVES 0-7


                               ;THESE FLAGS WHEN SET INDICATE THAT A DRIVE IS BEING
                               ;POSITIONED OR HAS ALREADY BEEN POSITIONED.

001436  000010                 POS:    .BLKB   10      ;DRIVE 0 POSITIONED



                               ;RETRY COUNTS FOR A PARTICULAR FUNCTION ON A DRIVE THE FUNCTION IS ABORTED
                               ;ON A DRIVE WHEN THE RETRY COUNT REACHES 3.

001446  000010                 RETRY:  .BLKB   10      ;DRIVES 0-7 RERTY COUNTS

001456  000000                 WCFLG:  .WORD   0       ;IF BIT 15 IS SET WRITE CHK IS TO BE DONE
                                                       ;FOLLOWING THE WRITE. BITS 0-3 CONTAIN THE
                                                       ;OFFSET TO KEY# (FROM BASE=KEY)

001460  000000                 QSCNT:  .WORD   0       ;THIS IS A COUNT FOR KEEPING TRACK OF THE TIME
                                                       ;TAKEN BY ALL THE 8 COMMANDS IN THE QUEUE.
                                                       ;IF THIS COUNTS DOWN TO 0 AN ERROR IS REPORTED


001462  000000                 PRSFNC: .WORD   0       ;COTAINS INFO ABOUT THE PRESENT COMMAND
                                                       ;BEING PERFORMED ON THE RK11
001464  000000                 PSTFNC: .WORD   0       ;CONTAINS INFO ABOUT THE COMMAND PERFORMED
                                                       ;BEFORE THE 'PRSCMND'


001466  000000                 CICNT:  .WORD   0       ;THIS IS A COUNT-TIMER USED FOR KEEPING TRACK
001470  000000                 CICNT1: .WORD   0       ;OF THE TIME TAKEN BY ANY FUNCTION TO BE
                                                       ;COMPLETED. IF THE COUNT GOES TO 0 AN ERROR IS REPORTED.

001472  000000                 TIMER:  .WORD   0
001474  000000                 ERCODE: .WORD   0
001476  000000                 DRVPTR: .WORD   0
001500  000000                 DRVCNT: .WORD   0


001502  000000                 QDRV:   .WORD   0       ;TEMPORARY REGISTERS USED BY 'GEN8RQ'
001504  000000                 QCYL:   .WORD   0       ;ROUTINE TO STORE VARIOUS PARAMETERS
001506  000000                 QSUR:   .WORD   0       ;OF A COMMAND AS THEY ARE GENERATED.
001510  000000                 QSEC:   .WORD   0
001512  000000                 QFNC:   .WORD   0
001514  000000                 QBUSAD: .WORD   0
001516  000000                 QWRCNT: .WORD   0
```

```


                               ;THIS TABLE CONTAINS VARIOUS MAPPING FACTORS TO BE USED
                               ;FOR GENERATING RANDOM PARAMETERS FROM RANDOM NUMBERS

001520  000000                 DRMAP:  .WORD   0       ;MAPPING FACTOR FOR GENERATING RANDOM DRIVE NUMBER
001522  000000                 CYLMAP: .WORD   0       ;MAPPING FACTOR FOR CYLINDER
001524  000000                 SECMAP: .WORD   0       ;MAPPING FACTOR FOR SECTOR
001526  000000                 FNMAP:  .WORD   0       ;MAPPING FACTOR FOR FUNCTION
001530  000000                 BAMAP:  .WORD   0       ;MAPPING FACTOR FOR BUS ADDRESS
001532  000000                 WCMAP:  .WORD   0       ;MAPPING FACTOR FOR WORD COUNT


                               ;THESE TWO FLAGS CORRESPOND TO THE 2 INTERRUPT HANDLERS (RK11) USED
                               ;IN THIS PROGRAM. WHEN THE INTERRUPT HANDLER IS ENTERED THE FLAG IS
                               ;CLEARED OR SET.

001534     000                 INTFLG: .BYTE   0       ;FOR 'INTHND ', CLEARED ON ENTERING HANDLER
001535     000                 INT1FL: .BYTE   0       ;FOR 'INT1SK', SET ON ENTERING HANDLER

001536  000000                 SAVKEY: .WORD   0
001540  000000                 ECOUNT: .WORD   0

                               ;THIS TABLE CONTAINS COUNTS FOR THE NUMBER OF OF ERRORS OCCURING ON A
                               ;DRIVE (NOTE: ONLY THOSE ERRORS WHICH ARE POSITIVELY ATTRIBUTABLE TO A
                               ;SPECIFIC DRIVE). THE COUNT  KEPT ONLY IF SWITCH 2 IS SET. WHEN THE COUNT
                               ;REACHES THE MAXIMUM ALLOWABLE (USUALLY 3) THE DRIVE IS DROPPED FROM
                               ;TESTING AND IS TAKEN OUT OF THE DRIVE SELECTION TABLE.

001542  000010                 ERDRV:  .BLKB   10      ;COUNT FOR DRIVES 0-7

001552  000000                 KWHR:   .WORD   0       ;COUNTS HOURS (2'S COMPLEMENT)
001554  000000                 KWMIN:  .WORD   0       ;COUNTS MINUTES (2'S COMPLEMENT)
001556  000000                 KWSEC:  .WORD   0       ;COUNTS SECONDS (2'S COMPLEMENT)
001560  000000                 KWCOUNT: .WORD  0       ;COUNTS CPS FROM KW11L (2'S COMPLMNT)

                               ;THIS TABLE CONTAINS COUNTS FOR HARD ERRORS ON A PARTICULAR DRIVE.
                               ;EX HECN2 WILL CONTAIN THE TOTAL NUMBER OF HARD ERRORS THAT OCCURED ON
                               ;DRIVE 2

001562  000010                 HECN:   .BLKW   10      ;DRIVE 0-7 HARD ERROR COUNTS

                               ;THIS TABLE CONTAINS COUNTS FOR SEEK ERRORS
                               ;ON A PARTICULAR DRIVE.

001602  000010                 SKECN:  .BLKW   10      ;DRIVE 0-7 SEEK ERROR COUNTS


                               ;THIS TABLE CONTAINS COUNTS FOR SIN ERRORS ON A
                               ;PARTICULAR DRIVE

001622  000010                 SINCN:  .BLKB   10      ;DRIVE 0-7 SIN COUNTS

                               ;THIS TABLE CONTAINS COUNTS FOR WRITE CHECK ERRORS
                               ;THAT OCCURED ON A PARTICULAR DRIVE

```

```
001632  000010                      WCECN:  .BLKW   10      ;WCE COUNT FOR DRIVES 0-7


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

001652  000010                      CSECN:  .BLKW   10      ;CSE COUNT FOR DRIVES 0-7


                                    ;THIS TABLE CONTAINS COUNT OF NUMBER OF FUNCTIONS
                                    ;THAT WERE ABORTED ON A PARTICULAR DRIVE. A
                                    ;FUNCTION IS ABORTED ONLY AFTER DOING RETRIES

001672  000010                      ABORT:  .BLKW   10      ;ABORT COUNT FOR DRIVES 0-7

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

001712  000010                      DATER:  .BLKW   10      ;DRIVES 0-7
```

```
001732  000000                      NWRTL:  .WORD   0       ;LO WORD: OF THE 2 WORD COUNT-GIVING TOTAL
001734  000000                      NWRTH:  .WORD   0       ;HI WORD:# OF WORDS WRITTEN ON DRIVE 0
001736  000016                              .BLKW   14.     ;FOR REST OF DRIVES 1-7


001772  000000                      NRDL:   .WORD   0       ;LO WORD: 2 WORD COUNT GIVING TOTAL
001774  000000                      NRDH:   .WORD   0       ;HI WORD: # OF WORDS READ ON DRIVE 0
001776  000016                              .BLKW   14.     ;FOR DRIVES 1-7

002032  001326                      PCMND:  .WORD   CMND    ;POINTERS TO PARAMETERS FOR COMMANDS IN QUEUE
002034  001336                              .WORD   CMND+10 ;POINTER TO SECOND COMMAND
002036  001346                              .WORD   CMND+20 ;POINTER TO THIRD COMMAND
002040  001356                              .WORD   CMND+30 ;POINTER TO FOURTH COMMAND
002042  001366                              .WORD   CMND+40 ;POINTER TO FIFTH COMMAND
002044  001376                              .WORD   CMND+50 ;POINTER TO SIXTH COMMAND
002046  001406                              .WORD   CMND+60 ;POINTER TO SEVENTH COMMAND
002050  001416                              .WORD   CMND+70 ;POINTER TO EIGHTH COMMAND


002052  000000                      BASEBA: .WORD   0       ;CONTAINS THE LOWEST BUS ADDRESS STARTING WHICH DATA TRANSFERS
                                                            ;CAN BE DONE
002054  000000                      MAXBA:  .WORD   0       ;CONTAINS THE HIGHEST BUS ADDRESS TO WHICH DATA TRANSFERS
                                                            ;CAN BE DONE.
002056  000000                      REPCNT:.WORD    0       ;CONTAINS THE REPETITION COUNT- THE NUMBER
                                                            ;OF TIMES Q REQUESTS WILL BE GENERATED, WHEN THIS
                                                            ;COUNT GOES TO 0, IT MEANS AN END OF PASS. HOWEVER
                                                            ;NOTE THAT THERE IS NO TRUE END OF PASS, IN THIS KIND
                                                            ;OF EXERCISER PROGRAM. THE EXERCISER RESUMES FROM
                                                            ;THE POINT IT LEFT OFF, AFTER TYPING OUT THE END IF
                                                            ;PASS MESSAGE.
002060  000000                      XXDPMD: .WORD   0       ;LOW BYTE CONTAINS ADDRESS OF RK05 DRIVE
                                                            ;WHICH PROGRAM WAS LOADED FROM; HIGH BYTE
                                                            ;CONTAINS THE RK05 'XXDP' CODE.

                                    ;ASCII MESSAGES

002062  005015  045523  000105      MSG1:   .ASCIZ  <15><12>/SKE/
002070  005015  041527  000105      MSG2:   .ASCIZ<15><12>/WCE/
002076  005015  051503  000105      MSG3:   .ASCIZ<15><12>  /CSE/
002104  005015  040510  042122      MSG4:   .ASCIZ  <15><12>/HARD EROR/
002112  042440  047522  000122
002120  047440  020116  047504      MSG5:   .ASCIZ/ ON DOING /
002126  047111  020107     000
002133     127  044522  042524      MSG6:   .ASCIZ  /WRITE/
002140     000
002141     122  040505  000104      MSG7:   .ASCIZ  /READ/
002146  051127  020124  044103      MSG8:   .ASCIZ  /WRT CHK/
002154  000113
002156  042122  041440  045510      MSG9:   .ASCIZ  /RD CHK/
002164     000
002165     015  040412  047502      MSG10:  .ASCIZ  <15><12>/ABORTED/<15><12>
002172  052122  042105  005015
002200     000
002201     123  042505  000113      MSG11:  .ASCIZ  /SEEK/
002206  005015  041520  000075      MSG12:  .ASCIZ  <15><12>/PC=/
002214  044120  051531  041040      MSG13:  .ASCIZ  /PHYS BA=/
```

```
002222  036501     000
002225     015  047012  020117  MSG14:  .ASCIZ  <15><12>/NO DRVS PRSNT/
002232  051104  051526  050040
002240  051522  052116     000
002245     015  042012  053122  MSG15:  .ASCIZ  <15><12>/DRVE # DIDN'T INTERUPT AFTER /
002252  020105  020043  044504
002260  047104  052047  044440
002266  052116  051105  050125
002274  020124  043101  042524
002302  020122     000
002305     015  045412  054505  MSG16:  .ASCIZ  <15><12>/KEY-8   BUSY-7/
002312  034055  020040  041040
002320  051525  026531  000067
002326  051440  020122  047516  MSG17:  .ASCII  / SR NO/
002334  000072                  MSG18:  .ASCIZ  /:/
002336  005015  042040  047522  MSG19:  .ASCIZ  <15><12>/ DROPPED DRIVE # /
002344  050120  042105  042040
002352  044522  042526  021440
002360  000040
002362  005015  051104  053111  MSG20:  .ASCIZ  <15><12>/DRIVE/
002370  000105
002372  020054     000          MSG24:  .ASCIZ  /, /
002375     106     000          MSG25:  .ASCIZ  /F/
002377     015  042012  044522  MSG26:  .ASCIZ  <15><12>/DRIVE  WRDS WRITN  WRDS READ    CSE    WCE DATERR     HE  /
002404  042526  020040  051127
002412  051504  053440  044522
002420  047124  020040  051127
002426  051504  051040  040505
002434  020104  020040  041440
002442  042523  020040  020040
002450  041527  020105  040504
002456  042524  051122  020040
002464  020040  044040  020105
002472  000040
002474  005015  051104  053111  MSG26A: .ASCIZ  <15><12>/DRIVE     SKE  ABORT    SIN/
002502  020105  020040  020040
002510  045523  020105  040440
002516  047502  052122  020040
002524  020040  044523  000116
002532  005015  047125  041101  MSG27:  .ASCIZ  <15><12>/UNABLE TO CLEAR ERROR AFTER THREE TRIES/
002540  042514  052040  020117
002546  046103  040505  020122
002554  051105  047522  020122
002562  043101  042524  020122
002570  044124  042522  020105
002576  051124  042511  000123
002604  005015  051105  047522  MSG28:  .ASCIZ  <15><12>/ERROR CONDITION CLEARED ON RETRY # /
002612  020122  047503  042116
002620  052111  047511  020116
002626  046103  040505  042522
002634  020104  047117  051040
002642  052105  054522  021440
002650  000040
002652  005015  044524  042515  MSG29:  .ASCIZ  <15><12>/TIME /
002660  000040
```

```
002662     040                  BLNKS3: .ASCII  / /
002663     040                  BLNKS2: .ASCII  / /
002664  000040                  BLNKS1: .ASCIZ  / /
                                        .EVEN
```

```
                                .SBTTL  ERROR POINTER TABLE

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

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


002666                          $ERRTB:
                                ;*THERE ARE TWO CLASSES OF ERRORS:
                                ;*1. ERRORS IN EXERCISER PART OF THE PROGRAM - ERROR NUMBERS BELOW 100
                                ;*2. ERRORS IN THE NON-EXERCISER PART OF THE PROGRAM - ERROR NUMBERS EQUAL
                                ;*TO AND GREATER THAN 100.
                                ;*THE DOCUMENT CONTAINS MORE INFORMATION ON THESE.
                                ;*THE FOLLOWING ERRORS OCCUR IN THE EXERCISER PART OF THE PROGRAM.


                                ;ITEM   1

002666  027530                          EM1     ;ERROR ON WRITE
002670  031626                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
002672  032320                          DT1     ;;$ERRPC $REG0  $REG1   $REG2   $REG3
002674  000000                          0

                                ;ITEM   2

002676  027546                          EM2     ;ATTEMPT TO INITIATE FUNCTION ON 'BUSY' DRIVE
002700  031674                          DH2     ;PC     DRIVE
002702  032334                          DT2     ;;$ERRPC $REG0
002704  000000                          0

                                ;ITEM   3

002706  027621                          EM3     ;CONTROL READY NOT SET
002710  031626                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
002712  032320                          DT1     ;;$ERRPC $REG0  $REG1   $REG2   $REG3
002714  000000                          0

                                ;ITEM   4

002716  027644                          EM4     ;R/W/S READY NOT SET
002720  031626                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
002722  032320                          DT1     ;;$ERRPC $REG0  $REG1   $REG2   $REG3
002724  000000                          0

                                ;ITEM   5

002726  027667                          EM5     ;CONTROL READY NOT SET AFTER FIRST INTERRUPT ON ISSUING SEEK
002730  031626                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
002732  032320                          DT1     ;;$ERRPC $REG0  $REG1   $REG2   $REG3
```

```
002734  000000                          0


                                ;ITEM   6

002736  027754                          EM6     ;WRONG BITS IN RKCS, EXPECT SEEK
002740  031626                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
002742  032320                          DT1     ;;$ERRPC $REG0  $REG1   $REG2   $REG3
002744  000000                          0

                                ;ITEM   7

002746  030013                          EM7     ;'BUSY' FLAG CLEAR ON INTERRUPTING DRIVE
002750  031674                          DH2     ;PC     DRIVE
002752  032334                          DT2     ;;$ERRPC $REG0
002754  000000                          0

                                ;ITEM   10

002756  030060                          EM10    ;'POSITIONING' FLAG FOR INTERRUPTING DRIVE CLEAR
002760  031674                          DH2     ;PC     DRIVE
002762  032334                          DT2     ;;$ERRPC $REG0
002764  000000                          0

                                ;ITEM   11

002766  030135                          EM11    ;'ERR'OR SET AFTER FIRST INTERRUPT ON ISSUING SEEK
002770  031626                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
002772  032320                          DT1     ;;$ERRPC $REG0  $REG1   $REG2   $REG3
002774  000000                          0

                                ;ITEM   12

002776  030215                          EM12    ;SCP SET AFTER FIRST INTERRUPT ON ISSUING SEEK
003000  031626                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
003002  032320                          DT1     ;;$ERRPC $REG0  $REG1   $REG2   $REG3
003004  000000                          0

                                ;ITEM   13

003006  030267                          EM13    ;CONTROL READY NOT SET AFTER SEEK DONE INTERRUPT
003010  031626                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
003012  032320                          DT1     ;;$ERRPC $REG0  $REG1   $REG2   $REG3
003014  000000                          0

                                ;ITEM   14

003016  030342                          EM14    ;INTERRUPTING DRIVE (SEEK DONE) WAS NOT 'BUSY'
003020  031626                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
003022  032320                          DT1     ;;$ERRPC $REG0  $REG1   $REG2   $REG3
003024  000000                          0

                                ;ITEM   15

003026  030415                          EM15    ;R/W/S READY NOT SET FOR INTERRUPTING DRIVE (SEEK DONE)
003030  031626                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
```

```
003032  032320                            DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
003034  000000                            0

                                  ;ITEM   16

003036  030501                            EM16    ;'SIN' ERROR
003040  031626                            DH1     ;PC     RKCS    RKER    RKDS    RKDA
003042  032320                            DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
003044  000000                            0

                                  ;ITEM   17

003046  030512                            EM17    ;'ERR'OR ON DOING SEEK
003050  031626                            DH1     ;PC     RKCS    RKER    RKDS    RKDA
003052  032320                            DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
003054  000000                            0

                                  ;ITEM   20

003056  030540                            EM20    ;SCP DID NOT SET AFTER SEEK WAS DONE
003060  031626                            DH1     ;PC     RKCS    RKER    RKDS    RKDA
003062  032320                            DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
003064  000000                            0

                                  ;ITEM   21

003066  030604                            EM21    ;SOFT ERROR
003070  031711                            DH21    ;$ERRPC RKCS    RKER    RKDS    RKDA:   DRV#
                                                  ;CYL    SUR     SEC
003072  032342                            DT21    ;$ERRPC $REG0   $REG1   $REG2           $REG3
                                                  ;$REG4  $REG5   $REG6
003074  000000                            0

                                  ;ITEM   22

003076  000000                            0
003100  031711                            DH21    ;$ERRPC RKCS    RKER    RKDS    RKDA:   DRV#
                                                  ;CYL    SUR     SEC
003102  032342                            DT21    ;$ERRPC $REG0   $REG1   $REG2           $REG3
                                                  ;$REG4  $REG5   $REG6
003104  000000                            0

                                  ;ITEM   23

003106  030616                            EM23    ;DATA (COMPARISON) ERROR
003110  032006                            DH23    ;PC     RKBA    EXPCT   RECVD   RKDA
003112  032320                            DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
003114  000000                            0

                                  ;ITEM   24

003116  030645                            EM24    ;CONTROL READY CLEAR ON  INTERRUPT AFTER RK FUNCTION
003120  031626                            DH1     ;PC     RKCS    RKER    RKDS    RKDA
003122  032320                            DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
003124  000000                            0

```

```
                                  ;ITEM   25

003126  000000                            0
003130  032053                            DH25    ;PC     RKCS    RKER    RKDS    RKDA    DRIVE #
003132  032364                            DT25    ;$ERRPC $REG0   $REG1   $REG2   $REG3   $REG4
003134  000000                            0

                                  ;ITEM   26

003136  030721                            EM26    ;STUCK IN LOOP, 8 Q-COMMANDS SHOULD BE DONE BY NOW
003140  031626                            DH1     ;PC     RKCS    RKER    RKDS    RKDA
003142  032320                            DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
003144  000000                            0

                                  ;ITEM   27

003146  030774                            EM27    ;ATTEMPT TO DO WRITE CHECK BEFORE WRITE
003150  032130                            DH27    ;PC     KEY     FUNCTION CODE
003152  032402                            DT103   ;PC     $REG0   $REG1
003154  000000                            0

                                  ;ITEM   30

003156  031035                            EM30    ;ATTEMPT TO REEXECUTE A COMMAND IN PROGRESS OR ALREADY FINISHED
003160  032211                            DH30    ;PC     KEY
003162  032334                            DT2     ;$ERRPC $REG0
003164  000000                            0

                                  ;ITEM   31

003166  031130                            EM31    ;'FUNCTION IN PROGRESS' FLAG FOR INTERRUPTING DRIVE IS NOT SET
003170  031674                            DH2     ;PC     DRIVE
003172  032334                            DT2     ;$ERRPC $REG0
003174  000000                            0

                                  ;ITEM   32

003176  031221                            EM32    ;UNEXPECTED DRIVE INTERRUPTED
003200  032163                            DH103   ;PC     EXPCT   RECVD
003202  032402                            DT103   ;$ERRPC $REG0   $REG1
003204  000000                            0

                                  ;ITEM   33

003206  031253                            EM33    ;WRONG FUNCTION CODE IN RKCS AFTER INTERRUPT
003210  032163                            DH103   ;PC     EXPCT   RECVD
003212  032402                            DT103   ;$ERRPC $REG0   $REG1
003214  000000                            0

                                  ;ITEM   34

003216  031330                            EM34    ;DRIVE READY CLEAR
003220  031626                            DH1     ;PC     RKCS    RKER    RKDS    RKDA
003222  032320                            DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
003224  000000                            0

```

```
                                        ;ITEM   35

        003226  031347                          EM35    ;DRIVE POWER LOW
        003230  031626                          DH1     ;PC      RKCS     RKER     RKDS     RKDA
        003232  032320                          DT1     ;$ERRPC $REG0    $REG1    $REG2    $REG3
        003234  000000                          0

                                        ;ITEM   36

        003236  031365                          EM36    ;DRIVE UNSAFE
        003240  031626                          DH1     ;PC      RKCS     RKER     RKDS     RKDA
        003242  032320                          DT1     ;$ERRPC $REG0    $REG1    $REG2    $REG3
        003244  000000                          0

                                        ;ITEM   37

        003246  031401                          EM37    ;WPS SET
        003250  031626                          DH1     ;PC      RKCS     RKER     RKDS     RKDA
        003252  032320                          DT1     ;$ERRPC $REG0    $REG1    $REG2    $REG3
        003254  000000                          0

                                        ;*
                                        ;*THE FOLLOWING ERRORS OCCUR IN THE NON-EXERCISER PART OF THE PROGRAM.
                                        ;*

                                        ;ITEM   100

        003256  030616                          EM23    ;DATA (COMPARISON) ERROR
        003260  032006                          DH23    ;PC      RKBA     EXPCT    RECVD    RKDA
        003262  032320                          DT1     ;$ERRPC $REG0    $REG1    $REG2    $REG3
        003264  000000                          0

                                        ;ITEM   101

        003266  031411                          EM101   ;INTERRUPT DID NOT OCCUR AFTER WRITE
        003270  031626                          DH1     ;PC      RKCS     RKER     RKDS     RKDA
        003272  032320                          DT1     ;$ERRPC $REG0    $REG1    $REG2    $REG3
        003274  000000                          0

                                        ;ITEM   102

        003276  031451                          EM102   ;'ERR'OR SET
        003300  031626                          DH1     ;PC      RKCS     RKER     RKDS     RKDA
        003302  032320                          DT1     ;$ERRPC $REG0    $REG1    $REG2    $REG3
        003304  000000                          0

                                        ;ITEM   103

        003306  031465                          EM103   ;RKDA   INCREMENTED WRONGLY
        003310  032163                          DH103   ;PC      EXPCT    RECVD
        003312  032402                          DT103   ;$ERRPC $REG0    $REG1
        003314  000000                          0

                                        ;ITEM   104

        003316  031513                          EM104   ;RKBA INCREMENTED WRONGLY
```

```
        003320  032163                          DH103   ;PC      EXPCT    RECVD
        003322  032402                          DT103   ;$ERRPC $REG0    $REG1
        003324  000000                          0

                                        ;ITEM   105

        003326  031541                          EM105   ;RKWC DID NOT OVERFLOW TO 0
        003330  032225                          DH105   ;PC      RKDA     RKWC
        003332  032402                          DT103   ;$ERRPC $REG0    $REG1
        003334  000000                          0

                                        ;ITEM   106

        003336  031571                          EM106   ;MEX BITS INCORRECT
        003340  031626                          DH1     ;PC      RKCS     RKER     RKDS     RKDA
        003342  032320                          DT1     ;$ERRPC $REG0    $REG1    $REG2    $REG3
        003344  000000                          0

                                        ;ITEM   107

        003346  030616                          EM23    ;DATA (COMPARISON) ERROR ON READ
        003350  032163                          DH103   ;PC      EXPCT    RECVD
        003352  032402                          DT103   ;$ERRPC $REG0    $REG1
        003354  000000                          0

                                        ;ITEM   110

        003356  031610                          EM110   ;WRITE CHECK ERROR
        003360  032252                          DH110   ;PC      RKCS     RKER     RKBA     RKDA
        003362  032320                          DT1     ;$ERRPC $REG0    $REG1    $REG2    $REG3
        003364  000000                          0
```

```
                                 ;IF POWER FAILED, ON RETURN OF POWER ENTER HERE.

003366  004737  022536           PFSTRT: JSR     PC,WATIME         ;WAIT SOME TIME
003372  105237  001253                   INCB    FRSTRT            ;INDICATE THAT THE STATISTICS HAVE
                                                                   ;TO BE SAVED, ON RETRN FROM PWR FAIL.




003376  000005                   START:  RESET                     ;CLEAR THE BUS
                                 .SBTTL  INITIALIZE THE COMMON TAGS
                                 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
003400  012706  001100                   MOV     #$CMTAG,R6        ;;FIRST LOCATION TO BE CLEARED
003404  005026                           CLR     (R6)+             ;;CLEAR MEMORY LOCATION
003406  022706  001140                   CMP     #SWR,R6 ;;DONE?
003412  001374                           BNE     .-6               ;;LOOP BACK IF NO
003414  012706  001100                   MOV     #STACK,SP         ;;SETUP THE STACK POINTER
                                 ;;INITIALIZE A FEW VECTORS
003420  012737  027102  000020           MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
003426  012737  000340  000022           MOV     #340,@#IOTVEC+2 ;;LEVEL 7
003434  012737  027246  000034           MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
003442  012737  000340  000036           MOV     #340,@#TRAPVEC+2;LEVEL 7
003450  012737  027346  000024           MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
003456  012737  000340  000026           MOV     #340,@#PWRVEC+2 ;;LEVEL 7
003464  012737  003464  001106           MOV     #.,$LPADR         ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
003472  012737  003472  001110           MOV     #.,$LPERR         ;;SETUP THE ERROR LOOP ADDRESS
                                 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
003500  013746  000004                   MOV     @#ERRVEC,-(SP)    ;;SAVE ERROR VECTOR
003504  012737  003540  000004           MOV     #64$,@#ERRVEC     ;;SET UP ERROR VECTOR
003512  012737  177570  001140           MOV     #DSWR,SWR         ;;SETUP FOR A HARDWARE SWICH REGISTER
003520  012737  177570  001142           MOV     #DDISP,DISPLAY    ;;AND A HARDWARE DISPLAY REGISTER
003526  022777  177777  175404           CMP     #-1,@SWR          ;;TRY TO REFERENCE HARDWARE SWR
003534  001012                           BNE     66$               ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                   ;;AND  THE HARDWARE SWR IS NOT = -1
003536  000403                           BR      65$               ;;BRANCH IF NO TIMEOUT
003540  012716  003546           64$:    MOV     #65$,(SP)         ;;SET UP FOR TRAP RETURN
003544  000002                           RTI
003546  012737  000176  001140   65$:    MOV     #SWREG,SWR        ;;POINT TO SOFTWARE SWR
003554  012737  000174  001142           MOV     #DISPREG,DISPLAY
003562  012637  000004           66$:    MOV     (SP)+,@#ERRVEC    ;;RESTORE ERROR VECTOR

                                 .SBTTL  TYPE PROGRAM NAME
                                 ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
003566  005227  177777                   INC     #-1               ;;FIRST TIME?
003572  001052                           BNE     67$               ;;BRANCH IF NO
003574  104401  003632                   TYPE    ,68$              ;;TYPE ASCIZ STRING
                                 .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
003600  005737  000042                   TST     @#42              ;;ARE WE RUNNING UNDER XXDP/ACT?
003604  001006                           BNE     69$               ;;BRANCH IF YES
003606  023727  001140  000176           CMP     SWR,#SWREG        ;;SOFTWARE SWITCH REG SELECTED?
003614  001005                           BNE     70$               ;;BRANCH IF NO
003616  104406                           GTSWR                     ;;GET SOFT-SWR SETTINGS
003620  000403                           BR      70$
003622  112737  000001  001134   69$:    MOVB    #1,$AUTOB         ;;SET AUTO-MODE INDICATOR
003630                           70$:
```

```
003630  000433                           BR      67$               ;;GET OVER THE ASCIZ
                                 ;;68$:  .ASCIZ  <CRLF>*RK11/RK05 PERFORMANCE EXERCISER*<15><12>*MAINDEC-11-DZRKH-F*<CRLF
003720                           67$:
003720  012737  026114  000030           MOV     #$ERROR,@#EMTVEC          ;EMT VECTOR FOR ERROR ROUTINE
003726  013737  001244  000032           MOV     PPRLVL,@#EMTVEC+2         ;LEVEL 5
003734  012737  022460  000100           MOV     #KWSRVE,@#KWLVEC          ;KW11L CLOCK SERVICE
003742  013737  001246  000102           MOV     KWPLVL,@#KWLVEC+2         ;LEVEL 7

                                 ;THE FOLLOWING CODE FINDS OUT THE PROGRAM CONTROL MODE:
                                 ;PAPER TAPE (MANUAL), ACT11, RKDP CHAIN OR DUMP


003750  005037  002060                   CLR     XXDPMD            ;CLEAR 'XXDP' LOAD DEVICE STORAGE
003754  122737  000002  000041           CMPB    #2,41             ;LOADED FROM AN RK05 ?
003762  001160                           BNE     ST2               ;BR IF NOT
003764  013737  000040  002060           MOV     40,XXDPMD         ;GET DEVICE INDICATOR AND DRIVE ADDRESS OF
                                                                   ;LOADING RK05
003772  122737  000010  002060           CMPB    #10,XXDPMD        ;VALID DRIVE ADDRESS ?
004000  101002                           BHI     2$                ;BR IF YES
004002  105037  002060                   CLRB    XXDPMD            ;CHANGE TO DRIVE ZERO
004006  005737  000042           2$:     TST     42                ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
004012  001424                           BEQ     3$                ;BR IF NEITHER
004014  104401  004022                   TYPE    ,72$              ;;TYPE ASCIZ STRING
004020  000413                           BR      71$               ;;GET OVER THE ASCIZ
                                 ;;72$:  .ASCIZ  <15><12>/NOT TESTING DRIVE /
004050                           71$:
004050  005046                           CLR     -(SP)             ;CLEAR WORD ON STACK
004052  113716  002060                   MOVB    XXDPMD,(SP)       ;GET DRIVE ADDRESS
004056  104403                           TYPOS                     ;TYPE THE ADDRESS
004060     001                           .BYTE   1                 ;ONLY 1 CHARACTER
004061     000                           .BYTE   0                 ;SUPRESS LEADING ZEROS
004062  000520                           BR      ST2               ;GET NUMBER OF DRIVES
004064  005227  177777           3$:     INC     #-1               ;FIRST TIME THROUGH HERE ?
004070  001115                           BNE     ST2               ;BR IF NOT
004072  104401  004100                   TYPE    ,74$              ;;TYPE ASCIZ STRING
004076  000411                           BR      73$               ;;GET OVER THE ASCIZ
                                 ;;74$:  .ASCIZ  <15><12>/TO TEST DRIVE /
004122                           73$:
004122  005046                           CLR     -(SP)             ;CLEAR WORD ON THE STACK
004124  113716  002060                   MOVB    XXDPMD,(SP)       ;GET DRIVE ADDRESS
004130  104403                           TYPOS                     ;TYPE THE DRIVE ADDRESS
004132     001                           .BYTE   1                 ;ONLY 1 CHARACTER
004133     000                           .BYTE   0                 ;SUPRESS LEADING ZEROS
004134  104401  004142                   TYPE    ,76$              ;;TYPE ASCIZ STRING
004140  000431                           BR      75$               ;;GET OVER THE ASCIZ
                                 ;;76$:  .ASCIZ  / HALT PROGRAM, REMOVE RKDP PACK AND REPLACE IT/<15><12>
004224                           75$:
004224  104401  004232                   TYPE    ,78$              ;;TYPE ASCIZ STRING
004230  000435                           BR      77$               ;;GET OVER THE ASCIZ
                                 ;;78$:  .ASCIZ  /WITH A WORK PACK, CLEAR LOCATION 40, AND RESTART PROGRAM/
004324                           77$:



004324  104416                   ST2:    CON.RESET
004326  005037  001264                   CLR     DRVPRS            ;FIND WHICH DRIVE #'S ARE PRESENT
```

```
004332  005000                          CLR     R0
004334  005002                          CLR     R2
004336  005037  001254                  CLR     PDR             ;CLEAR DRIVES PRESENT TABLE
004342  005037  001256                  CLR     PDR+2
004346  005037  001260                  CLR     PDR+4
004352  005037  001262                  CLR     PDR+6
004356  012701  001254                  MOV     #PDR,R1
004362  012703  001306                  MOV     #KEY,R3
004366  010277  174636          1$:     MOV     R2,@RKDA        ;SELECT A DRIVE
004372  032777  000200  174616          BIT     #200,@RKDS      ;IS IT IN SYSTEM?
004400  001415                          BEQ     3$              ;NO
004402  010237  001502                  MOV     R2,QDRV         ;LOAD DRIVE ADDRESS INTO QDRV
004406  104420                          DRV.RESET               ;RESET THE DRIVE
004410  005737  002060                  TST     XXDPMD          ;PROGRAM LOADED FROM AN RK05 ?
004414  001403                          BEQ     2$              ;BR IF NOT
004416  120037  002060                  CMPB    R0,XXDPMD       ;LOADED FROM THIS RK05 ?
004422  001404                          BEQ     3$              ;BR IF YES
004424  110021                  2$:     MOVB    R0,(R1)+        ;STORE THE DRIVE NUMBER
004426  010223                          MOV     R2,(R3)+        ;STORE ADDRESS IN KEY TABLE
004430  005237  001264                  INC     DRVPRS          ;BUMP THE NUMBER OF DRIVES COUNTER
004434  062702  020000          3$:     ADD     #20000,R2       ;NEXT DRIVE ADDRESS
004440  005200                          INC     R0              ;NEXT DRIVE NUMBER
004442  022700  000010                  CMP     #8.,R0          ;DONE ALL DRIVES???
004446  001347                          BNE     1$              ;LOOP TILL DONE
004450  013703  001264                  MOV     DRVPRS,R3       ;FIND WHICH DRIVES ARE TYPE F
004454  001510                          BEQ     ST4             ;BR IF NOT DRIVES PRESENT
004456  012701  001254                  MOV     #PDR,R1
004462  005000                          CLR     R0
004464  005002                          CLR     R2
004466  104401  002362                  TYPE    ,MSG20
004472  111102                  4$:     MOVB    (R1),R2         ;GET DRIVE NUMBER
004474  010200                          MOV     R2,R0
004476  002472                          BLT     9$

004500  104401  002372                  TYPE    ,MSG24

004504  010246                          MOV     R2,-(SP)        ;TYPE THE DRIVE NUMBER
004506  104403                          TYPOS
004510     001                          .BYTE   1
004511     000                          .BYTE   0
004512  000241                          CLC                     ;MOVE DRIVE NUMBER TO BITS 15,14,13
004514  006002                          ROR     R2              ;BIT0 TO CARRY
004516  006002                          ROR     R2              ;BIT0 TO BIT15
004520  006002                          ROR     R2              ;BIT0 TO BIT14
004522  006002                          ROR     R2              ;BIT0 TO BIT13
004524  042702  017777                  BIC     #17777,R2       ;CLEAR ANY EXTRANEOUS BITS

004530  010237  001502                  MOV     R2,QDRV
004534  104420                          DRV.RESET               ;RESET THE DRIVE VIA QDRV

004536  032702  020000                  BIT     #20000,R2       ;EVEN DRIVE NUMBER???
004542  001003                          BNE     5$              ;NO - CLEAR BIT

004544  052702  020000                  BIS     #20000,R2       ;MAKE IT AN ODD DRIVE
004550  000402                          BR      6$

```

```
004552  042702  020000          5$:     BIC     #20000,R2       ;MAKE IT AN EVEN DRIVE

004556  010277  174446          6$:     MOV     R2,@RKDA        ;SELECT THE NEW DRIVE
004562  032777  000200  174426          BIT     #200,@RKDS      ;MAKE SURE DRIVE IS IN SYSTEM
004570  001420                          BEQ     8$              ;IF NOT, SKIP THIS TEST

004572  012777  000011  174422          MOV     #11,@RKCS       ;START A SEEK TO CYL 0
004600  104417                          CON.RDY                 ;WAIT FOR CONTROLLER
004602  032777  000100  174406          BIT     #100,@RKDS      ;IS IT IN MOTION???
004610  001010                          BNE     8$              ;NO - J TYPE DRIVE

004612  152711  000200                  BISB    #200,(R1)       ;YES - SET THE F TYPE BIT
004616  104401  002375                  TYPE    ,MSG25

004622  032777  000100  174366  7$:     BIT     #100,@RKDS      ;WAIT FOR HEADS TO STOP
004630  001774                          BEQ     7$

004632  105737  001253          8$:     TSTB    FRSTRT
004636  001012                          BNE     9$

004640  032777  000002  174272          BIT     #SW1,@SWR       ;SERIAL NO. SW SET?
004646  001406                          BEQ     9$              ;NO

004650  104401  002326                  TYPE    ,MSG17          ;TYPE "SR  NO"
004654  104413                          RDDEC                   ;READ FROM TTY INPUT
004656  006300                          ASL     R0              ;SAVE SERIAL NO FOR THE DRIVE
004660  012660  001266                  MOV     (SP)+,SRNO(R0)

004664  005201                  9$:     INC     R1
004666  005303                          DEC     R3
004670  003300                          BGT     4$
004672  104401  001213                  TYPE    ,$CRLF
```

```
                                        ;'MAXBA' IS THE HIGHEST BUS ADDRESS (MEMORY) TO WHICH DATA TRANSFERS CAN
                                        ;BE DONE BY THE PROGRAM.
                                        ;'MAXBA' IS FIGURED USING THE FOLLOWING ALGORITHM:

                                        ;1.  IF KT11 IS NOT PRESENT,
                                        ;A.  AND THE PROGRAM IS RUN UNDER XXDP, THEN THE TOP 1.5 K IS RESERVED
                                        ;AND THE  'MAXBA' IS COMPUTED ($LSTAD-6000).
                                        ;B.  AND THE PROGRAM IS NOT RUNNING UNDER XXDP, THEN THE TOP 320 WORDS
                                        ;ARE RESERVED FOR 'MOM',LOADER,ETC. AND THE 'MAXBA' IS COMPUTED ($LSTAD-500).

                                        ;2. IF KT11 IS PRESENT,
                                        ;A.  AND MORE THAN 28K MEMORY IS PRESENT, THEN THE MAXIMUM BUS ADDRESS
                                        ;IS 147776 (OCTAL).
                                        ;B.  AND LESS THAN 28K IS PRESENT, THEN THE TOP 2K IS RESERVED FOR RKDP
                                        ;MONITOR AND 'MAXBA' IS COMPUTED.
                                        ;FIGURE OUT THE AVAILABLE MEMORY AND 'MAXBA'

004676  004737  017374                  ST4:    JSR     PC,$SIZE        ;GO SIZE THE MEMORY
004702  012702  002052                          MOV     #BASEBA,R2      ;INITIALIZE POINTERS
004706  012703  002054                          MOV     #MAXBA,R3
004712  005737  017432                          TST     $KT11           ;KT11 AVAILABLE?
004716  100022                                  BPL     4$              ;NO
004720  013700  017700                          MOV     $LSTBK,R0       ;GET THE LAST BANK OF MEMORY
004724  020027  001540                          CMP     R0,#1540        ;28K OR MORE?
004730  002012                                  BGE     3$              ;YES

004732  162700  000040                          SUB     #40,R0          ;BACK UP 2 K'S (RKDP MONITOR, ETC.)
004736  012701  177772                          MOV     #-6,R1          ;AND FORM THE MAXIMUM BUS ADDRESS
                                                                        ;FOR DATA TRANSFER
004742  006300                          1$:     ASL     R0
004744  005201                                  INC     R1
004746  001375                                  BNE     1$
004750  162700  000002                          SUB     #2,R0
004754  000415                          2$:     BR      6$

004756  012713  147776                  3$:     MOV     #147776,(R3)    ;FOR 28K OR MORE, THIS IS THE 'MAXBA'
004762  000413                                  BR      7$

004764  013700  017676                  4$:     MOV     $LSTAD,R0       ;KT11 NOT PRESENT, GET THE LAST
                                                                        ;AVAILABLE ADDRESS

004770  005737  000040                  5$:     TST     @#40            ;'XXDP' LOADED PROGRAM ?
004774  001003                                  BNE     8$              ;YES
004776  162700  000500                          SUB     #500,R0         ;NO, SAVE THE LAST 320 WORDS
005002  000402                                  BR      6$
005004  162700  006000                  8$:     SUB     #6000,R0        ;SAVE THE LAST 1.5K OF MEMORY (RKDP
                                                                        ;MONITOR, ETC.)
005010  010013                          6$:     MOV     R0,(R3)         ;SAVE THE MAXIMUM BUS ADDRESS(MAXBA) TO
                                                                        ;WHICH DATA TRANSFER CAN BE DONE SAFELY
005012  012712  032412                  7$:     MOV     #PGEND,(R2)     ;'BASEBA'
005016  032777  000100  174114                  BIT     #SW06,@SWR
005024  001510                                  BEQ     ST3
005026  104401  005034                          TYPE    ,65$            ;;TYPE ASCIZ STRING
005032  000432                                  BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <15><12>/TYPE OCTAL BUS ADDRESSES FOR DATA XFER, BETWEEN /
005120                                  64$:
```

```
005120  011246                                  MOV     (R2),-(SP)      ;'BASEBA'
005122  104402                                  TYPOC
005124  104401  005132                          TYPE    ,67$            ;;TYPE ASCIZ STRING
005130  000402                                  BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  / & /
005136                                  66$:
005136  011346                                  MOV     (R3),-(SP)      ;'MAXBA'
005140  104402                                  TYPOC
005142                                  9$:
005142  104401  005150                          TYPE    ,69$            ;;TYPE ASCIZ STRING
005146  000407                                  BR      68$             ;;GET OVER THE ASCIZ
                                        ;;69$:  .ASCIZ  <15><12>/LO LIMIT? /
005166                                  68$:
005166  104412                                  RDOCT
005170  012600                                  MOV     (SP)+,R0
005172  020012                                  CMP     R0,(R2)         ;CORRECT LO LIMIT?
005174  103762                                  BLO     9$
005176  020013                                  CMP     R0,(R3)         ;CORRECT LO LIMIT?
005200  103360                                  BHIS    9$
005202  010012                                  MOV     R0,(R2)         ;'BASEBA'
005204                                  10$:
005204  104401  005212                          TYPE    ,71$            ;;TYPE ASCIZ STRING
005210  000407                                  BR      70$             ;;GET OVER THE ASCIZ
                                        ;;71$:  .ASCIZ  <15><12>/HI LIMIT? /
005230                                  70$:
005230  104412                                  RDOCT
005232  012600                                  MOV     (SP)+,R0
005234  020013                                  CMP     R0,(R3)         ;CORRECT HI LIMIT?
005236  101362                                  BHI     10$
005240  020012                                  CMP     R0,(R2)         ;CORRECT LO LIMIT?
005242  101760                                  BLOS    10$
005244  010013                                  MOV     R0,(R3)         ;'MAXBA'

005246  023727  002054  037476          ST3:    CMP     MAXBA,#37476    ;8K MEMORY - CLOBBER XXDPT
005254  002003                                  BGE     1$
005256  012737  037476  002054                  MOV     #37476,MAXBA    ;BUT SAVE LOADER
005264  105737  001253                  1$:     TSTB    FRSTRT          ;PROGRAM RESTARTED AT 210?
005270  001402                                  BEQ     BCTST           ;NO
005272  000137  007716                          JMP     EXRCSR          ;YES, SKIP TEST 1 TO 7
```

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

                                        ;FIND OUT THE DRIVE NUMBER TO BE TESTED.
005276  012737  001306  001476  RCTST:  MOV     #KEY,DRVPTR     ;INITIALIZE PTR TO DRV#
005304  013737  001264  001500          MOV     DRVPRS,DRVCNT   ;NUMBER OF DRIVES PRESENT
005312  017737  174160  001502  NXTDRV: MOV     @DRVPTR,QDRV    ;SAVE DRIVE #(BITS 15-13)
005320  062737  000002  001476          ADD     #2,DRVPTR       ;INCRMENT PTR TO NXT DRV#
005326  005337  001500                  DEC     DRVCNT          ;DONE ALL DRIVES?
005332  100002                          BPL     TST1            ;NO, GO TEST THIS DRIVE
005334  000137  007716                  JMP     EXRCSR          ;ALL DONE, GO TO EXERCISER PART
```

```
                                        ;;*************************************************************
                                        ;*TEST 1         PERFORM WRITE OF 401 WORDS (1 SECTOR + 1 WORDS)
                                                ;THIS TEST PERFORMS A WRITE OF 401 WORDS (1 SECTOR + 1WORD) AND
                                                ;CHECKS IF RKDA,RKBA,RKWC INCREMENTED CORRECTLY.WRITING IS DONE
                                                ;ON CYLINDER 0, SURFACE 0, SECTORS 0,1 AND 10,11. IT SHOULD BE
                                                ;NOTED THAT THIS IS A BOUNDARY CONDITION TRANSFER. THE VALIDITY
                                                ;OF THE TRANSFER IS CHECKED IN THE NEXT TEST.
                                                ;DATA PATTERN WRITTEN IS 111111.
                                        ;;*************************************************************
005340  000004                          TST1:   SCOPE

005342  013701  001502                          MOV     QDRV,R1         ;GET RKDA
005346  010102                                  MOV     R1,R2           ;SAVE RKDA
005350  062702  000002                          ADD     #2,R2           ;EXPCTD RKDA AFTER WRITE IS DONE

005354  012737  005362  001110                  MOV     #1$,$LPERR      ;RETURN ADDRESS FOR LUPING

005362  104416                          1$:     CON.RESET
005364  104420                                  DRV.RESET
005366  104416                                  CON.RESET               ;CLEAR MASK BITS IN POLLING LOGIC
005370  012737  111111  032412  2$:     MOV     #111111,DBUF    ;PATTERN TO BE WRITTEN
005376  012703  000401                          MOV     #401,R3         ;WORD COUNT FOR WRITE
005402  004737  006230                          JSR     PC,DOWRITE      ;GO DO WRITE

005406  104101                                  ERROR   101             ;INTERRUPT DID NOT OCCUR AFTER WRITE
005410  004737  020020                          JSR     PC,CHKCS        ;CHECK ERROR BIT IN RKCS
005414  104102                                  ERROR   102             ;ERROR BIT IN RKCS SET ON DOING WRITE
005416  004737  020034                          JSR     PC,CHKDA        ;CHECK IF RKDA INCREMENTED RIGHT
005422  104103                                  ERROR   103             ;RKDA DID NOT INCREMENT RIGHT AFTER
                                                                        ;A WRITE OF 401 WORDS.
005424  004737  020136                          JSR     PC,CHKWC        ;CHECK IF RKWC OVERFLOWED TO 0
005430  104105                                  ERROR   105             ;RKWC DID NOT OVERFLOW TO 0 AFTER
                                                                        ;A WRITE OF 401 WORDS.
005432  032701  000010                          BIT     #10,R1          ;SECTORS 10,11 WRITTEN?
005436  001005                                  BNE     TST2            ;YES
005440  062701  000012                          ADD     #12,R1          ;RKDA TO BE USED NEXT (SEC 10)
005444  062702  000016                          ADD     #16,R2          ;EXPCTD RKDA AFTER WRITE IS DONE
005450  000747                                  BR      2$              ;GO WRITE SECS 10,11
```

```
                                        ;;*************************************************************
                                        ;*TEST 2         READ & CHECK THAT 401 WORD WRITE WAS DONE CORRECTLY
                                                ;THIS TEST PERFORMS A READ OF THE 401 WORDS WRITTEN IN THE
                                                ;PREVIOUS TEST AND CHECKS THAT THEY WERE CORRECTLY READ.MOREOVER
                                                ;IT CHECKS THAT ONLY ONE NON-ZERO WORD (401TH) WAS WRITTEN IN THE
                                                ;SECOND SECTOR AND THE REST OF THE WORDS ARE ALL ZEROS.
                                        ;;*************************************************************
005452  000004                          TST2:   SCOPE

005454  013701  001502                          MOV     QDRV,R1         ;GET DRIVE #
005460  012737  005466  001110                  MOV     #1$,$LPERR      ;ADDRESS FOR LUPING ON EROR
005466  104416                          1$:     CON.RESET
005470  104420                                  DRV.RESET
005472  104416                                  CON.RESET

                                                                        ;CLEAN UP THE DATA BUFFER
005474  004737  006360                          JSR     PC,CLEANBUF     ;INTO WHICH READ WILL
                                                                        ;BE DONE
005500  012703  001000                          MOV     #1000,R3        ;WORD COUNT

005504  004737  006350                          JSR     PC,DOREAD       ;GO DO A READ OF 2 SECTORS
                                                                        ;FROM DISK ADDRESS GIVEN IN R1
005510  104101                                  ERROR   101             ;INTERRUPT DID NOT OCCUR AFTER
                                                                        ;READ OF 401 WORDS WAS DONE.
005512  004737  020020                          JSR     PC,CHKCS        ;CHECK IF EROR BIT IN RKCS SET?
005516  104102                                  ERROR   102             ;EROR BIT IN RKCS SET ON DOING A
                                                                        ;READ OF 401 WORDS.
005520  012704  032412                          MOV     #DBUF,R4        ;STARTING BUS ADDRESS, INTO WHICH READ
005524  010402                                  MOV     R4,R2           ;WAS DONE
005526  004737  020056                          JSR     PC,CHKBA        ;CHECK IF RKBA INCREMENTED RIGHT
005532  104104                                  ERROR   104             ;RKBA DID NOT INCREMENT RIGHT AFTER READ
                                                                        ;OF 401 WORDS.
005534  012705  177764                          MOV     #-14,R5         ;ALLOW 12 ERRORS, AT THE MOST
005540  022712  111111                  2$:     CMP     #111111,(R2)    ;CORRECT DATA READ?
005544  001410                                  BEQ     3$              ;YES
005546  012737  111111  001164                  MOV     #111111,$REG1   ;GET EXPCTD DATA WORD
005554  004737  005644                          JSR     PC,ERINF1       ;GET ERROR INFORMATION
005560  104100                                  ERROR   100             ;DATA ERROR OCCURRED WHEN A
                                                                        ;READ OF 401 WORDS WAS DONE
                                                                        ;THE DISK ADDRESS FROM WHERE
                                                                        ;THE DATA WAS READ INCORRECTLY
                                                                        ;IS GIVEN IN THE ERROR MESSAGE

005562  005205                                  INC     R5              ;REPORT 12 ERORS AT MOST
005564  001421                                  BEQ     6$
005566  005722                          3$:     TST     (R2)+           ;INCREMENT POINTER
005570  020227  033414                          CMP     R2,#DBUF+1002   ;CHECKED ALL 401 WORDS?
005574  001361                                  BNE     2$

005576  005712                          4$:     TST     (R2)            ;CHECK THAT REST OF 377 WORDS
005600  001407                                  BEQ     5$              ;ARE ALL 0'S
005602  005037  001164                          CLR     $REG1           ;GET EXPCTD DATA WORD (0)
005606  004737  005644                          JSR     PC,ERINF1       ;GET ERROR INFO
005612  104100                                  ERROR   100             ;DATA ERROR. IN A PREVIOUS
                                                                        ;TEST A WRITE OF 401 WORDS
                                                                        ;(1 SECTOR + 1 WORD) WAS DONE
```

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

005614  005205                                  INC     R5              ;REPORT 12 ERORS AT MOST
005616  001404                                  BEQ     6$
005620  005722                          5$:     TST     (R2)+           ;ALL WORDS CHECKED?
005622  020227  034412                          CMP     R2,#DBUF+2000
005626  001363                                  BNE     4$              ;IF NOT GO BAK

005630  032701  000010                  6$:     BIT     #10,R1          ;WERE SECTORS 10,11 READ
005634  001030                                  BNE     TST3            ;YES
005636  062701  000012                          ADD     #12,R1          ;FROM NEW RKDA, SEC 10
005642  000711                                  BR      1$              ;GO BACK AND READ FROM SECS 10,11

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

005644  010237  001162                  ERINF1: MOV     R2,$REG0        ;GET BUS ADDRESS OF DATA ERROR
005650  011237  001166                          MOV     (R2),$REG2      ;GET BAD DATA WORD (READ)
005654  010146                                  MOV     R1,-(SP)
005656  020227  033410                          CMP     R2,#DBUF+776    ;FIGURE OUT THE DISK ADDRESS
005662  003001                                  BGT     1$              ;WHERE DATA ERROR OCCURRED
005664  005316                                  DEC     (SP)
005666  005216                          1$:     INC     (SP)
005670  032716  000010                          BIT     #10,(SP)
005674  001405                                  BEQ     2$
005676  032716  000004                          BIT     #4,(SP)
005702  001402                                  BEQ     2$
005704  062716  000004                          ADD     #4,(SP)
005710  012637  001170                  2$:     MOV     (SP)+,$REG3
005714  000207                                  RTS     PC
```

```
                                        ;;*************************************************************
                                        ;*TEST 3         PERFORM WRITE OF 12 SECTORS + 1 WORD
                                                ;THIS TEST CHECKS FOR ANOTHER BOUNDARY CONDITION.  IT
                                                ;PERFORMS A WRITE OF 12 SECTORS + 1 WORD.  RKDA,RKBA,
                                                ;RKWC ARE CHECKED TO SEE IF THEY ARE INCREMENTED CORRECTLY.
                                                ;VALIDITY OF THE DATA WRITTEN IS CHECKED IN THE NEXT
                                                ;TEST.  DATA IS WRITTEN ON SECTORS 0-11, SURFACE 0
                                                ;CYLINDER 0 (6001TH WORD ON SECTOR 0, SURFACE 1, CYL 0).
                                                ;ALSO ON SECTORS 0-11, SURFACE 1 (6001TH WORD ON SECTOR
                                                ;0, CYL 1)
                                        ;;*************************************************************
005716  000004                          TST3:   SCOPE
005720  013701  001502                          MOV     QDRV,R1         ;GET DRIVE #
005724  012737  005744  001110                  MOV     #1$,$LPERR      ;LUP ON EROR TO '1$'
005732  010102                                  MOV     R1,R2
005734  062702  000021                          ADD     #21,R2
005740  012703  006001                          MOV     #6001,R3
005744  104416                          1$:     CON.RESET
005746  104420                                  DRV.RESET
005750  104416                                  CON.RESET


005752  012737  044444  032412                  MOV     #44444,DBUF     ;PATTERN TO BE WRITTEN

005760  004737  006230                          JSR     PC,DOWRITE      ;GO DO WRITE

005764  104101                                  ERROR   101             ;INTERRUPT DID NOT OCCUR ON
                                                                        ;COMPLETION OF WRITE
005766  004737  020020                          JSR     PC,CHKCS        ;CHECK IF EROR BIT IN RKCS SET
005772  104102                                  ERROR   102             ;EROR BIT IN RKCS SET ON DOING WRITE
005774  004737  020034                          JSR     PC,CHKDA        ;CHECK IF RKDA INCREMENTED RIGHT
006000  104103                                  ERROR   103             ;RKDA DID NOT INCREMENT CORRECTLY
                                                                        ;AFTER A WRITE OF 6001 (OCTAL) WORDS.
                                                                        ;(12 SECTORS + 1)
006002  004737  020136                          JSR     PC,CHKWC        ;CHECK IF RKWC OVERFLOWED TO 0
006006  104105                                  ERROR   105             ;RKWC DID NOT OVERFLOW TO 0
006010  032701  000020                          BIT     #20,R1          ;WRITTEN ON SURFACE 1?
006014  001006                                  BNE     TST4            ;YES
006016  010201                                  MOV     R2,R1
006020  062702  000020                          ADD     #20,R2          ;SURFACE 1
006024  012703  005401                          MOV     #5401,R3        ;WORD COUNT
006030  000745                                  BR      1$              ;GO WRITE SURFACE 1
```

```
                                        ;;*************************************************************
                                        ;*TEST 4         READ & CHECK THAT 6001 WORD WRITE WAS DONE CORRCTLY
                                                ;THIS TEST CHECKS THAT THE 6001-WORD WRITE THAT WAS DONE IN THE
                                                ;PREVIOUS TEST WAS CORRECT, ESPECIALLY THE LAST 401 WORDS. THE
                                                ;FIRST WORD OF THE SECTOR( IN WHICH THE 6001TH WORD) IS WRITTEN
                                                ;IS THE ONLY NON-ZERO WORD IN THAT SECTOR. THE REST 377 WORDS ARE
                                                ;ALL ZEROS, IF THE WRITING WAS DONE CORRECTLY.
                                        ;;*************************************************************
006032  000004                          TST4:   SCOPE
006034  013701  001502                          MOV     QDRV,R1         ;GET DRIVE #
006040  062701  000013                          ADD     #13,R1          ;DISK ADDRESS FROM WHERE READ IS DONE
006044  012737  006052  001110                  MOV     #1$,$LPERR
006052  104416                          1$:     CON.RESET
006054  104420                                  DRV.RESET
006056  104416                                  CON.RESET

006060  004737  006360                          JSR     PC,CLEANBUF     ;CLEAN UP THE BUFFER INTO WHICH
                                                                        ;READ WILL BE DONE
                                                                        ;SET UP RKDA
                                                                        ;SECTOR 11, SURFACE 0
006064  012703  001000                          MOV     #1000,R3        ;WORD COUNT

006070  004737  006350                          JSR     PC,DOREAD       ;GO READ 1000 WORDS (2 SECS)
006074  104101                                  ERROR   101             ;INTERRUPT DID NOT OCCUR AFTER
                                                                        ;COMPLETION OF READ
006076  004737  020020                          JSR     PC,CHKCS        ;CHECK IF EROR BIT IN RKCS SET
006102  104102                                  ERROR   102             ;EROR (RKCS) SET ON DOING READ
006104  012704  032412                          MOV     #DBUF,R4        ;STARTING BA OF DATA BUFER
006110  010402                                  MOV     R4,R2
006112  004737  020056                          JSR     PC,CHKBA        ;RKBA INCREMENTED CORRECTLY?
006116  104104                                  ERROR   104             ;RKBA DID NOT INCREMENT CORRECTLY
006120  012705  177764                          MOV     #-14,R5
006124  022712  044444                  2$:     CMP     #44444,(R2)     ;DATA WORD OK?
006130  001410                                  BEQ     3$
006132  012737  044444  001164                  MOV     #44444,$REG1    ;NO, GET EXPCTD DATA WORD
006140  004737  005644                          JSR     PC,ERINF1       ;GET, OTHER ERROR INFO
006144  104100                                  ERROR   100             ;DATA ERROR.  A WRITE OF 6001
                                                                        ;WORDS (12 SECS + 1 WORD) WAS DONE
                                                                        ;IN A PREVIOUS TEST.  THE LAST TWO
                                                                        ;SECTORS (LAST 401 WORDS) WERE READ
                                                                        ;BACK.  THIS ERROR INDICATES THAT
                                                                        ;SEC #11 (LAST BUT ONE SECTOR) GAVE
                                                                        ;BAD DATA WORDS
006146  005205                                  INC     R5              ;REPORT 12 ERORS AT MOST
006150  001421                                  BEQ     6$
006152  005722                          3$:     TST     (R2)+           ;INCREMENT POINTER TO BA
006154  020227  033414                          CMP     R2,#DBUF+1002   ;CHECKED 401 WORDS?
006160  001361                                  BNE     2$
006162  005712                          4$:     TST     (R2)            ;CHECK THAT THE REMAINING 377
006164  001407                                  BEQ     5$              ;WORDS OF THE LAST SECTOR (SEC #0)
006166  005037  001164                          CLR     $REG1           ;WERE READ BACK AS 0'S
006172  004737  005644                          JSR     PC,ERINF1
006176  104100                                  ERROR   100             ;DATA ERROR.  IF WRITE WAS DONE CORRECTLY
                                                                        ;IN THE PREVIOUS TEST, THE LAST SECTOR
                                                                        ;OF THE DATA BLOCK (12 SECS + 1 WORD)
                                                                        ;SHOULD CONTAIN ONLY 1 (FIRST) WORD
```

```
                                                                ;AS NON-ZERO, THE REST 377 SHOULD BE
                                                                ;ALL 0'S.  THIS ERROR INDICATES THAT
                                                                ;THE SOME OF 377 WORDS
                                                                ;WERE NOT CORRECT
006200  005205                          INC     R5              ;REPORT 12 ERORS AT MOST
006202  001404                          BEQ     6$
006204  005722                  5$:     TST     (R2)+           ;INCREMENT POINTER
006206  020227  034412                  CMP     R2,#DBUF+2000   ;CHECKED ALL WORDS?
006212  001363                          BNE     4$              ;NO

006214  032701  000020          6$:     BIT     #20,R1          ;DONE CHECKING FOR SURFACE 1?
006220  001070                          BNE     TST5            ;YES
006222  062701  000020                  ADD     #20,R1          ;SO SET UP FOR SURFACE 1
006226  000711                          BR      1$              ;GO BACK & READ SURFACE 1
```

```
                                ;DOWRITE
                                ;THIS ROUTINE PERFORMS A WRITE ON A DISK.AT THE TIME OF ENTRY, R1 CONTAINS
                                ;DISK ADDRESS (RKDA) WHERE WRITE IS TO BE DONE, R3 CONTAINS THE WORD COUNT
                                ;(RKWC), 'DBUF' CONTAINS THE DATA TO BE WRITTEN. NOTE IBA BIT IS SET.

                                ;WRITE IS DONE IN INTERRUPT MODE. IF THE INTERRUPT DOES NOT OCCUR WITHIN
                                ;A CERTAIN TIME, RETURN IS MADE TO THE ERROR MESSAGE FOLLOWING THE 'JSR'
                                ;CALL. IF THE INTERRUPT OCCURS, RETURN ADDRESS IS ADJUSTED TO SKIP OVER
                                ;THE ERROR MESSAGE.

006230  012777  004002  172764  DOWRITE: MOV    #4002,@RKCS     ;WRITE, IBA
006236  010177  172766          DOXFER: MOV     R1,@RKDA        ;ADDRESS THE DRIVE
006242  010377  172756                  MOV     R3,@RKWC        ;XFER THIS # OF WORDS
006246  005477  172752                  NEG     @RKWC
006252  012777  032412  172746          MOV     #DBUF,@RKBA     ;USE THIS BUS ADDRESS
006260  012777  006340  172752          MOV     #3$,@RKVEC      ;SET UP INTERRUPT VECTOR
006266  005046                          CLR     -(SP)           ;NEW PSW
006270  012746  006276                  MOV     #1$,-(SP)       ;SET NEW PC TO STACK **********
006274  000002                          RTI
006276  052777  000101  172716  1$:     BIS     #101,@RKCS      ;SET IDE, GO (WRITE,IBA/ READ)
006304  005037  001466                  CLR     CICNT
006310  012737  177760  001470          MOV     #-20,CICNT1
006316  005237  001466          2$:     INC     CICNT           ;WAIT  FOR INTERRUPT
006322  001375                          BNE     .-4

006324  005237  001470                  INC     CICNT1
006330  001372                          BNE     2$

006332  004737  021740                  JSR     PC,GT4RG        ;TIMED OUT, INTERRUPT DID NOT OCCUR
006336  000207                          RTS     PC              ;RETURN TO THE EROR MEASGE

006340  022626                  3$:     CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
006342  062716  000002                  ADD     #2,(SP)         ;ADJUST RETURN ADDRESS TO SKIP OVER
006346  000207                          RTS     PC              ;EROR MESAGE ON RETURN
```

```
                                ;THIS ROUTINE PERFORMS A READ ON THE DISK.AT THE TIME OF ENTRY R1 CONTAINS
                                ;THE DISK ADDRESS FROM WHERE THE READ IS TO BE DONE. R3 CONTAINS THE WORD
                                ;COUNT (RKWC). READ WILL BE DONE INTO DATA BUFFER AT 'DBUF'.

                                ;READ IS DONE IN INTERRUPT MODE. IF THE INTERRUPT DOES NOT OCCUR WITHIN
                                ;A CERTAIN TIME, RETURN IS MADE TO THE ERROR MESSAGE FOLLOWING THE 'JSR'
                                ;CALL. IF THE INTERRUPT OCCURS AS EXPECTED, RETURN ADDRESS IS ADJUSTED
                                ;TO SKIP OVER THE ERROR MESSAGE.

ØØ635Ø  Ø12777  ØØØØØ4  172644  DOREAD: MOV     #4,@RKCS        ;READ
ØØ6356  ØØØ727                          BR      DOXFER


                                ;CLEANBUF
                                ;CLEANS OUT THE DATA BUFFER (ALL WORDS WRITTEN TO 177777) INTO WHICH THE
                                ;READ FROM THE DISK WILL BE DONE.  DATA BUFFER STARTS AT 'DBUF'AND IS
                                ;1ØØØ (OCTAL) WORDS LONG.

ØØ636Ø  Ø127Ø2  177ØØØ          CLEANBUF:MOV    #-1ØØØ,R2       ;SET COUNT
ØØ6364  Ø127Ø5  Ø32412                  MOV     #DBUF,R5        ;INITIALIZE BA
ØØ637Ø  Ø12725  Ø22222          1$:     MOV     #22222,(R5)+
ØØ6374  ØØ52Ø2                          INC     R2              ;DONE ALL WORDS?
                                                                ;BUFFER STARTING AT (PHYSICAL) BUS
                                                                ;ADDRESS 177ØØØ (177ØØØ-2ØØ776)
ØØ6376  ØØ1374                          BNE     1$
ØØ64ØØ  ØØØ2Ø7                          RTS     PC              ;YES RETURN
```

```
                                ;;*********************************************************************
                                ;*TEST 5         CHECK DATA TRANSFER AROUND 32K BOUNDARY
                                        ;*THIS TEST PERFORMES A WRITE OF 2 SECTORS ON THE DISK FROM MEMORY
                                        ;*LOCATIONS AROUND THE 32K BOUNDARY.  SECTORS Ø,1, CYL Ø, SURFACE
                                        ;*Ø ARE WRITTEN.  PHYSICAL BUS ADDRESSES FOR THE DATA BUFFER:
                                        ;* 177ØØØ TO 2ØØ776 I.E. (32K-256) TO (32K+255)

                                        ;*CHECKING IS DONE TO SEE IF MEX BITS, RKBA,RKDA,RKWC INCREMENTED
                                        ;*CORRECTLY. THEN DATA BUFFER IS CLEARED OUT AND A READ IS DONE
                                        ;*INTO IT.  A CHECK IS MADE TO SEE IF THE CORRECT DATA WAS RECIEVED.
                                        ;*ONLY 12 DATA ERRORS ARE REPORTED.
                                ;;*********************************************************************
ØØ64Ø2  ØØØØØ4                  TST5:   SCOPE
ØØ64Ø4  Ø23727  Ø177ØØ  ØØ2ØØØ          CMP     #LSTBK,#2ØØØ    ;33K OR MORE OF MEMORY?
ØØ6412  1Ø3ØØ2                          BHIS    1$              ;YES
ØØ6414  ØØØ137  ØØ6764                  JMP     TST6            ;IF NOT, DONT DO THIS TEST
ØØ642Ø  Ø12737  ØØ16ØØ  172352  1$:     MOV     #16ØØ,@#KIPAR5  ;MAP 28-32K THRU PAR 5
ØØ6426  Ø12737  ØØ2ØØØ  172354          MOV     #2ØØØ,@#KIPAR6  ;MAP 32-36K THRU PAR 6
ØØ6434  Ø12737  ØØØØØ1  177572          MOV     #1,@#SRØ        ;TURN ON MEMORY MANAGEMENT

                                ;SET UP DATA BUFFER (1ØØØ OCTAL WORDS LONG) FOR WRITING TWO SECTORS
                                ;ON THE DISK.  THE TRANSFER IS DONE AROUND THE 32K BOUNDARY FROM
                                ;BUS ADDRESS (PHYSICAL) 177ØØØ TO 2ØØ776, (32K-256) TO (32+256)

                                ;DATA IN THE BUFFER IS A COUNT PATTERN STARTING FROM 1 FOR THE FIRST
                                ;WORD TO (ØØØ FOR THE LAST WORD.
                                ;PHYSICAL BUFFER ADDRESS:  177ØØØ TO  2ØØ776 (32K-256 TO 32K+256)
ØØ6442  Ø12737  ØØ645Ø  ØØ111Ø          MOV     #2$,$LPERR      ;LUP TO 2$ ON EROR (SW 9)
ØØ645Ø  1Ø4416                  2$:     CON.RESET
ØØ6452  1Ø442Ø                          DRV.RESET

ØØ6454  Ø127ØØ  ØØØØØ1                  MOV     #1,RØ           ;INITIALIZE DATA PATTERN TO BE
                                                                ;WRITTEN
ØØ646Ø  Ø127Ø1  137ØØØ                  MOV     #137ØØØ,R1      ;BA TO START PHYSICAL ADDRESS=177ØØØ
ØØ6464  Ø1ØØ21                  3$:     MOV     RØ,(R1)+        ;WRITE COUNT PATTERN (1-1ØØØ)
ØØ6466  ØØ52ØØ                          INC     RØ              ;INTO DATA BUFFER (PHYS ADDRES
ØØ647Ø  Ø2ØØ27  ØØ1ØØ1                  CMP     RØ,#1ØØ1        ;177ØØØ TO 2ØØ776)
ØØ6474  ØØ1373                          BNE     3$
ØØ6476  Ø13777  ØØ15Ø2  172524          MOV     QDRV,@RKDA      ;SUR Ø, SEC Ø, CYL Ø
ØØ65Ø4  Ø12777  177ØØØ  172512          MOV     #-1ØØØ,@RKWC    ;WORD COUNT =2 SECS
ØØ6512  Ø12777  177ØØØ  1725Ø6          MOV     #177ØØØ,@RKBA   ;BUS ADDRESS,
ØØ652Ø  Ø12777  ØØØØØ3  172474          MOV     #3,@RKCS        ;WRITE,GO
ØØ6526  1Ø4417                          CON.RDY                 ;WAIT FOR CNTROL RDY

ØØ653Ø  ØØ4737  Ø2ØØ2Ø                  JSR     PC,CHKCS        ;ANY EROR IN RKCS?
ØØ6534  1Ø41Ø2                          ERROR   1Ø2             ;'ERR' SET IN RKCS, ON DOING A
                                                                ;WRITE OF SECTORS (Ø,1) FROM
                                                                ;(PHYSICAL) BUS ADDRESSS 177ØØØ
                                                                ;(177ØØØ TO 2ØØ776)
ØØ6536  ØØ4737  Ø2Ø112                  JSR     PC,CHKMEX
                                                                ;CHECK THAT RKBA OVERFLOWED INTO
                                                                ;EXTENDED MEM. BIT (Ø) OF RKCS (BIT 4)
                                                                ;EX MEM BIT Ø SET?
                                                                ;GET RKCS,ER,DS,DA
ØØ6542  1Ø41Ø6                          ERROR   1Ø6             ;MEX BITS INCORRECT. BIT 4 OF RKCS
                                                                ;(MEX BIT Ø) SHOULD HAVE SET
```

```
                                                                 ;AFTER RKBA OVERFLOWED ON DOING
                                                                 ;A TRANSFER OF 2 SECTORS FROM BUS
                                                                 ;ADDRESS (PHYSICAL) STARTING AT 177ØØØ
ØØ6544  Ø127Ø3  ØØ1ØØØ                  MOV     #1ØØØ,R3         ;WORD COUNT
ØØ655Ø  Ø127Ø4  177ØØØ                  MOV     #177ØØØ,R4       ;STARTING BUS ADDRESS
ØØ6554  ØØ4737  Ø2ØØ56                  JSR     PC,CHKBA         ;CHECK IF RKBA INCREMENTED CORRECTLY
ØØ656Ø  1Ø41Ø4                          ERROR   1Ø4              ;;RKBA DID NOT INCREMENT CORRECTLY
                                                                 ;AFTER WRITE IF 2 SECTORS FROM A DATA
                                                                 ;BUFFER STARTING AT (PHYSICAL) BUS
                                                                 ;ADDRESS (177ØØØ-2ØØ776)
ØØ6562  Ø137Ø2  ØØ15Ø2                  MOV     QDRV,R2          ;GET EXPECTED
ØØ6566  Ø627Ø2  ØØØØØ2                  ADD     #2,R2            ;DISK ADDRESS
ØØ6572  ØØ4737  Ø2ØØ34                  JSR     PC,CHKDA         ;CHECK IF RKDA INCREMENTED CORRECTLY
ØØ6576  1Ø41Ø3                          ERROR   1Ø3              ;RKDA INCREMENTED WRONGLY AFTER
                                                                 ;A WROTE OF 2 SECTOR (Ø,1) FROM
                                                                 ;DATA BUFFER STARTING AT BUS
                                                                 ;ADDRESS (PHYSICAL) 177ØØØ.

ØØ66ØØ  ØØ4737  Ø2Ø136                  JSR     PC,CHKWC         ;CHECK IF RKWC OVERFLOWED CORRECTLY
ØØ66Ø4  1Ø41Ø5                          ERROR   1Ø5              ;RKWC DID NOT OVERFLOW TO Ø ON
                                                                 ;DOING A WRITE OF 2 SECTORS FROM
                                                                 ;BA = 177ØØØ

                                                                 ;NOW, READ IS DONE OF THE DATA
                                                                 ;THAT WAS WRITTEN TO SEE IF IT
                                                                 ;CAN BE READ CORRECTLY

ØØ66Ø6  Ø12737  ØØ6614  ØØ111Ø          MOV     #4$,$LPERR       ;LUP TO 4$ ON EROR (SW 9)
ØØ6614  1Ø4416                  4$:     CON.RESET
ØØ6616  1Ø442Ø                          DRV.RESET

ØØ662Ø  Ø127Ø1  137ØØØ                  MOV     #137ØØØ,R1       ;CLEAR THE 1ØØØ-WORD DATA
ØØ6624  ØØ5Ø21                  5$:     CLR     (R1)+            ;BUFFER (PA 177ØØØ TO 2ØØ776)
ØØ6626  Ø2Ø127  141ØØØ                  CMP     R1,#141ØØØ       ;ALL DONE?
ØØ6632  ØØ1374                          BNE     5$

                                                                 ;NOW, READ BACK INTO THE
                                                                 ;SAME BUFFER 2 SECTORS
                                                                 ;WRITTEN PREVIOUSLY

ØØ6634  Ø13777  ØØ15Ø2  172366          MOV     QDRV,@RKDA       ;ADDRESS THE DRIVE CYL Ø, SEC Ø, Ø
ØØ6642  Ø12777  177ØØØ  172354          MOV     #-1ØØØ,@RKWC     ;READ 2 SECTORS
ØØ665Ø  Ø12777  177ØØØ  17235Ø          MOV     #177ØØØ,@RKBA    ;INTO THIS BUS ADDRESS
ØØ6656  Ø12777  ØØØØØ5  172336          MOV     #5,@RKCS         ;READ, GO

ØØ6664  1Ø4417                          CON.RDY                  ;WAIT FOR CNTROL RDY

ØØ6666  ØØ4737  Ø2ØØ2Ø                  JSR     PC,CHKCS         ;ANY ERROR IN RKCS?
ØØ6672  1Ø41Ø2                          ERROR   1Ø2              ;ERROR BIT SET IN RKCS ON DOING
                                                                 ;A READ OF 2 SECTORS (Ø,1), CYL Ø,
                                                                 ;SUR Ø, INTO DATA BUFFER STARTING
                                                                 ;AT BUS ADDRESS 177ØØØØ, NOTE AFTER
                                                                 ;1777776, RKBA WILL OVERFLOW (Ø)
                                                                 ;INTO MEX BITS (BIT 4) OF RKCS.
                                                                 ;IF THE ENTIRE TRANSFER (1ØØØ WORD)
                                                                 ;WAS DONE RKBA WILL CONTAIN 1ØØØ
```

```
                                                                 ;AND BIT 4 OF RKCS (MEX) WILL BE SET.

ØØ6674  Ø127Ø5  177764          6$:     MOV     #-14,R5          ;REPORT ONLY 12 ERRORS.
ØØ67ØØ  Ø127Ø2  137ØØØ                  MOV     #137ØØØ,R2       ;STARTING ADDRESS OF BUFFER
                                                                 ;(PA=177ØØØ)
ØØ67Ø4  Ø127Ø1  ØØØØØ1                  MOV     #1,R1            ;INITIALIZE DATA PATTERN

ØØ671Ø  Ø2Ø112                  7$:     CMP     R1,(R2)          ;CORRECT DATA WORD RECVD?
ØØ6712  ØØ1414                          BEQ     8$
ØØ6714  ØØ57Ø5                          TST     R5               ;REPORT THIS ERROR?
ØØ6716  ØØ142Ø                          BEQ     9$
ØØ672Ø  ØØ52Ø5                          INC     R5               ;COUNT ERORS

ØØ6722  Ø1Ø137  ØØ1162                  MOV     R1,$REGØ         ;GET EXPCTED DATA WORD

ØØ6726  Ø11237  ØØ1164                  MOV     (R2),$REG1       ;GET DATA WORD RECVD
ØØ6732  1Ø41Ø7                          ERROR   1Ø7              ;DATA COMPARISON ERROR ON DOING A
                                                                 ;READ OF 2 SECTORS (Ø,1, CYL Ø, SURFACE Ø)
                                                                 ;INTO DATA BUFFER (PHYSICAL ADDRESS
                                                                 ;177ØØØ TO 2ØØ776)

ØØ6734  1Ø4421  ØØ2214                  TYPMSG  ,MSG13           ;TYPE 'PHYSICAL BUS ADDRESS'
ØØ674Ø  ØØ4737  Ø177Ø2                  JSR     PC,TYPDBO        ;TYPE THE 6 DIGIT PHYSICAL BUS ADDRESS
                                                                 ;WHERE THE DATA ERROR OCCURRED.

ØØ6744  Ø627Ø2  ØØØØØ2          8$:     ADD     #2,R2            ;INCREMENT POINTER TO BA
ØØ675Ø  ØØ52Ø1                          INC     R1
ØØ6752  Ø227Ø1  ØØ1ØØ1                  CMP     #1ØØ1,R1         ;CHECKED THE ENTIRE BUFFER?
ØØ6756  ØØ1354                          BNE     7$

                                                                 ;**OFF
ØØ676Ø  ØØ5Ø37  177572          9$:     CLR     @#SRØ            ;TURN OFF MEMORY MANAGEMENT
```

```
                                        ;;*************************************************************
                                        ;*TEST 6         CHECK DATA TRANSFER FROM 28K TO 32K
                                                ;*THIS TEST DOES A WRITE OF 4K WORDS FROM A BUFFER LOCATED AT 28K
                                                ;*THEN THE DATA IS READ BACK INTO THE SAME BUFFER(28K-32K) AND IS
                                                ;*CHECKED TO SEE IF IT IS CORRECT.  NOTE THAT THE BUFFER IS FILLED
                                                ;*WITH ALL 1'S BEFORE DOING THE READ.

                                                ;*THE WRITE IS DONE STARTING AT CYLINDER 0, SECTOR 0, SURFACE 0.
                                        ;;*************************************************************
 006764  000004                         TST6:   SCOPE

 006766  023727  017700  001740                 CMP     $LSTBK,#1740    ;32K OR MORE OF MEMORY?
 006774  103002                                 BHIS    1$              ;YES
 006776  000137  007320                         JMP     TST7            ;IF NOT, DONT DO THIS TEST
 007002  012737  001600  172352         1$:     MOV     #1600,@#KIPAR5  ;MAP 28-32K THRU PAR 5
 007010  012737  000001  177572                 MOV     #1,@#SR0        ;TURN ON MEM MANAGEMENT

                                        ;WRITE A COUNT PATTERN (1-10000) INTO DATA BUFFER (PHYSICAL ADDRESS
                                        ;160000-177776).  THIS DATA BUFFER WILL BE USED FOR WRITING ON THE DISK.

 007016  012737  007024  001110                 MOV     #2$,$LPERR      ;LUP TO 2$ ON EROR
 007024  104416                         2$:     CON.RESET
 007026  104420                                 DRV.RESET
 007030  012700  000001                         MOV     #1,R0           ;INITIALIZE DATA PATTERN TO BE WRITTEN
 007034  012701  120000                         MOV     #120000,R1      ;INITIALIZE BA (PA=160000) 28K
 007040  010021                         3$:     MOV     R0,(R1)+        ;WRITE COUNT PATTERNS (1-10000)
 007042  005200                                 INC     R0              ;INTO DATA BUFFER (PA 160000 TO
 007044  020027  010001                         CMP     R0,#10001       ;177776, 28K-32K)
 007050  001373                                 BNE     3$

 007052  013777  001502  172150                 MOV     QDRV,@RKDA      ;WRITE ON SEC 0, CYL 0, SUR1
 007060  012777  170000  172136                 MOV     #-10000,@RKWC   ;4K WORDS
 007066  012777  160000  172132                 MOV     #160000,@RKBA   ;FROM THIS BUS ADDRESS
 007074  012777  000003  172120                 MOV     #3,@RKCS        ;WRITE, GO

 007102  104417                                 CON.RDY                 ;WAIT FOR CONTROL READY

 007104  004737  020020                         JSR     PC,CHKCS        ;ANY ERROR IN RKCS?
 007110  104102                                 ERROR   102             ;'ERR' BIT SET IN RKCS ON DOING
                                                                        ;A WRITE OF 4K WORDS FROM 160000
                                                                        ;(28K-32K).  DISK WRITE BEGAN ON
                                                                        ;SEC 0, CYL 0, SUR 0.  IF ALL 4K
                                                                        ;WORDS WERE WRITTEN RKBA SHOULD OVERFLOW
                                                                        ;AND CONTAIN 0.  BIT 4 OF RKCS
                                                                        ;(MEX BIT) SHOULD BE 1

 007112  004737  020112                         JSR     PC,CHKMEX       ;CHECK IF RKBA OVERFLOWED AND MEX
                                                                        ;BITS (4) IN RKCS WAS SET.
 007116  104106                                 ERROR   106             ;MEX BIT 4 NOT SET AFTER OVERLFOW OF
                                                                        ;RKBA.  WRITE OF 4K WORDS (28K-32K) WAS DONE.

                                                                        ;RETURN HERE IF NO ERROR
 007120  012703  010000                         MOV     #10000,R3       ;WORD COUNT
 007124  012704  160000                         MOV     #160000,R4      ;STARTING BUS ADDRESS
 007130  004737  020056                         JSR     PC,CHKBA        ;CHECK IF RKBA INCREMENTED CORRECTLY
 007134  104104                                 ERROR   104             ;RKBA DID NOT OVERFLOW TO AFTER A WRITE IF 4K
```

```
                                                                        ;WORDS (160000 TO 177776)

 007136  004737  020136                         JSR     PC,CHKWC        ;CHECK RKWC OVERFLOWED CORRECTLY
 007142  104105                                 ERROR   105             ;RKWC DID NOT OVEFLOW TO 0
                                                                        ;AFTER A WRITE OF 4K WORD (160000
                                                                        ;TO 177776)


 007144  012737  007152  001110                 MOV     #4$,$LPERR      ;RETURN ADDRESS FOR LUPING
 007152  104416                         4$:     CON.RESET
 007154  104420                                 DRV.RESET

 007156  012701  120000                         MOV     #120000,R1      ;CLEAR THE DATA BUFFER BEFORE
 007162  005021                         5$:     CLR     (R1)+           ;DOING A READ INTO IT
 007164  022701  140000                         CMP     #140000,R1
 007170  001374                                 BNE     5$

 007172  013777  001502  172030                 MOV     QDRV,@RKDA      ;READ FROM SEC 0, SUR 0, CYL 0
 007200  012777  170000  172016                 MOV     #-10000,@RKWC   ;4K WORDS
 007206  012777  160000  172012                 MOV     #160000,@RKBA   ;INTO THIS BUS ADDRESS

 007214  012777  000005  172000                 MOV     #5,@RKCS        ;READ, GO
 007222  104417                                 CON.RDY                 ;WAIT FOR CNTROL RDY

 007224  004737  020020                         JSR     PC,CHKCS        ;ANY ERROR IN RKCS?
 007230  104102                                 ERROR   102             ;ERROR BIT SET IN RKCS ON DOING
                                                                        ;A READ OF 4K WORDS INTO MEMORY
                                                                        ;(160000-177776)
 007232  012705  177764                         MOV     #-14,R5         ;REPORT 12 ERRORS AT MOST
 007236  012702  120000                         MOV     #120000,R2      ;STARTING ADDRESS OF BUFFER
                                                                        ;(PA=160000)
 007242  012701  000001                         MOV     #1,R1           ;INITIALIZE DATA PATTERN

 007246  020112                         6$:     CMP     R1,(R2)         ;CORRECT DATA WORD?
 007250  001413                                 BEQ     7$
 007252  010137  001162                         MOV     R1,$REG0        ;GET EXPCTD DATA WORD
 007256  011237  001164                         MOV     (R2),$REG1      ;GET DATA WORD RECVD
 007262  104107                                 ERROR   107             ;DATA ERROR ON DOING A READ OF
                                                                        ;4K WORDS STARTING FROM SEC 0,
                                                                        ;CYL 0, SUR 0 INTO MEMORY (160000-
                                                                        ;1777776)
 007264  104421  002214                         TYPMSG  ,MSG13          ;TYPE 'PHYSICAL BUS ADDRESS'
 007270  004737  017702                         JSR     PC,TYPDBO       ;TYPE OUT THE PHYSICAL BUS ADDRESS
                                                                        ;WHERE DATA
 007274  005205                                 INC     R5              ;ERROR OCCURRED.  R2 CONTAINS
 007276  001406                                 BEQ     8$              ;THE VIRTUAL ADDRESS

 007300  062702  000002                 7$:     ADD     #2,R2           ;POINTER TO NEXT BA
 007304  005201                                 INC     R1              ;NEXT PATTERN
 007306  022701  001000                         CMP     #1000,R1        ;ALL DONE?
 007312  001355                                 BNE     6$              ;NO

 007314  005037  177572                 8$:     CLR     @#SR0           ;TURN OFF MEM MANAGEMENT
```

```
                                ;;************************************************************
                                ;*TEST 7        PERFORM THE LARGEST POSSIBLE DATA TRANSFER
                                        ;THIS TEST PERFORMS THE LARGEST DATA TRANSFER POSSIBLE
                                        ;WITHIN AVAILABLE MEMORY.

                                        ;1) IF KT11 IS PRESENT A WRITE OF 16K WORDS IS DONE ON
                                        ;THE DISK (PROVIDED 28K OR MORE IS PRESENT).

                                        ;2) IF KT11 IS NOT PRESENT THEN ALL BUT TOP 1.5K OF MEMORY WILL BE USED
                                        ;FOR WRITING (RKDP MONITOR).

                                        ;ALL DATA TRANSFERS ARE DONE STARTING AT BA 'PGEND'.
                                        ;ALL WRITES ARE DONE BEGINNING AT SECTOR 0, CYLINDER 0,
                                        ;SURFACE 0.

                                        ;AFTER DOING THE 'WRITE' A 'WRITE CHECK' IS DONE
                                        ;TO SEE IF THE WRITE WAS DONE CORRECTLY.  ANY WRITE
                                        ;CHECK ERROR IS REPORTED.
                                ;;************************************************************
007320  000004                  TST7:   SCOPE
007322  005737  017432                  TST     $KT11           ;MEMORY MANAGEMENT PRESENT?
007326  100004                          BPL     1$              ;NO
                                                                ;YES
007330  023727  017700  001540          CMP     $LSTBK,#1540    ;28K OR MORE?
007336  002034                          BGE     XFR16K          ;YES

007340  013703  002054          1$:     MOV     MAXBA,R3        ;FIGURE OUT THE MAXIMUM
007344  162703  032412                  SUB     #PGEND,R3       ;XFER THAT CAN BE DONE
007350  000241                          CLC
007352  006003                          ROR     R3              ;FROM 'PGEND' TO THE
                                                                ;'MAXBA'
007354  005001                          CLR     R1
007356  010346                          MOV     R3,-(SP)        ;PUT DIVIDEND ON STACK (LO)
007360  005046                          CLR     -(SP)           ;(HIGH PART)
007362  012746  000400                  MOV     #400,-(SP)      ;DIVISOR
007366  004737  025120                  JSR     PC,@#$DIV       ;GO TO DIVIDE ROUTINE
007372  005726                          TST     (SP)+           ;POP OFF REMAINDER FROM STACK
007374  001401                          BEQ     2$
007376  005201                          INC     R1              ;GET # OF SECTORS REQUIRED TO
                                                                ;DO WRITE OF # OF WORDS
007400  062601                  2$:     ADD     (SP)+,R1        ;(CONTAINED IN R3). R1 CONTAINS
                                                                ;# OF SECTORS
007402  005002                          CLR     R2              ;FORM THE EXPECTED DISK
007404  162701  000014          3$:     SUB     #14,R1          ;ADDRESS - AFTER THE WRITE
007410  100403                          BMI     4$              ;IS DONE.
007412  062702  000020                  ADD     #20,R2
007416  000772                          BR      3$
007420  062701  000014          4$:     ADD     #14,R1          ;R2 CONTAINS EXPCTD RKDA
007424  050102                          BIS     R1,R2           ;AFTER WRITE IS DONE
007426  000404                          BR      XFR

007430  012703  040000          XFR16K: MOV     #40000,R3       ;# OF WORDS = 16K
007434  012702  000124                  MOV     #124,R2         ;RKDA SHOULD INCREMENT TO
                                                                ;THIS AFTER A TRANSFER OF 16K
                                                                ;WORDS STARTING AT DISK
                                                                ;ADDRESS = 0 (CYL 2,SUR 1,SEC 4)
```

```
007440  053702  001502          XFR:    BIS     QDRV,R2         ;ADD THE DRIVE # TO
                                                                ;EXPCTD RKDA AFTER XFER
007444  012737  007452  001110          MOV     #1$,$LPERR      ;LUP BAK TO '1$' ON ERROR
                                                                ;(SW 9)
007452  104416                  1$:     CON.RESET
007454  104420                          DRV.RESET
007456  013777  001502  171544          MOV     QDRV,@RKDA      ;ADDRESS THE DRIVE, CYL 0,SUR 0,
                                                                ;SEC 0
007464  010377  171534                  MOV     R3,@RKWC
007470  005477  171530                  NEG     @RKWC           ;WORD COUNT (IF MORE THAN
                                                                ;28K IS AVAILABLE A TRANSFER
                                                                ;OF 20K WILL BE DONE, IF
                                                                ;LESS THAN 28K, THE
                                                                ;LARGEST TRANSFER WITHIN THE
                                                                ;AVAILABLE MEMORY WILL
                                                                ;BE DONE, IN BOTH
                                                                ;CASES, DATA TRANSFER
                                                                ;WILL BE DONE STARTING
                                                                ;AT 'PGEND'
007474  012777  032412  171524          MOV     #PGEND,@RKBA    ;START WRITE FROM HERE
007502  012777  000003  171512          MOV     #3,@RKCS        ;WRITE, GO

007510  104417                          CON.RDY                 ;WAIT FOR CONTROL READY

007512  004737  020020                  JSR     PC,CHKCS        ;CHCK RKCS FOR ERROR?
007516  104102                          ERROR   102             ;ERROR BIT SET IN RKCS ON
                                                                ;DOING A LARGE 'WRITE'

007520  004737  020034                  JSR     PC,CHKDA        ;CHECK IF RKDA INCREMENTED CORRECTLY?
007524  104103                          ERROR   103             ;RKDA DID NOT INCREMENT
                                                                ;CORRECTLY

007526  012704  032412                  MOV     #PGEND,R4
007532  004737  020056                  JSR     PC,CHKBA        ;CHECK THAT RKBA INCREMENTED
                                                                ;CORRECTLY?
007536  104104                          ERROR   104             ;RKBA DID NOT INCREMENT
                                                                ;CORRECTLY

007540  004737  020136                  JSR     PC,CHKWC        ;CHECK THAT RKWC OVERFLOWED
007544  104105                          ERROR   105             ;RKWC DID NOT OVERFLOW TO
                                                                ;ZERO AFTER A WRITE

007546  012737  007554  001110          MOV     #2$,$LPERR      ;LUP BAK TO '2$' ON EROR (SW 9)
007554  104416                  2$:     CON.RESET
007556  104420                          DRV.RESET
007560  013777  001502  171442  3$:     MOV     QDRV,@RKDA      ;ADDRESS THE DRIVE, CYL 0,SEC 0,SUR 0
007566  010377  171432                  MOV     R3,@RKWC
007572  005477  171426                  NEG     @RKWC
007576  012777  032412  171422          MOV     #PGEND,@RKBA    ;STARTING BA

007604  012777  000407  171410  4$:     MOV     #407,@RKCS      ;WRITE CHECK, SSE, GO

007612  104417                          CON.RDY                 ;WAIT FOR CONTROL RDY

007614  004737  020020                  JSR     PC,CHKCS        ;ANY ERROR IN RKCS
```

```
007620  104102                              ERROR   102

007622  032777  040000  171372              BIT     #BIT14,@RKCS    ;SKIP CHECKING FOR WCE IF HE SET
007630  001030                              BNE     7$
007632  032777  000001  171360      5$:     BIT     #WCE,@RKER      ;WRITE CHECK ERROR?
007640  001406                              BEQ     6$              ;NO

007642  004737  021740                      JSR     PC,GT4RG        ;GET RKCS,ER,DS,DA
007646  017737  171354  001166              MOV     @RKBA,$REG2
007654  104110                              ERROR   110             ;WRITE CHECK ERROR. RKDA GIVES THE DISK ADDRESS
                                                                    ;WHERE WCE OCCURRED. (NOTE THE SECTOR IN
                                                                    ;ERROR IS OBTAINED BY BACKING OFF 1 SECTOR).
                                                                    ;NOTE THAT THE DATA BUFFER WHICH WAS WRITE
                                                                    ;CHECKED IS NOT ON A SECTOR BOUNDARY
                                                                    ;(LIKE 256, 512 WORD ETC.), BUT IS SOME
                                                                    ;SECTORS & A FRACTION (LIKE 300 WORDS ETC.)

007656  005777  171342              6$:     TST     @RKWC           ;WAS THE ENTIRE DATA BUFFER
007662  001413                              BEQ     7$              ;WRITE CHECKED?

007664  017705  171334                      MOV     @RKWC,R5
007670  042705  177760                      BIC     #177760,R5      ;FORM THE RKBA AND RKWC
007674  006305                              ASL     R5
007676  160577  171324                      SUB     R5,@RKBA        ;TO BE USED FOR DOING
                                                                    ;WRITE-CHECK IF THE REST
007702  042777  000017  171314              BIC     #17,@RKWC       ;OF THE DATA BUFFER

007710  000735                              BR      4$              ;GO DO WRT-CHK FOR
                                                                    ;REST OF THE BUFFER

                                    ;GO CHECK THE REST OF THE DRIVES.

007712  000137  005312              7$:     JMP     NXTDRV
```

```
                                    .SBTTL  EXERCISER PROGRAM

                                    ;BEGINING OF THE EXERCISER PART OF THE PROGRAM.
                                    ;IF THE PROGRAM WAS RESTARTED AT 210, THEN THE STATISTICS COLLECTED
                                    ;SO FAR WILL NOT BE CLEARED.


007716  104416                      EXRCSR: CON.RESET
007720  012700  001306                      MOV     #KEY,R0         ;CLEAR UP THE LOCATIONS FROM
007724  005020                      1$:     CLR     (R0)+           ;'KEY' TO 'KWHR' (EXCLUDED)
007726  020027  001552                      CMP     R0,#KWHR
007732  001374                              BNE     1$
007734  105737  001253                      TSTB    FRSTRT          ;RESTARTED AT 210?
007740  001045                              BNE     3$              ;YES, SAVE THE STATISTICS COLLECTED
                                                                    ;UPTILL NOW, DONT CLEAR THEM

007742  005020                      2$:     CLR     (R0)+           ;CLEAR LOCATIONS FROM 'HECN'
007744  020027  002032                      CMP     R0,#PCMND       ;TO 'PCMND' (EXCLUDED)
007750  001374                              BNE     2$

007752  012700  001554                      MOV     #KWMIN,R0       ;INITIALIZE COUNTS FOR MINS,
007756  012701  177704                      MOV     #-60.,R1        ;SECS, KW11
007762  010120                              MOV     R1,(R0)+
007764  010120                              MOV     R1,(R0)+
007766  010120                              MOV     R1,(R0)+

007770  012700  025636                      MOV     #RSDRVL,R0      ;INITIALIZE PTR TO RANDOM NO. SEEDS
007774  012720  123456                      MOV     #123456,(R0)+   ;USED BY THE RANDOM NUMBER GENERATOR
010000  012720  176543                      MOV     #176543,(R0)+   ;SET UP RANDOM NUMBER SEEDS TO BE
010004  012720  001201                      MOV     #1201,(R0)+
010010  012720  062465                      MOV     #62465,(R0)+
010014  012720  176105                      MOV     #176105,(R0)+
010020  012720  174532                      MOV     #174532,(R0)+
010024  012720  157650                      MOV     #157650,(R0)+
010030  012720  030753                      MOV     #30753,(R0)+
010034  012720  131547                      MOV     #131547,(R0)+
010040  012720  032070                      MOV     #32070,(R0)+
010044  012720  123456                      MOV     #123456,(R0)+
010050  012720  176543                      MOV     #176543,(R0)+


010054  013737  001236  002056      3$:     MOV     PCNTR,REPCNT    ;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.

010062  004737  022564                      JSR     PC,CHDPRS       ;CHECK IF ANY DRIVES ARE PRESENT
                                                                    ;IF NONE, GO TO $EOP, END OF PASS




010066  012746  177777                      MOV     #177777,-(SP)   ;PUT LOW DIVIDEND ON STACK
010072  005046                              CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
                                                                    ;IT ON STACK
010074  013746  001264                      MOV     DRVPRS,-(SP)    ;PUSH DIVISOR ON STACK
```

```
010100  004737  025120                JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
010104  005726                        TST     (SP)+           ;DISCARD THE REMAINDER. QUOTIENT IS
                                                              ;NOW ON TOP OF THE STACK
010106  012637  001520                MOV     (SP)+,DRMAP     ;GET MAPPING FACTOR FOR DRIVES
010112  012737  000504  001522        MOV     #504,CYLMAP     ;GET MAPPING FACTOR FOR CYLINDERS
010120  012737  012527  001524        MOV     #5463.,SECMAP   ;GET MAPPING FACTOR FOR SECTORS
010126  012737  031463  001526        MOV     #13107.,FNMAP   ;GET MAPPING FACTOR FOR FUNCTION

010134  013700  002054                MOV     MAXBA,R0        ;COMPUTE THE MAXIMUM ALLOWABLE
010140  163700  002052                SUB     BASEBA,R0       ;WORD COUNT FOR DATA TRANSFERS
010144  000241                        CLC
010146  006000                        ROR     R0              ;CONVERT TO WORDS

010150  012746  177777                MOV     #177777,-(SP)   ;PUT LOW DIVIDEND ON STACK
010154  005046                        CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
                                                              ;IT ON STACK
010156  010046                        MOV     R0,-(SP)        ;PUSH DIVISOR ON STACK
010160  004737  025120                JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
010164  005726                        TST     (SP)+           ;DISCARD THE REMAINDER. QUOTIENT IS
                                                              ;NOW ON TOP OF THE STACK
010166  005216                        INC     (SP)
010170  012637  001530                MOV     (SP)+,BAMAP     ;SAVE THE MAPPING FACTOR FOR BUS ADDRESS
```

```
                                      ;THE ENTIRE DISK (ALL DRIVES) IS WRITTEN WITH RANDOM PATTERNS. THE FIRST
                                      ;WORD OF EACH SECTOR GIVES THE NUMBER OF WORDS (2'S COMPLEMENT) WRITTEN IN
                                      ;THAT SECTORS. REST OF THE DATA WORDS FOR THE SECTOR ARE GENERATED USING
                                      ;THE ABSOLUTE DISK ADDRESS AS THE RANDOM SEED.
                                      ;IF THE PROGRAM WAS RESTARTED AT 210 THEN CHECK IF SW 4 IS SET. IF IT IS
                                      ;THEN DO NOT REWRITE THE DISK WITH RANDOM PATTERNS. IF SW 4 IS NOT SET THEN
                                      ;ALL THE DISKS (PRESENT) ARE WRITTEN WITH RANDOM PATTERNS.

010174  105737  001253          WRDSK:  TSTB    FRSTRT          ;RESTARTED AT 210?
010200  001407                          BEQ     1$              ;NO
010202  105037  001253                  CLRB    FRSTRT          ;YES, CLEAR THE RESTART FLAG
010206  032777  000020  170724          BIT     #SW4,@SWR       ;SW 4 SET?
010214  001401                          BEQ     1$              ;NO
010216  000540                          BR      BEGEX1          ;YES, DONT REWRITE ALL DISKS WITH
                                                                ;RANDOM PATTERNS
010220  012737  010236  001110  1$:     MOV     #2$,$LPERR      ;LUP TO '2$' ON EROR, SW 9 SET!
010226  012702  001254                  MOV     #PDR,R2         ;POINTER TO DRIVE #'S TABLE
010232  013703  001264                  MOV     DRVPRS,R3       ;# OF DRIVES PRESENT
010236  012700  011410          2$:     MOV     #4872.,R0       ;NUMBER OF SECTORS PER DISK
010242  112201                          MOVB    (R2)+,R1        ;GET THE FIRST AVAILABLE DRIVE
010244  042701  177770                  BIC     #177770,R1      ;POSITION THE BITS (15,14,13)
010250  010137  001502                  MOV     R1,QDRV
010254  000241                          CLC
010256  006001                          ROR     R1
010260  006001                          ROR     R1
010262  006001                          ROR     R1
010264  006001                          ROR     R1
010266  010177  170736                  MOV     R1,@RKDA        ;BASE DISK ADDRESS

010272  013704  002054                  MOV     MAXBA,R4        ;CALCULATE MAXIMUM BUFFER
010276  163704  002052                  SUB     BASEBA,R4
010302  006204                          ASR     R4              ;CONVERT TO WORDS
010304  042704  000377                  BIC     #377,R4         ;KEEP ONLY WHOLE BLOCKS
010310  000304                          SWAB    R4
010312  020427  000014                  CMP     R4,#12.         ;MAX OF 12 SECTORS
010316  003402                          BLE     3$
010320  012704  000014                  MOV     #12.,R4

010324  010401                  3$:     MOV     R4,R1
010326  020400                          CMP     R4,R0
010330  003401                          BLE     4$
010332  010001                          MOV     R0,R1
010334  000301                  4$:     SWAB    R1
010336  005401                          NEG     R1
010340  010137  010354                  MOV     R1,5$
010344  010177  170654                  MOV     R1,@RKWC
010350  004537  016612                  JSR     R5,GENBUF       ;GENERATE RANDOM DATA BUFER
010354  000000                  5$:     .WORD   0
010356  032412                          .WORD   DBUF            ;STARTING ADDRESS OF DATA BUFER

010360  012777  032412  170640          MOV     #DBUF,@RKBA     ;FROM THIS BUS ADDRESS
010366  012777  000003  170626          MOV     #3,@RKCS        ;WRITE, GO

010374  104417                          CON.RDY                 ;WAIT FOR CONTROL RDY
010376  004737  020020                  JSR     PC,CHKCS        ;ANY ERROR?
010402  104001                          ERROR   1               ;AN ERROR OCCURED WHILE DOING WRITE
```

```
                                                                            ;ON THE DISK, YOU ARE ADVISED TO USE
                                                                            ;BASIC AND DYNAMIC TESTS.

 010404  032777  000400  170526            BIT     #BIT8,@SWR              ;TYPE CURRENT STATUS
 010412  001435                            BEQ     6$
 010414  032700  000377                    BIT     #377,R0
 010420  001032                            BNE     6$
 010422  104401  010430                    TYPE    ,65$                    ;;TYPE ASCIZ STRING
 010426  000421                            BR      64$                     ;;GET OVER THE ASCIZ
                                   ;;65$:  .ASCIZ  <15><12>/WRITTING RANDOM PATTERN, DRIVE /
 010472                            64$:
 010472  013746  001502                    MOV     QDRV,-(SP)
 010476  104403                            TYPOS
 010500     001                            .BYTE   1
 010501     000                            .BYTE   0

 010502  104401  001213                    TYPE    ,$CRLF
 010506  104407                    6$:     CKSWR                           ;CHECK FOR SOFTWARE SWR CHANGE
 010510  160400                            SUB     R4,R0                   ;DECREMENT SECTOR COUNT
 010512  003304                            BGT     3$

 010514  005303                            DEC     R3                      ;MORE DRIVES TO DO?
 010516  001247                            BNE     2$

                                   ;IF SW 3 IS SET, ENABLE THE LINE CLOCK AND INITIALIZE COUNTS.

 010520  032777  000010  170412    BEGEX1: BIT     #SW3,@SWR               ;KW11L CLOCK PRESENT?
 010526  001403                            BEQ     BEGNEX                  ;IF NOT, SKIP. NOTE:IF KW11L IS
                                                                           ;NOT PRESENT SW3 SHOULD NOT BE SET
                                                                           ;OTHERWISE, A TIMEOUT WILL OCCUR.
 010530  052777  000100  170476            BIS     #BIT6,@KWLS             ;ENABLE KW11L CLOCK
                                                                           ;SERVICED AT PRIORITY 7 (KWSRVE)
```

```
                                   ;THE PROGRAM IS GOING TO LOOP BACK TO THIS POINT AT THE END OF A PASS.

 010536  104416                    BEGNEX: CON,RESET                       ;CLEAR EVERYTHING IN DRIVES
 010540  012706  001100                    MOV     #STACK,SP               ;RESET STACK
 010544  005737  001264                    TST     DRVPRS                  ;ANY DRIVES LEFT IN SYSTEM?
 010550  001402                            BEQ     QMNGER
 010552  004737  014406                    JSR     PC,GEN8RQ               ;GO GENERATE 8 COMMANDS FOR THE Q



                                   ;CHECK IF THERE IS ANY UNFINISHED COMMAND IN THE Q, WAITING TO BE EXECUTED,
                                   ;IF THERE IS NONE, GO TO THE 'GEN8RQ' AND GENERATE 8 REQUESTS (COMMANDS),
                                   ;AND PUT THEM IN THE Q. IF THERE IS AN UNFINISHED COMMAND, FIND OUT IF
                                   ;THERE IS A PRIORITY COMMAND TO BE EXECUTED IMMEDIATELY.

 010556  013746  001244            QMNGER: MOV     PPRLVL,-(SP)            ;NEW PSW , RAISE PRIORITY
 010562  012746  010570                    MOV     #1$,-(SP)               ;RETURN PC ***************
 010566  000002                            RTI

 010570  005737  001264            1$:     TST     DRVPRS                  ;ANY DRIVES IN SYSTEM?
 010574  001040                            BNE     2$
 010576  104401  010604                    TYPE    ,65$                    ;;TYPE ASCIZ STRING
 010602  000432                            BR      64$                     ;;GET OVER THE ASCIZ
                                   ;;65$:  .ASCIZ  <15><12>/HALTING - PRESS CONTINUE TO RESTART AT '200'/<15><12><12><12>
 010670                            64$:
 010670  000000                            HALT
 010672  000137  003376                    JMP     START

 010676  012700  001306            2$:     MOV     #KEY,R0
 010702  005720                    3$:     TST     (R0)+                   ;ANY UNFINISHED COMMAND
 010704  100006                            BPL     4$                      ;IN Q
 010706  020027  001326                    CMP     R0,#KEY+20
 010712  001373                            BNE     3$
 010714  004737  014406                    JSR     PC,GEN8RQ               ;IF NOT, GO GENERATE 8 MORE COMMANDS
 010720  000460                            BR      CHFAFN


 010722  012700  001306            4$:     MOV     #KEY,R0
 010726  032710  010000            5$:     BIT     #BIT12,(R0)             ;ANY HIGH PRIORITY COMMAND IN Q
 010732  001404                            BEQ     6$
 010734  005710                            TST     (R0)                    ;FINISHED?
 010736  100402                            BMI     6$                      ;YES
 010740  105710                            TSTB    (R0)                    ;IN EXECUTION?
 010742  100165                            BPL     PRICMND                 ;NO, GO PROCESS HIGH PRIORITY

 010744  005720                    6$:     TST     (R0)+                   ;CHECK THE ENTIRE Q
 010746  020027  001326                    CMP     R0,#KEY+20
 010752  001365                            BNE     5$


                                   ;FIND OUT IF THERE IS A WRITE CHECK FUNCTION TO BE DONE IMMEDIATELY AFTER
                                   ;A WRITE, THAT WAS DONE PREVIOUSLY.

 010754  005737  001456                    TST     WCFLG                   ;IS WRITE CHECK TO FOLLOW
 010760  100040                            BPL     CHFAFN                  ;WRITE? IF NOT, BRANCH
                                                                           ;YES
```

```
Ø1Ø762  Ø137ØØ  ØØ1456                  MOV     WCFLG,RØ        ;GET WC FLAG
Ø1Ø766  Ø427ØØ  1774ØØ                  BIC     #1774ØØ,RØ      ;LOWER BYTE CONTAINS KEY#X2 (OFFSET FROM
                                                                ;KEY), OF THE WRITE FUNCTION
Ø1Ø772  Ø16ØØ1  ØØ2Ø32                  MOV     PCMND(RØ),R1    ;GET POINTER
Ø1Ø776  Ø627ØØ  ØØ13Ø6                  ADD     #KEY,RØ         ;FROM ADDRESS OF THE KEY
                                                                ;WHICH WAS USED FOR PREVIOUS
                                                                ;WRITE
Ø11ØØ2  Ø22761  ØØØØØ2  ØØØØØ2          CMP     #2,2(R1)        ;CHECK THAT THE FUNCTION
Ø11Ø1Ø  ØØ1413                          BEQ     7$              ;WAS A WRITE
Ø11Ø12  Ø11Ø37  ØØ1162                  MOV     (RØ),$REGØ      ;GET KEY CONTENTS
Ø11Ø16  Ø16137  ØØØØØ2  ØØ1164          MOV     2(R1),$REG1     ;GET THE FUNCTION INDICATED BY THIS KEY
Ø11Ø24  1Ø4Ø27                          ERROR   27              ;IF NOT, ERROR
                                                                ;BEFORE DOING A WRITE CHECK A WRITE IS
                                                                ;ALWAYS DONE. OCCURANCE OF THIS ERROR
                                                                ;INDICATES THAT SOMEHOW AN ATTEMPT IS BEING
                                                                ;MADE TO DO WRITE CHECK BEFORE A WRITE IS
                                                                ;PERFORMED. THE KEY IN ERROR MESSAGE WAS
                                                                ;THE ONE WHICH GAVE RISE TO THIS ATTEMPT.
                                                                ;THE FUNCTION CODE IS THE FUNCTION ASSOSCIATED
                                                                ;WITH THAT KEY
Ø11Ø26  ØØ5Ø37  ØØ1456                  CLR     WCFLG           ;ABORT THIS WRITE CHECK
Ø11Ø32  Ø5271Ø  1ØØØØØ                  BIS     #BIT15,(RØ)
Ø11Ø36  ØØØ647                          BR      QMNGER
Ø11Ø4Ø  Ø12761  ØØØØØ6  ØØØØØ2  7$:     MOV     #6,2(R1)        ;SET UP BITS FOR WRT CHK
                                                                ;NOW
Ø11Ø46  Ø4271Ø  17434Ø                  BIC     #17434Ø,(RØ)    ;CLEAR OUT THE UNNECESSARY
                                                                ;FLAGS FROM THE KEY
Ø11Ø52  Ø5271Ø  ØØØ1ØØ                  BIS     #BIT6,(RØ)      ;SET FLAG FOR WRITE CHECK, FOR
                                                                ;THIS KEY
Ø11Ø56  ØØØ137  Ø1152Ø                  JMP     EXCUT

                                                                ;NO HIGH PRIORITY COMMAND IN Q
```

```
                                ;NO HIGH PRIORITY COMMAND WAS FOUND IN THE Q. FIND OUT THE FIRST AVAILABLE
                                ;COMMAND IN THE Q. FOR EXECUTION.
Ø11Ø62  ØØ5ØØØ                  CHFAFN: CLR     RØ              ;WAIT FOR ANY IMMEDIATE INTERUPT
Ø11Ø64  ØØ5Ø46                          CLR     -(SP)           ;NEW PSW
Ø11Ø66  Ø12746  Ø11Ø74                  MOV     #1$,-(SP)       ;RETURN FOR RTI
Ø11Ø72  ØØØØØ2                          RTI


Ø11Ø74  1052ØØ                  1$:     INCB    RØ
Ø11Ø76  ØØ1376                          BNE     .-2
Ø111ØØ  Ø13746  ØØ1244                  MOV     PPRLVL,-(SP)    ;NEW PSW, LOCK OUT CPU
Ø111Ø4  Ø12746  Ø11112                  MOV     #2$,-(SP)       ;RETURN FOR RTI *************
Ø1111Ø  ØØØØØ2                          RTI


Ø11112  Ø127ØØ  ØØ13Ø6          2$:     MOV     #KEY,RØ
Ø11116  ØØ571Ø                  CH1:    TST     (RØ)            ;UNFINISHED COMMAND?
Ø1112Ø  1ØØ436                          BMI     CH2
Ø11122  1Ø571Ø                          TSTB    (RØ)            ;IN PROGRESS?
Ø11124  1ØØ434                          BMI     CH2
Ø11126  Ø11ØØ1                          MOV     (RØ),R1
Ø1113Ø  Ø427Ø1  17777Ø                  BIC     #17777Ø,R1
Ø11134  1Ø5761  ØØ1426                  TSTB    BUSY(R1)        ;IS THIS DRIVE BUSY?
Ø1114Ø  1ØØ426                          BMI     CH2
Ø11142  1Ø5761  ØØ1436                  TSTB    POS(R1)         ;HAS THIS DRIVE BEEN POSITIONED
                                                                ;FOR ANY OTHER COMMAND.  IF YES,
                                                                ;SKIP.  IF NOT, PROCEED
Ø11146  ØØ1Ø23                          BNE     CH2

                                                                ;CHECK IF THERE IS AY OTHER COMMAND
Ø1115Ø  Ø1ØØØ2                          MOV     RØ,R2           ;ON A DRIVE THAT IS NOT THE SAME
                                                                ;AS THE PREVIOUS DRIVE.  THIS COMMAND
Ø11152  ØØØ414                          BR      2$              ;SHOULD NOT BE IN PROGRESS
                                                                ;AND SHOULD NOT HAVE BEEN COMPLETED
Ø11154  ØØ5712                  1$:     TST     (R2)            ;UNFINISHED COMMAND?
Ø11156  1ØØ412                          BMI     2$
Ø1116Ø  1Ø5712                          TSTB    (R2)            ;IN PROGRESS?
Ø11162  1ØØ41Ø                          BMI     2$
Ø11164  Ø112Ø3                          MOV     (R2),R3
Ø11166  Ø427Ø3  17777Ø                  BIC     #17777Ø,R3
Ø11172  1Ø5763  ØØ1426                  TSTB    BUSY(R3)        ;IS THE DRIVE BUSY?
Ø11176  1ØØ4Ø2                          BMI     2$
Ø112ØØ  Ø2Ø3Ø1                          CMP     R3,R1           ;IS THIS COMMAND ON THE SAME DRIVE?
Ø112Ø2  ØØ1447                          BEQ     POSTION         ;IF YES, GO AND POSITION THE COMMAND
                                                                ;POINTED TO BY RØ, (BECAUSE THERE IS
                                                                ;ONE MORE COMMAND THAT CAN BE PERFORMED
                                                                ;ON THE SAME DRIVE)
Ø112Ø4  ØØ5722                  2$:     TST     (R2)+           ;CHECK REST OF THE Q
Ø112Ø6  Ø2Ø227  ØØ1326                  CMP     R2,#KEY+2Ø
Ø11212  ØØ136Ø                          BNE     1$
                                                                ;IF THERE WAS NO EXECUTABLE COMMAND
Ø11214  ØØØ47Ø                          BR      EXNSK           ;ON ANY OTHER DRIVE, THEN EXECUTE
                                                                ;COMMAND POINTED TO BY RØ

Ø11216  ØØ572Ø                  CH2:    TST     (RØ)+           ;CHECK ENTIRE Q
Ø1122Ø  Ø2ØØ27  ØØ1326                  CMP     RØ,#KEY+2Ø
```

```
011224  001334                            BNE     CH1


                                  ;NO (UNPOSITIONED) EXECUTABLE COMMAND WAS FOUND IN THE Q. CHECK IF THERE
                                  ;IS ANY POSITIONED COMMAND IN THE Q, WAITING TO BE EXECUTED.

011226  105777  167770                    TSTB    @RKCS           ;IF THE CONTROLLER IS BUSY GO TO
011232  100402                            BMI     1$              ;STATUS AND WAIT FOR INTERRUPTS
011234  000137  021344                    JMP     STATUS          ;IF THE CONTROLLER IS NOT BUSY FIND

011240  012700  001306            1$:     MOV     #KEY,R0         ;ANY POSITIONED COMMAND AND EXECUTE
011244  032710  000020            2$:     BIT     #BIT4,(R0)
011250  001414                            BEQ     3$              ;IT
011252  005710                            TST     (R0)            ;MAKE SURE COMMAND IS POSITIONED
011254  100412                            BMI     3$
011256  105710                            TSTB    (R0)            ;COMMAND IS UNFINISHED,
011260  100410                            BMI     3$              ;IT IS NOT IN PROGRESS,
011262  011001                            MOV     (R0),R1         ;THE DRIVE IS NOT IN BUSY
011264  042701  177770                    BIC     #177770,R1
011270  105761  001426                    TSTB    BUSY(R1)
011274  100402                            BMI     3$
011276  000137  011520                    JMP     EXCUT           ;GO EXECUTE THE COMMAND IF THE

011302  005720                    3$:     TST     (R0)+           ;ABOVE CONDITIONS ARE SATISFIED
011304  020027  001326                    CMP     R0,#KEY+20              ;CHECK ALL COMMANDS IN Q
011310  001355                            BNE     2$
011312  000137  021344                    JMP     STATUS



                                  ;ENTER HERE IF THERE IS A PRIORITY COMMAND TO BE EXECUTED.


011316  000137  011520            PRICMN: JMP     EXCUT           ;GO EXECUTE NON-SEEK COMMAND.

                                  ;ENTER THIS IF A COMMAND IS TO BE PREPOSITIONED (BEFORE EXECUTING A
                                  ;FUNCTION)

011322  032710  000020            POSTION: BIT    #BIT4,(R0)      ;ALREADY POSITIONED?
011326  001333                            BNE     CH2             ;YES, GO BACK AND CHECK IF REST OF THE
                                                                  ;COMMANDS IN THE Q ARE TO BE POSITIONED

011330  004737  012072                    JSR     PC,POSK         ;GO CHECK, & SET UP FLAGS
011334  004737  020256                    JSR     PC,POSCMND      ;SAVE INFO ABOUT PRESENT & PAST COMAND
011340  012777  000111  167654            MOV     #111,@RKCS      ;POSITION THE COMMAND
011346  005046                            CLR     -(SP)           ;NEW PSW, DROP PRIORITY
011350  012746  011356                    MOV     #1$,-(SP)       ;RETURN FOR RTI
011354  000002                            RTI
011356                            1$:
011356  105737  001535                    TSTB    INT1FL
011362  001775                            BEQ     .-4             ;WAIT FOR INTERRUPT
011364  012777  013164  167646            MOV     #INTHND,@RKVEC  ;RE-ESTABLISH RK INTERRUPT VECTOR
011372  000137  011062                    JMP     CHFAFN          ;GO BACK AND CHECK IF ANY COMMANDS TO BE
                                                                  ;POSITIONED OR EXECUTED


```

```
                                                                  ;IS THERE ANY POSITIONED COMMAND IN
                                                                  ;THE Q.  FIND OUT? IF THERE IS
                                                                  ;ONE EXECUTE THE POSITIONED COMMAND,
                                                                  ;BUT FIRST POSITION THE COMMAND (KEY)
                                                                  ;POINTED TO BY R0

011376  012701  001306            EXNSK:  MOV     #KEY,R1
011402  032711  000020            1$:     BIT     #BIT4,(R1)      ;HEADS POSITIONED?
011406  001412                            BEQ     2$
011410  005711                            TST     (R1)            ;FUNCTION COMPLETED?
011412  100410                            BMI     2$              ;YES, BRANCH
011414  105711                            TSTB    (R1)            ;FUNCTION IN PROGRESS?
011416  100406                            BMI     2$
011420  011102                            MOV     (R1),R2
011422  042702  177770                    BIC     #177770,R2
011426  105762  001426                    TSTB    BUSY(R2)        ;DRIVE BUSY?
011432  100005                            BPL     3$

011434  005721                    2$:     TST     (R1)+           ;CHECK ALL COMMANDS IN Q
011436  020127  001326                    CMP     R1,#KEY+20
011442  001357                            BNE     1$

011444  000425                            BR      EXCUT           ;NO POSITIONED COMMAND WAS
                                                                  ;FOUND IN THE Q. SO EXECUTE
                                                                  ;COMMAND POINTED TO BY R0


                                                                  ;AN ALREADY POSITIONED COMMAND HAS
                                                                  ;BEEN FOUND.
011446  010137  001536            3$:     MOV     R1,SAVKEY       ;SAVE POINTER TO THE COMMAND(KEY)
                                                                  ;WHICH IS ALREADY POSITIONED
011452  004737  012072                    JSR     PC,POSK         ;GO AND POSITION THE COMMAND(KEY)
                                                                  ;POINTED TO BY R0
011456  004737  020256                    JSR     PC,POSCMND      ;SAVE INFO ABOUT PAST & PRESENT COMAND
011462  012777  000111  167532            MOV     #111,@RKCS      ;SEEK, IDE, GO
011470  005046                            CLR     -(SP)           ;ALLOW FIRST INTERRUPT
011472  012746  011500                    MOV     #4$,-(SP)       ;RETURN FOR RTI **********
011476  000002                            RTI


011500  105737  001535            4$:     TSTB    INT1FL          ;DID INTERRUPT OCCUR?
011504  001775                            BEQ     4$              ;BR IF NOT
011506  012777  013164  167524            MOV     #INTHND,@RKVEC  ;REESTABLISH INTERRUPT ADDRESS
011514  013700  001536                    MOV     SAVKEY,R0       ;RESTORE THE SAVED POINTER TO KEY
```

```
                                                                ;EXECUTE THE COMMAND POINTED TO BY R0
                                                                ;R0 CONTAINS THE POINTER TO THE
                                                                ;COMMAND KEY
011520  011001                  EXCUT:  MOV     (R0),R1         ;GET DRIVE NO
011522  042701  177770                  BIC     #177770,R1
011526  005710                          TST     (R0)            ;THIS COMMAND UNFINISHED?
011530  100004                          BPL     1$
011532  011037  001162                  MOV     (R0),$REG0      ;GET KEY
011536  104030                          ERROR   30              ;IF NOT, ERROR
                                                                ;AN ATTEMPT WAS MADE TO REEXECUTE A COMMAND
                                                                ;THAT HAS ALREADY BEEN EXECUTED BEFORE.
                                                                ;(ON DRIVE NO. GIVEN IN ERROR MESSAGE)
011540  000512                          BR      ABRT1
011542  105710                  1$:     TSTB    (R0)            ;IS IT IN PROGRESS?
011544  100004                          BPL     3$
011546  011037  001162                  MOV     (R0),$REG0      ;GET KEY
011552  104030                          ERROR   30              ;IF YES, ERROR

011554  000504                          BR      ABRT1           ;AN ATTEMPT WAS MADE TO REEXECUTE A COMMAND
                                                                ;THAT IS IN PROGRESS (ON DRIVE NO. GIVEN
                                                                ;IN EROR MESAGE)

011556  105761  001426          3$:     TSTB    BUSY(R1)        ;IS THE DRIVE BUSY?
011562  100004                          BPL     4$
011564  010137  001162                  MOV     R1,$REG0        ;GET DRIVE #
011570  104002                          ERROR   2               ;'BUSY' FLAG FOR THE DRIVE (CONTAINED
011572  000470                          BR      ABRT            ;IN R1) IS SET, INDICATING THAT THE DRIVE
                                                                ;IS 'BUSY' AND ENGAGED IN AN ACTIVITY,
                                                                ;STILL AN ATTEMPT WAS MADE TO INITIATE
                                                                ;A FUNCTION ON THIS DRIVE.  THIS IS AN
                                                                ;UNEXCEPTED ERROR CONDITION.

011574  105777  167422          4$:     TSTB    @RKCS           ;CONTROL READY SET?
011600  100404                          BMI     5$              ;YES
011602  004737  021740                  JSR     PC,GT4RG        ;GET RKCS, ER, DS, DA
011606  104003                          ERROR   3               ;CONTROL READY IS NOT SET.  THIS IS AN
011610  000461                          BR      ABRT            ;UNEXPECTED ERROR CONDITION AT THIS
                                                                ;POINT OF EXECUTION, CONTROL SHOULD
                                                                ;BE NORMALLY READY.
011612  011002                  5$:     MOV     (R0),R2
011614  000302                          SWAB    R2
011616  042702  177770                  BIC     #177770,R2      ;FORM THE ADDRESS OF THE
011622  006302                          ASL     R2              ;PARAMETER LIST FOR THIS
011624  010204                          MOV     R2,R4           ;COMMAND
011626  016205  002032                  MOV     PCMND(R2),R5

011632  011577  167372                  MOV     (R5),@RKDA      ;GET THE FIRST ITEM FROM LIST
                                                                ;DISK ADDRES
011636  032777  000100  167352          BIT     #RWS,@RKDS      ;R/W/S RDY SET?
011644  001006                          BNE     6$              ;YES
011646  010137  001250                  MOV     R1,SRDRV        ;GET DRIVE #, FOR TYPING SERIAL #
011652  004737  021740                  JSR     PC,GT4RG        ;GET RKCS, ER, DS, DA
011656  104004                          ERROR   4               ;R/W/S RDY IS NOT SET FOR THE DRIVE.
011660  000435                          BR      ABRT            ;A (NON-SEEK) FUNCTION WAS SUPPOSED
                                                                ;TO BE EXECUTED ON THIS DRIVE.  THIS IS
                                                                ;AN UNEXPECTED ERROR CONDITIONS AT THIS
```

```
                                                                ;POINT OF EXECUTION R/W/S RDY SHOULD
                                                                ;BE NORMALLY SET.
011662  016503  000002          6$:     MOV     2(R5),R3        ;GET FUNCTION TO BE DONE

011666  122703  000002                  CMPB    #2,R3           ;IS IT A WRITE FUNCTION?
011672  001437                          BEQ     WRFNC           ;YES, IT IS WRITE
                                                                ;IT'S NOT A WRITE FUNCTION

011674  016503  000002          NWRFNC: MOV     2(R5),R3        ;GET FUNCTION TO BE DONE
011700  052703  000501                  BIS     #501,R3         ;SET SSE,IDE,GO BITS
011704  016577  000004  167312          MOV     4(R5),@RKWC     ;GET WORD COUNT
011712  016577  000006  167306          MOV     6(R5),@RKBA     ;GET BUS ADDRESS

011720  004737  020300                  JSR     PC,FNCMND       ;SAVE INFO ABOUT PAST & PRESENT COMAND

011724  010377  167272                  MOV     R3,@RKCS        ;SET SSE,IDE, FUNCTION, GO

011730  052710  000200                  BIS     #BIT7,(R0)      ;SET FLAG INDICATING FUNCTION
011734  052704  000200                  BIS     #BIT7,R4        ;IN PROGRESS
                                                                ;R4 CONTAINS OFFSET TO THE COMMAND KEY
                                                                ;(FROM 'KEY') BITS 0-3, BIT 7 IS SET
011740  110461  001426                  MOVB    R4,BUSY(R1)     ;SET FLAG INDICATING DRIVE BUSY
011744  110437  001534                  MOVB    R4,INTFLG       ;SET FLAG FOR INTERUPT USE
011750  000137  021344                  JMP     STATUS

011754  004737  014334          ABRT:   JSR     PC,DRVABT       ;ABORT THE FUNCTION
011760  004737  016446                  JSR     PC,CHKDRV       ;GO CHECK, DRIVE ERRORS
011764  000240                          NOP
011766  000137  010556          ABRT1:  JMP     QMNGER
```

```
                                                                     ;THE FUNCTION TO BE EXECUTED IS A
                                                                     ;WRITE FUNCTION
011772  016537  000004  012012  WRFNC:  MOV     4(R5),1$             ;GET # OF WORDS TO BE WRITTEN
012000  016537  000006  012014          MOV     6(R5),2$             ;GET STARTING BA OF BUFFER
012006  004537  016612                  JSR     R5,GENBUF            ;GO GENERATE BUFFER
012012  000000                  1$:     .WORD   0                    ;SAVE # OF WORDS
012014  000000                  2$:     .WORD   0                    ;SAVE STARTING BUS ADDRESS OF BUFFER
012016  052703  000101                  BIS     #101,R3              ;SET IDE AND GO BIT
012022  013777  012012  167174          MOV     1$,@RKWC             ;SET WORD COUNT
012030  013777  012014  167170          MOV     2$,@RKBA             ;SET BUS ADDRESS
012036  004737  020300                  JSR     PC,FNCMND            ;SAVE INFO ABOUT PAST & PRESENT COMAND

012042  010377  167154                  MOV     R3,@RKCS             ;ISSUE WRITE,IDE,GO
012046  052710  000200                  BIS     #BIT7,(R0)           ;SET FLAG INDICATING FUNCTION IN
012052  052704  000200                  BIS     #BIT7,R4             ;PROGRESS

012056  110461  001426                  MOVB    R4,BUSY(R1)          ;SET FLAG INDICATING DRIVE BUSY
012062  110437  001534                  MOVB    R4,INTFLG            ;SET FLAG FOR INTERUPT USE
                                                                     ;R4 CONTAINS OFFSET TO THE COMMAND KEY
                                                                     ;(FROM 'KEY') BITS 0-3, BIT 7 IS SET
012066  000137  021344                  JMP     STATUS
```

```
012072  011001                  POSK:   MOV     (R0),R1              ;INITIAL CHECKING PRIOR TO DOING
012074  042701  177770                  BIC     #177770,R1           ;POSITIONING COMMAND POINTED TO BY R0
012100  105761  001426                  TSTB    BUSY(R1)             ;DRIVE BUSY?
012104  100004                          BPL     1$
012106  010137  001162                  MOV     R1,$REG0             ;GET DRIVE # FOR WHICH 'BUSY' IS SET
012112  104002                          ERROR   2                    ;'BUSY' FLAG FOR THE DRIVE (CONTAINED
012114  000470                          BR      7$                   ;IN R1) IS SET, INDICATING THAT THE DRIVE
                                                                     ;IS 'BUSY' AND ENGAGED IN AN ACTIVITY,
                                                                     ;STILL AN ATTEMPT WAS MADE TO INITIATE
                                                                     ;POSITIONINGN ON THIS DRIVE.  THIS IS AN
                                                                     ;UNEXCEPTED ERROR CONDITION.
012116  105777  167100          1$:     TSTB    @RKCS                ;CONTROL READY SET?
012122  100404                          BMI     2$                   ;YES
012124  004737  021740                  JSR     PC,GT4RG             ;GET RKCS, ER, DS, DA
012130  104003                          ERROR   3                    ;CONTROL READY IS NOT SET.  THIS IS AN
012132  000454                          BR      6$                   ;UNEXPECTED ERROR CONDITION AT THIS
                                                                     ;POINT OF EXECUTION, CONTROL SHOULD
                                                                     ;BE NORMALLY READY.
012134  011002                  2$:     MOV     (R0),R2
012136  000302                          SWAB    R2
012140  042702  177770                  BIC     #177770,R2
012144  006302                          ASL     R2
012146  016203  002032                  MOV     PCMND(R2),R3         ;STARTING WORD OF COMMAND TABLE
012152  011377  167052                  MOV     (R3),@RKDA
012156  032777  000100  167032          BIT     #RWS,@RKDS           ;R/W/S RDY SET?
012164  001006                          BNE     3$                   ;YES
012166  010137  001250                  MOV     R1,SRDRV             ;GET DRIVE #, FOR TYPING SERIAL #
012172  004737  021740                  JSR     PC,GT4RG             ;GET RKCS, ER, DS, DA
012176  104004                          ERROR   4                    ;R/W/S RDY IS NOT SET FOR THE DRIVE.  A
012200  000431                          BR      6$                   ;(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.
012202  110261  001426          3$:     MOVB    R2,BUSY(R1)          ;SET FLAG INDICATING DRIVE BUSY
012206  152761  000200  001426          BISB    #BIT7,BUSY(R1)
012214  032710  000010                  BIT     #BIT3,(R0)           ;IS THIS A SEEK COMMAND
012220  001006                          BNE     4$
012222  112761  000377  001436          MOVB    #377,POS(R1)         ;SET FLAG INDICATING, THIS DRIVE POSITIONS
012230  052710  000020                  BIS     #BIT4,(R0)           ;SET FLAG INDICATING THIS COMMAND
012234  000402                          BR      5$
012236  052710  000200          4$:     BIS     #BIT7,(R0)           ;POSITIONED.  SET FLAG INDICATING
                                                                     ;FUNCTION IN PROGRESS
012242  012777  012304  166770  5$:     MOV     #INT1SK,@RKVEC       ;SET UP RK11 VECTOR
012250  013777  001244  166764          MOV     PPRLVL,@RKSTAT       ;PSW ON INTERRUPT
012256  105037  001535                  CLRB    INT1FL               ;CLEAR FLAG INDICATING - INTERRUPT
012262  000207                          RTS     PC                   ;OCCURRED

012264  004737  014334          6$:     JSR     PC,DRVABT            ;ABORT THE FUNCTION
012270  004737  016446                  JSR     PC,CHKDRV            ;GO CHECK, DRIVE ERRORS
012274  000240                          NOP
012276  005726                  7$:     TST     (SP)+                ;POP THE RETURN ADDRESS
012300  000137  010556                  JMP     QMNGER
```

```
                                          ;AFTER ISSUING A SEEK FOR POSITIONING, AN INTERRUPT IS ALLLOWED TO TAKE
                                          ;*PLACE. THIS HANDLER SERVICES THE FIRST INTERRUPT, WHICH OCCURS AS A RESULT
                                          ;OF THE SEEK BEING ACCEPTED.

012304  105237  001535            INT1SK: INCB    INT1FL          ;INDICATE THAT INTERRUPT TOOK PLACE
012310  053766  001244  000002            BIS     PPRLVL,2(SP)    ;CPU PRIORITY TO 5 ON RETURN FROM
                                                                  ;INTERRUPT

012316  004737  020200                    JSR     PC,CHKCRDY      ;CONTROL READY SET?
012322  104005                            ERROR   5               ;CONTROL READY NOT SET AFTER THE FIRST
                                                                  ;INTERRUPT ON INITIATING SEEK

012324  032777  020000  166670            BIT     #SCP,@RKCS      ;SCP BIT SET?
012332  001403                            BEQ     1$
012334  004737  022130                    JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
                                                                  ;TYPING SERIAL DRIVE #
012340  104012                            ERROR   12              ;SCP SET AFTER FIRST INTERRUPT ON
                                                                  ;ACCEPTING SEEK.  A SEEK WAS INITIATED,
                                                                  ;AS A RESULT OF WHICH (THIS) FIRST
                                                                  ;INTERRUPT OCCURRED, BUT SCP SHOULD
                                                                  ;NOT BE SET AFTER THE FIRST INTERRUPT.
                                                                  ;(IT GETS SET ONLY AFTER THE SEEK
                                                                  ;IS DONE).
                                                                  ;THIS IS THE FIRST INTERRUPT
                                                                  ;AFTER ISSUING SEEK
012342  017700  166654            1$:     MOV     @RKCS,R0
012346  042700  177760                    BIC     #177760,R0      ;CHECK THERE IS A SEEK FUNCTION
012352  022700  000010                    CMP     #10,R0          ;IN RKCS
012356  001403                            BEQ     2$
012360  004737  021740                    JSR     PC,GT4RG        ;GET RKCS,ER,DS,DA
012364  104006                            ERROR   6               ;RKCS DOES NOT CONTAIN 'SEEK' FUNCTION.
                                                                  ;A SEEK WAS INITIATED AS A RESULT OF WHICH
                                                                  ;(THIS) FIRST INTERRUPT OCCURRED.  RKCS
                                                                  ;SHOULD CONTAIN THE SEEK FUNCTION BITS,
                                                                  ;BUT IT DID NOT

012366  017700  166636            2$:     MOV     @RKDA,R0        ;GET THE DRIVE # FROM RKDA
012372  042700  017777                    BIC     #17777,R0
012376  000241                            CLC
012400  006100                            ROL     R0
012402  006100                            ROL     R0
012404  006100                            ROL     R0
012406  006100                            ROL     R0
012410  010001                            MOV     R0,R1
012412  105761  001426                    TSTB    BUSY(R1)        ;CHECK THIS DRIVE NO. WAS BUSY
012416  100403                            BMI     3$              ;OK, IF IT WAS
012420  010137  001162                    MOV     R1,$REG0        ;GET DRIVE #(FROM RKDA)
012424  104007                            ERROR   7               ;'BUSY' FLAG FOR THE DRIVE WAS NOT SET.
                                                                  ;THE DRIVE # (IN RKDA) WHICH CAUSED (?)
                                                                  ;THIS (FIRST) INTERRUPT SHOULD HAVE BEEN
                                                                  ;'BUSY' (SOFTWARE FLAG).  NOTE WHENEVER
                                                                  ;ANY OPERATION IS INITIATED ON ANY DRIVE
                                                                  ;'BUSY' FLAG FOR THAT DRIVE IS SET.
012426  116100  001426            3$:     MOVB    BUSY(R1),R0
012432  042700  177600                    BIC     #177600,R0      ;FORM THE ADDRESS OF
012436  062700  001306                    ADD     #KEY,R0         ;THIS KEY
```

```

012442  032710  000020                    BIT     #BIT4,(R0)      ;IS THE KEY ACTIVE FOR 'POSITIONING'?
012446  001003                            BNE     4$
012450  010137  001162                    MOV     R1,$REG0        ;GET DRIVE NUMBER
012454  104010                            ERROR   10              ;POSITIONING FLAG (IN THE KEY) IS NOT
                                                                  ;SET FOR THE INTERRUPTING DRIVE (GIVEN
                                                                  ;IN EROR MESAGE)

012456  105761  001436            4$:     TSTB    POS(R1)         ;IS THE 'POSITIONING' FLAG SET?
012462  001003                            BNE     5$
012464  010137  001162                    MOV     R1,$REG0        ;GET DRIVE # FROM RKDA
012470  104010                            ERROR   10              ;THIS INTERRUPT IS SUPPOSED TO BE AS
                                                                  ;A RESULT OF INITIATING A (POSITIONING)
                                                                  ;SEEK.  WHENEVER A SEEK IS INITIATED
                                                                  ;TO POSITION THE HEADS THE POSITIONING
                                                                  ;FLAG (POS#) IS SET. OCCURANCE OF THIS
                                                                  ;ERROR INDICATED THAT THE DRIVE #
                                                                  ;(FROM RKDA)GIVING THIS INTERRUPT DID
                                                                  ;NOT HAVE ITS POSITIONING FLAG SET.

012472  005777  166524            5$:     TST     @RKCS           ;ANY ERROR?
012476  100044                            BPL     8$              ;NO - RETURN

012500  032737  000040  001474            BIT     #BIT5,ERCODE
012506  001012                            BNE     6$              ;SAME ERROR
012510  042737  000040  001474            BIC     #BIT5,ERCODE
012516  001026                            BNE     7$
012520  052737  000040  001474            BIS     #BIT5,ERCODE

012526  004737  022130                    JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
                                                                  ;TYPING SERIAL DRIVE #
012532  104011                            ERROR   11              ;BIT 15, 'ERR' SET IN RKKCS AFTER FIRST
                                                                  ;INTERRUPT - WHICH OCCURRED AFTER A
                                                                  ;POSITIONING SEEK WAS INITIATED.  RKER
                                                                  ;WILL CONTAIN ERROR BIT/S
012534  042710  000020            6$:     BIC     #BIT4,(R0)      ;CLEAR 'POSITIONING' BIT
012540  105061  001426                    CLRB    BUSY(R1)        ;CLEAR 'BUSY' FLAG
012544  105061  001436                    CLRB    POS(R1)         ;CLEAR 'POSITIONING' FLAG

012550  004737  015714                    JSR     PC,CLRERR       ;CLEAR THE ERROR
012554  052710  010000                    BIS     #BIT12,(R0)     ;INDICATE HIGH PRIORITY ON
012560  105261  001446                    INCB    RETRY(R1)       ;INCREMENT RETRY COUNT
012564  126127  001446  000003            CMPB    RETRY(R1),#3    ;DONE RETRIES?
012572  003406                            BLE     8$              ;NO

012574  004737  014334            7$:     JSR     PC,DRVABT       ;ABORT THE FUNCTION
012600  005061  001446                    CLR     RETRY(R1)
012604  005037  001474                    CLR     ERCODE

012610  000002                    8$:     RTI
```

```
                                                                ;THIS ROUTINE IS ENTERED UPON COMPLETION
                                                                ;OF A SEEK OPERATION UNDER INTERRUPT MODE

012612  004737  020200          SKCMP:  JSR     PC,CHKCRDY      ;CONTROL READY SET?
012616  104013                          ERROR   13              ;CONTROL RDY NOT SET, AFTER A SEEK
                                                                ;DONE INTERRUPT!

012620  017746  166372          1$:     MOV     @RKDS,-(SP)     ;GET DRIVE # THAT INTERRUPTED
012624  004737  022076                  JSR     PC,CROTLF       ;ROTATE BITS 15,14,13 TO POSITION
                                                                ;0,1,2
012630  012601                          MOV     (SP)+,R1        ;GET THE ROTATED BITS (DRIVE ID BITS)
012632  105761  001426                  TSTB    BUSY(R1)        ;CHECK THAT DRIVE WAS BUSY
012636  100403                          BMI     2$
012640  004737  021740                  JSR     PC,GT4RG
012644  104014                          ERROR   14              ;'BUSY' FLAG FOR THE INTERRUPTING DRIVE
                                                                ;(RKDS) WAS NOT SET.  DRIVE # (15,14,13 - RKDS)
                                                                ;INTERRUPTED AFTER SEEK WAS COMPLETE, BUT
                                                                ;'BUSY' FLAG CORRSPONDING TO THAT DRIVE
                                                                ;WAS CLEAR.  IF THE CORRECT DRIVE WAS
                                                                ;INTERRUPTING 'BUSY' FLAG SHOULD HAVE
                                                                ;BEEN SET.
012646  116100  001426          2$:     MOVB    BUSY(R1),R0     ;FORM THE ADDRESS OF THE
012652  042700  177600                  BIC     #177600,R0      ;KEY
012656  062700  001306                  ADD     #KEY,R0
012662  032710  000020                  BIT     #BIT4,(R0)      ;CHECK THAT POSITION BIT WAS SET
012666  001003                          BNE     3$
012670  010137  001162                  MOV     R1,$REG0        ;GET DRIVE NUMBER
012674  104010                          ERROR   10              ;POSITIONING FLAG (IN KEY) IS NOT SET
                                                                ;SET FOR INTERRUPTING DRIVE (GIVEN IN EROR
                                                                ;MESAGE)
012676  105761  001436          3$:     TSTB    POS(R1)         ;CHECK POSITIONING FLAG WAS SET
012702  001003                          BNE     4$
012704  010137  001162                  MOV     R1,$REG0
012710  104010                          ERROR   10              ;'POSITIONING' FLAG WAS CLEAR.  WHEN
                                                                ;DOING A (POSITIONING) SEEK, THE FLAG
                                                                ;'POS#' IS SET TO INDICATE THAT DRIVE
                                                                ;IS POSITIONING HEADS (SEEKING).  THIS
                                                                ;ERROR INDICATES THAT FOR THE INTERRUPTING
                                                                ;DRIVE (RKDS) POSITIONING FLAG WAS
                                                                ;NOT SET.
012712  004737  016446          4$:     JSR     PC,CHKDRV       ;GO, CHECK FOR DRV ERORS (DPL,DRU,DRY,WPS)
012716  000501                          BR      PFTERR          ;IF THERE IS A DRV EROR, RETURN HERE
                                                                ;THE DRIVE HAS BEEN DESELECTED, ALL
                                                                ;FURTHER COMANDS ON THAT DRIVE ABORTED
                                                                ;RETURRN HERE IF NO DRIVE ERRORS
012720  017777  166272  166302          MOV     @RKDS,@RKDA     ;GET THE DRIVE # FROM DRIVE ID BITS
                                                                ;IN RKDS AND ADDRESS THE DRIVE
012726  032777  000100  166262          BIT     #RWS,@RKDS      ;R/W/S RDY SET?
012734  001005                          BNE     5$              ;YES
012736  004737  021740                  JSR     PC,GT4RG        ;GET RKCS,ER,DS,DA
012742  010137  001250                  MOV     R1,$RDRV        ;GET DRIVE #, FOR TYPING SERIAL #
012746  104015                          ERROR   15              ;R/W/S RDY NOT SET FOR INTERRUPTING
                                                                ;DRIVE (RKDS), AFTER SEEK WAS COMPLETED.
012750  032777  001000  166240  5$:     BIT     #SIN,@RKDS      ;'SIN' ERROR?
012756  001007                          BNE     PSINER          ;YES, BRANCH

```

```
012760  032777  040000  166234          BIT     #HE,@RKCS       ;ANY HARD ERROR?
012766  001060                          BNE     PORKER          ;YES
                                                                ;NO ERRORS DETECTED ON POSITIONING
012770  105061  001426                  CLRB    BUSY(R1)        ;CLEAR BUSY FLAG FOR THIS DRIVE
012774  000207                          RTS     PC
```

```
                                                                  ;THERE WAS A 'SIN' ERROR ON
                                                                  ;DOING POSITIONING (SEEK)
 012776  004737  021740          PSINER: JSR     PC,GT4RG
 013002  010137  001250                  MOV     R1,SRDRV         ;GET DRIVE #, FOR TYPING SERIAL #
 013006  104016                          ERROR   16               ;'SIN' ON DOING (POSITIONING) SEEK
 013010  042710  000020                  BIC     #BIT4,(R0)       ;CLEAR 'POSITIONING' BIT
 013014  105061  001436                  CLRB    POS(R1)          ;CLEAR POSITIONING FLAG
 013020  105061  001426                  CLRB    BUSY(R1)         ;CLEAR BUSY FLAG FOR THIS DRIVE
 013024  004737  016060                  JSR     PC,CLRSIN        ;CLEAR 'SIN' ERROR

 013030  004737  016172                  JSR     PC,SINCNT        ;KEEP COUNT OF 'SIN' ERRORS, IF MORE
                                                                  ;THAN 'MAX' DESELECT THE DRIVE
 013034  000432                          BR      PFTERR           ;RETUN HERE IF COUNT=MAX


                                                                  ;RETURN HERE IF COUNT LESS THAN MAX
 013036  105261  001446          PSRTRY: INCB    RETRY(R1)        ;INCRMNT REETRY COUNT
 013042  105061  001436                  CLRB    POS(R1)          ;CLEAR POSITION FLAG
 013046  105061  001426                  CLRB    BUSY(R1)         ;CLEAR BUSY FLAG
 013052  122761  000003  001446          CMPB    #3,RETRY(R1)     ;DONE ALL RETRIES?
 013060  001403                          BEQ     1$               ;YES
 013062  052710  010000                  BIS     #BIT12,(R0)      ;SET HI PRIORITY ON RETRY
 013066  000207                          RTS     PC               ;RETURN

 013070  052710  104000          1$:     BIS     #104000,(R0)     ;INDICATE COMMAND ABORTED
 013074  105061  001446                  CLRB    RETRY(R1)        ;CLEAR RETRY COUNT FOR FUTURE USE
 013100  010102                          MOV     R1,R2
 013102  006302                          ASL     R2
 013104  005262  001672                  INC     ABORT(R2)        ;INCREMENT ABORT COUNT
 013110  042710  010000                  BIC     #BIT12,(R0)      ;CLR HI PRIORITY  BIT
 013114  104421  002165                  TYPMSG  ,MSG10           ;PRINT ABORT MESSAGE
 013120  000207                          RTS     PC

 013122  042710  010000          PFTERR: BIC     #BIT12,(R0)      ;CLEAR HI PRIORITY BIT IF SET
 013126  000207                          RTS     PC

 013130  004737  021740          PORKER: JSR     PC,GT4RG         ;IF ANY ERROR BIT IN RKCS SET
 013134  010137  001250                  MOV     R1,SRDRV         ;GET DRIVE #, FOR TYPING SERIAL #
 013140  104017                          ERROR   17               ;ON DOING POSITIONING (SEEK)
                                                                  ;ENTER HERE

 013142  004737  015714                  JSR     PC,CLRERR        ;GO CLEAR THE HARD ERROR
 013146  042710  000020                  BIC     #BIT4,(R0)       ;CLEAR POSITIONING BIT IN KEY
 013152  105061  001436                  CLRB    POS(R1)          ;CLEAR POSITIONING FLAG FOR THIS DRIVE
 013156  105061  001426                  CLRB    BUSY(R1)         ;CLEAR BUSY FLAG FOR THIS DRIVE
 013162  000725                          BR      PSRTRY
```

```
                                                                  ;ENTER THIS ROUTINE WHEN AN INTERRUPT
                                                                  ;OCCURS. NOTE THERE IS ONE OTHER
                                                                  ;INTERRUPT ROUTINE- 'INTISK'
 013164  105037  001534          INTHND: CLRB    INTFLG           ;CLEAR FLAG TO INDICATE THAT
                                                                  ;AN INTERRUPT OCCURED.
 013170  013766  001244  000002          MOV     PPRLVL,2(SP)     ;CPU PRIORITY TO 5 ON RTI
 013176  004737  020200                  JSR     PC,CHKCRDY       ;CONTROL READY SET?
 013202  104024                          ERROR   24               ;CONTROL RDY CLEAR AFTER INTRUPT ON COMPLETION
                                                                  ;OF FUNCTION, IT SHOULD BE SET
 013204  032777  020000  166010  1$:     BIT     #SCP,@RKCS       ;SCP SET? SEARCH COMPLETE?
 013212  001403                          BEQ     2$
 013214  004737  012612                  JSR     PC,SKCMP         ;GO TO SEEK COMPLETE ROUTINE
 013220  000002                          RTI
 013222  017746  166002          2$:     MOV     @RKDA,-(SP)      ;GET THE DRIVE # TO WHICH
                                                                  ;THE COMMAND WAS ISSUED UNDER
                                                                  ;INTERRUPT MODE
 013226  004737  022076                  JSR     PC,CROTLF        ;ROTATE BITS 15,14,13 TO POSITION
                                                                  ;0,1,2
 013232  012605                          MOV     (SP)+,R5         ;POP OFF THE DRIVE # FROM THE STACK
 013234  105765  001426                  TSTB    BUSY(R5)         ;CHECK THAT DRIVE WAS BUSY
 013240  100403                          BMI     3$
 013242  010537  001162                  MOV     R5,$REG0
 013246  104007                          ERROR   7                ;'BUSY' FLAG FOR THE INTERUPTING DRIVE
                                                                  ;WAS NOT SET.  WHENEVER ANY ACTIVITY
                                                                  ;IS INITIATED ON A DRIVE, THE (SOFTWARE)
                                                                  ;'BUSY' FLAG IS SET TO INDICATE THAT
                                                                  ;DRIVE IS BUSY DOING SOMETHING.
                                                                  ;OCCURANCE OF THIS ERROR INDICATES
                                                                  ;THAT THE 'BUSY' FLAG FOR THE INTERRUPTING
                                                                  ;DRIVE (RKDA) IS CLEAR!
 013250  105065  001436          3$:     CLRB    POS(R5)          ;CLEAR THE POSITION FLAG
 013254  116500  001426                  MOVB    BUSY(R5),R0      ;GET THE ADDRESS OF THE
 013260  042700  177760                  BIC     #177760,R0       ;COMMAND KEY
 013264  062700  001306                  ADD     #KEY,R0
 013270  032710  000010                  BIT     #BIT3,(R0)       ;CHECK IT'S NOT A SEEK FUNCTION
 013274  001401                          BEQ     4$
 013276  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
                                                                  ;MISINTERPRETATION
 013300  105710                  4$:     TSTB    (R0)             ;CHECK THAT FUNCTION IN
 013302  100403                          BMI     5$               ;PROGRESS BIT WAS SET
 013304  010537  001162                  MOV     R5,$REG0         ;GET DRIVE NUMBER
 013310  104031                          ERROR   31               ;IF NOT, ERROR
                                                                  ;'FUNCTION IN PROGRESS' FLAG (IN THE KEY)
                                                                  ;WAS NOT SET FOR THE INTERRUPTING DRIVE.
                                                                  ;IF THIS DRIVE WAS BUSY DOING THE
                                                                  ;FUNCTION, THE ABOVE FLAG SHOULD BE SET.
 013312  011002                  5$:     MOV     (R0),R2          ;CHECK THAT THE INTERRUPTING
 013314  042702  177770                  BIC     #177770,R2       ;DRIVE (IN RKDA) IS THE SAME AS
 013320  020502                          CMP     R5,R2            ;THE DRIVE IN THE KEY
 013322  001405                          BEQ     6$
```

```
013324  010537  001162                   MOV     R5,$REG0        ;GET EXPCTD DRIVE NO.
013330  010237  001164                   MOV     R2,$REG1        ;GET DRIVE NO. RECVD
013334  104032                           ERROR   32              ;UNEXPECTED DRIVE NO. INTERRUPTED
013336  006302                   6$:     ASL     R2
013340  011003                           MOV     (R0),R3
013342  010037  001536                   MOV     R0,SAVKEY
013346  000303                           SWAB    R3
013350  042703  177770                   BIC     #177770,R3      ;GET POSITION OF THE PARAMETER
013354  006303                           ASL     R3              ;LIST FROM THE KEY
013356  017704  165640                   MOV     @RKCS,R4
013362  042704  177761                   BIC     #177761,R4      ;CLEAR ALL BUT FUNCTION CODE
013366  016305  002032                   MOV     PCMND(R3),R5    ;CHECK THAT THE FUNCTION IN
                                                                 ;RKCS AND THE KEY ARE SAME.
013372  126504  000002                   CMPB    2(R5),R4

013376  001406                           BEQ     7$              ;IF NOT, ERROR
013400  016537  000002  001162           MOV     2(R5),$REG0     ;GET EXPCTD FUNCTION CODE IN RKCS
013406  010437  001164                   MOV     R4,$REG1        ;GET CODE RECVD
013412  104033                           ERROR   33              ;FUNCTION CODE THAT IS IN RKCS AFTER THIS
                                                                 ;INTERRUPT OCCURED IS NOT THE EXPECTED ONE
013414  042710  000200           7$:     BIC     #BIT7,(R0)      ;CLEAR 'FUNCTION IN PROGRESS' BIT
013420  142761  000200  001426           BICB    #BIT7,BUSY(R1)  ;CLEAR BUSY FLAG FOR THIS DRIVE
013426  004737  016446                   JSR     PC,CHKDRV       ;CHECK FOR DRIVE ERRORS
013432  000002                           RTI                     ;IF THERE WAS ANY DRIVE EROR
                                                                 ;DESELECT THE DRIVE AND EXIT
                                                                 ;IF NO ERROR, RETURN HERE
013434  032777  001000  165554           BIT     #SIN,@RKDS      ;SIN ERROR?
013442  001426                           BEQ     10$
013444  004737  022130                   JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
                                                                 ;TYPING SERIAL DRIVE #
013450  104016                           ERROR   16              ;SIN ERROR
013452  004737  016060                   JSR     PC,CLRSIN
013456  004737  016172                   JSR     PC,SINCNT       ;HELP COUNT OF SIN'S. IF MORE
                                                                 ;THAN MAXM ALLOWABLE, DESELECT THE DRIVE
013462  000002                           RTI                     ;RETURN HERE IF COUNT=MAX
                                                                 ;RETURN HERE IF LESS THAN MAX
013464  105261  001446                   INCB    RETRY(R1)
013470  122761  000003  001446           CMPB    #3,RETRY(R1)
013476  001405                           BEQ     8$
013500  052710  010000                   BIS     #BIT12,(R0)
013504  042710  000020                   BIC     #BIT4,(R0)
013510  000002                           RTI

013512  004737  014334           8$:     JSR     PC,DRVABT       ;ABORT THIS FUNCTION
013516  000002                   9$:     RTI

013520  032777  040000  165474   10$:    BIT     #HE,@RKCS       ;HARD ERROR?
013526  001076                           BNE     HRDERR
013530  032777  100000  165464           BIT     #ERR,@RKCS      ;SOFT ERROR?
013536  001002                           BNE     SFTERR          ;YES
013540  000137  014206                   JMP     NOEROR          ;NO
```

```
                                                                 ;THERE WAS A SOFT ERROR


013544  032777  000001  165446   SFTERR: BIT     #WCE,@RKER      ;WCE?
013552  001417                           BEQ     1$

013554  032737  000001  001474           BIT     #BIT0,ERCODE    ;ALREADY WORKING ON A WC ERROR?
013562  001054                           BNE     3$              ;YES - IGNORE THIS ONE
013564  042737  000001  001474           BIC     #BIT0,ERCODE    ;ANY OTHER ERROR?
013572  001052                           BNE     4$              ;YES - ABORT THIS FUNCTION
013574  052737  000001  001474           BIS     #BIT0,ERCODE    ;SET THE WC ERROR BIT

013602  005262  001632                   INC     WCECN(R2)       ;INCREMENT WCE COUNT FOR
013606  104421  002070                   TYPMSG  ,MSG2           ;THIS DRIVE

013612  032777  000002  165400   1$:     BIT     #CSE,@RKER      ;CSE?
013620  001417                           BEQ     2$

013622  032737  000002  001474           BIT     #BIT1,ERCODE    ;ALREADY WORKING ON A CS ERROR?
013630  001031                           BNE     3$              ;YES - IGNORE THIS ONE
013632  042737  000002  001474           BIC     #BIT1,ERCODE    ;ANY OTHER ERROR?
013640  001027                           BNE     4$              ;YES - ABORT THIS FUNCTION
013642  052737  000002  001474           BIS     #BIT1,ERCODE    ;SET THE CS ERROR BIT

013650  005262  001652                   INC     CSECN(R2)       ;INCREMENT CSE COUNT FOR THIS DRIVE
013654  104421  002076                   TYPMSG  ,MSG3
013660  104421  002120           2$:     TYPMSG  ,MSG5
013664  004737  021644                   JSR     PC,TYPFN        ;GO TYPE OUT THE FUNCTION THAT GAVE ERROR
013670  004737  021772                   JSR     PC,GETINF
013674  004737  022134                   JSR     PC,CTSDRV       ;SAVE DRIVE #, FOR TYPING SERIAL #
013700  104021                           ERROR   21              ;SOFT ERROR ON PERFORMING A DATA
                                                                 ;TRANSFER RKDA PRINTED OUT IN ERROR
                                                                 ;MESSAGE IS BROKEN DOWN INTO DRVE #,
                                                                 ;CYL # SEC #, SUR #
013702  022704  000004                   CMP     #4,R4           ;WAS IT A READ FUNCTION?
013706  001002                           BNE     3$
013710  004737  017016                   JSR     PC,DATCHK       ;GO CHECK THE DATA READ

013714  000137  014066           3$:     JMP     CMNERR
013720  000137  014124           4$:     JMP     CMNABT
```

```
                                                                 ;IF THERE WAS A HARD
                                                                 ;ERROR DURING THE DATA
                                                                 ;TRANSFER ENTER HERE

013724  032777  010000  165266  HRDERR: BIT     #SKE,@RKER      ;SKE?
013732  001421                          BEQ     1$

013734  032737  000004  001474          BIT     #BIT2,ERCODE    ;ALREADY WORKING ON A SK ERROR?
013742  001051                          BNE     CMNERR          ;YES - IGNORE THIS ONE
013744  042737  000004  001474          BIC     #BIT2,ERCODE    ;ANY OTHER ERROR?
013752  001064                          BNE     CMNABT          ;YES - ABORT THIS FUNCTION
013754  052737  000004  001474          BIS     #BIT2,ERCODE    ;SET THE SK ERROR BIT

013762  005262  001602                  INC     SKECN(R2)       ;INCREMENT COUNT FOR SKE
013766  005262  001562                  INC     HECN(R2)        ;ON THIS DRIVE

013772  104421  002062                  TYPMSG  ,MSG1

013776  032777  167740  165214  1$:     BIT     #167740,@RKER   ;ANY HE BESIDES SKE?
014004  001417                          BEQ     2$

014006  032737  000010  001474          BIT     #BIT3,ERCODE    ;ALREADY WORKING ON A HE ERROR?
014014  001024                          BNE     CMNERR          ;YES - IGNORE THIS ONE
014016  042737  000010  001474          BIC     #BIT3,ERCODE    ;ANY OTHER ERROR?
014024  001037                          BNE     CMNABT          ;YES - ABORT THIS FUNCTION
014026  052737  000010  001474          BIS     #BIT3,ERCODE    ;SET THE HE ERROR BIT

014034  005262  001562                  INC     HECN(R2)        ;INCREMENT HE COUNT FOR
014040  104421  002104                  TYPMSG  ,MSG4           ;THIS DRIVE

014044  104421  002120          2$:     TYPMSG  ,MSG5           ;PRINT 'ON DOING'
014050  004737  021644                  JSR     PC,TYPFN        ;GO PRINT OUT THE FUNCTION
                                                                ;WHICH GAVE THIS ERROR


014054  004737  021772                  JSR     PC,GETINF
014060  004737  022134                  JSR     PC,GTSDRV       ;SAVE DRIVE #, FOR TYPING SERIAL #
014064  104022                          ERROR   22              ;HARD ERROR ON DOING DATA XFER

014066  105261  001446          CMNERR: INCB    RETRY(R1)       ;INCREMENT RETRY COUNT
014072  122761  000003  001446          CMPB    #3,RETRY(R1)    ;3 TRIES IN ALL?
014100  001411                          BEQ     CMNABT          ;YES, ABORT THIS FUNCTION

014102  052777  010000  165426          BIS     #BIT12,@SAVKEY  ;INDICATE HIGH PRIORITY ON RETRY
014110  042777  000020  165420          BIC     #BIT4,@SAVKEY   ;CLEAR 'POSITIONING HEADS', IF SET
014116  004737  015714                  JSR     PC,CLRERR       ;CLEAR THIS ERROR
014122  000002                          RTI

014124  005037  001474          CMNABT: CLR     ERCODE          ;CLEAR ERROR CODE
014130  104421  002532                  TYPMSG  ,MSG27
014134  004737  014334                  JSR     PC,DRVABT       ;ABORT THE FUNCTION
014140  032777  010000  165052          BIT     #SKE,@RKER      ;SEEK ERROR?
014146  001416                          BEQ     3$              ;NO
014150  104416                          CON.RESET               ;RESET THE CONTROLLER
014152  010137  001502                  MOV     R1,QDRV         ;SET DRIVE NUMBER
014156  000241                          CLC                     ;CLEAR THE 'C' BIT
014160  006037  001502                  ROR     QDRV            ;LEFT JUSTIFY DRIVE NUMBER
```

```
014164  006037  001502                  ROR     QDRV            ;LEFT JUSTIFY DRIVE NUMBER
014170  006037  001502                  ROR     QDRV            ;LEFT JUSTIFY DRIVE NUMBER
014174  006037  001502                  ROR     QDRV            ;LEFT JUSTIFY DRIVE NUMBER
014200  104420                          DRV.RESET               ;RESET THE DRIVE
014202  104416                          CON.RESET               ;RESET THE CONTROLLER
014204  000002                  3$:     RTI                     ;THIS DATA TRANSFER
```

```
                                                                   ;IF THERE WAS NO HARD OR
                                                                   ;SOFT ERROR ON DATA TRANSFER
                                                                   ;ENTER HERE
014206  105761  001446          NOEROR: TSTB    RETRY(R1)
014212  001406                          BEQ     1$
014214  104421  002604                  TYPMSG  ,MSG28
014220  116146  001446                  MOVB    RETRY(R1),-(SP)
014224  104403                          TYPOS
014226     002                          .BYTE   2
014227     000                          .BYTE   0

014230  005037  001474          1$:     CLR     ERCODE
014234  105061  001446                  CLRB    RETRY(R1)
014240  042777  010000  165270          BIC     #BIT12,@SAVKEY  ;CLEAR PRIORITY BIT IF SET
014246  004737  020714                  JSR     PC,STATSTC      ;GO, COLLECT STATISTICS ON THIS
                                                                ;DATA TRANSFER

014252  022704  000004                  CMP     #4,R4           ;WAS IT A 'READ' FUNCTION?
014256  001002                          BNE     2$
014260  004737  017016                  JSR     PC,DATCHK       ;IF IT WAS 'READ', CHECK THE DATA



014264  022704  000002          2$:     CMP     #2,R4           ;WAS IT A 'WRITE' FUNCTION?
014270  001011                          BNE     3$
014272  032777  000040  165236          BIT     #BIT5,@SAVKEY   ;IS 'WRT CHK' TO FOLLOW THIS
014300  001412                          BEQ     4$              ;'WRT' FUNCTION?
014302  052703  100000                  BIS     #BIT15,R3       ;SET FLAG BIT TO INDICATE
                                                                ;THAT WRITE CHECK SHOULD
                                                                ;FOLLOW THIS WRITE
014306  010337  001456                  MOV     R3,WCFLG        ;SET UP WC FLAG TO INDICATE
                                                                ;THE ABOVE, THE LOWER BYTE
                                                                ;CONTAINS THE POINTER TO THE
                                                                ;COMMAND LIST, WHICH WILL
                                                                ;BE USED, FOR DOING WRITE CHECK
014312  000407                          BR      5$              ;IF WRITE CHECK IS TO BE DONE, DONT
                                                                ;SET BIT 15 OF THE KEY TILL WRT CHK
                                                                ;IS DONE

014314  022704  000006          3$:     CMP     #6,R4           ;WRT CHK FUNCTION?
014320  001002                          BNE     4$
014322  005037  001456                  CLR     WCFLG           ;CLEAR WR CHK FLAG

014326  052710  100000          4$:     BIS     #BIT15,(R0)     ;SET FLAG TO INDICATE THIS
014332  000002                  5$:     RTI                     ;FUNCTION IS COMPLETED
```

```
                                                                ;ABORT THE FUNCTION ON DRIVE POINTED TO BY R1
                                                                ;CLEAR ASSOCIATED FLAGS
                                                                ;DROP THE DRIVE IF MORE THAN 8. ABORTS

014334  010246                  DRVABT: MOV     R2,-(SP)        ;SAVE R2
014336  105061  001446                  CLRB    RETRY(R1)
014342  052710  104000                  BIS     #104000,(R0)    ;INDICATE THAT FUNCTION IS ABORTED
014346  042710  010000                  BIC     #BIT12,(R0)     ;CLEAR HIGH PRIORITY BIT IF SET
014352  010102                          MOV     R1,R2
014354  006302                          ASL     R2
014356  005262  001672                  INC     ABORT(R2)
014362  026227  001672  000010          CMP     ABORT(R2),#10
014370  003402                          BLE     1$
014372  000137  016220                  JMP     DSELCT          ;DROP THE DRIVE
014376  012602                  1$:     MOV     (SP)+,R2        ;RESTORE R2
014400  104401  002165                  TYPE    ,MSG10          ;TYPE "ABORTED"
014404  000207                          RTS     PC              ;RETURN
```

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

                                ;THE QUEUE IS LOCATED AT 'CMND' (TO 'CMND8').

Ø144Ø6  1Ø44Ø7                  GEN8RQ: CKSWR
Ø1441Ø  ØØ5337  ØØ2Ø56                  DEC     REPCNT          ;DECREMENT REPETITION COUNT
Ø14414  ØØ1Ø25                          BNE     1$              ;CONTINUE IF NOT Ø
Ø14416  Ø13737  ØØ1236  ØØ2Ø56          MOV     PCNTR,REPCNT    ;REESTABLISH REPETITION COUNT FOR EXERCISER
Ø14424  1Ø44Ø1  Ø14432                  TYPE    ,65$            ;;TYPE ASCIZ STRING
Ø1443Ø  ØØØ4Ø7                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$: .ASCIZ  <15><12>/END OF PASS/
Ø1445Ø                          64$:
Ø1445Ø  Ø13746  ØØ11ØØ                  MOV     $PASS,-(SP)
Ø14454  ØØ5216                          INC     (SP)
Ø14456  1Ø44Ø5                          TYPDS
Ø1446Ø  ØØ4737  Ø21Ø24                  JSR     PC,REPSTAT
Ø14464  ØØØ137  Ø22612                  JMP     $EOP

Ø1447Ø  Ø32777  ØØØ4ØØ  164442  1$:     BIT     #SW8,@SWR
Ø14476  ØØ1422                          BEQ     2$
Ø145ØØ  Ø32777  ØØØØØ1  164432          BIT     #SWØØ,@SWR      ;SWITCH Ø SET ?
Ø145Ø6  ØØ141Ø                          BEQ     8$              ;BR IF NOT
Ø1451Ø  ØØ5727                          TST     (PC)+           ;CHECK INDICATOR
Ø14512  ØØØØØØ                  7$:     .WORD   Ø               ;TYPE TIME INDICATOR
Ø14514  ØØ1Ø13                          BNE     2$              ;BR IF TIME ALREADY TYPED
Ø14516  ØØ5237  Ø14512                  INC     7$              ;INCREMENT THE INDICATOR
Ø14522  ØØ4737  Ø26556                  JSR     PC,TIMTYP       ;TYPE THE TIME (IF SWR Ø3 SET)
Ø14526  ØØØ4Ø6                          BR      2$              ;CONTINUE
Ø1453Ø  ØØ5Ø37  Ø14512          8$:     CLR     7$              ;CLEAR THE INDICATOR
Ø14534  1Ø44Ø1  ØØ1213                  TYPE    ,SCRLF
Ø1454Ø  ØØ4737  Ø21Ø24                  JSR     PC,REPSTAT
Ø14544  Ø32777  ØØØØ4Ø  164366  2$:     BIT     #SW5,@SWR       ;HALT?
Ø14552  ØØ14Ø1                          BEQ     3$              ;NO
Ø14554  ØØØØØØ                          HALT                    ;YES, PRESSING CONTINUE RESUMES PROG EXECUTION

Ø14556  ØØ4737  Ø22564          3$:     JSR     PC,CHDPRS       ;CHECK IF ANY DRIVES PRESENT
Ø14562  Ø127ØØ  ØØ13Ø6          4$:     MOV     #KEY,RØ
Ø14566  Ø1ØØØ4                          MOV     RØ,R4
Ø1457Ø  ØØ5ØØ1                          CLR     R1
Ø14572  Ø1Ø12Ø                  5$:     MOV     R1,(RØ)+        ;CLEAR THE 8 COMMAND KEYS
Ø14574  Ø627Ø1  ØØØ4ØØ                  ADD     #4ØØ,R1         ;BITS 8,9,1Ø INDICATE THEIR
Ø146ØØ  Ø227Ø1  ØØ4ØØØ                  CMP     #4ØØØ,R1        ;POSITION Ø,1,2,3,4,5,6,7
Ø146Ø4  ØØ1372                          BNE     5$
Ø146Ø6  Ø127ØØ  ØØ1326                  MOV     #CMND,RØ        ;CLEAR THE PARAMETER TABLE
Ø14612  Ø1ØØØ5                          MOV     RØ,R5
Ø14614  Ø127Ø1  17774Ø                  MOV     #-4Ø,R1         ;FOR THE 8 COMMANDS IN Q
Ø1462Ø  ØØ5Ø2Ø                  6$:     CLR     (RØ)+           ;(8X4) WORDS IN ALL
Ø14622  ØØ52Ø1                          INC     R1
Ø14624  ØØ1375                          BNE     6$
```

```
Ø14626  ØØ4737  Ø15676                  JSR     PC,CLRFLGS      ;CLEAR THE FLAGS PERTAINING TO THE 8
                                                                ;COMMANDS IN THE Q
                                                                ;R4 CONTAINS 'KEY'
                                                                ;R5 CONTAINS 'CMND'
Ø14632  Ø22737  ØØØØØ1  ØØ1264  GEN1:   CMP     #1,DRVPRS       ;ONLY 1 DRV PRESENT?
Ø1464Ø  ØØ1ØØ2                          BNE     22$             ;NO
Ø14642  ØØ5ØØØ                          CLR     RØ              ;YES
Ø14644  ØØØ42Ø                          BR      3$
Ø14646  ØØ4737  Ø255Ø4          22$:    JSR     PC,$RAND        ;GENERATE A RANDOM NUMBER
                                                                ;GET A RANDOM DRIVE NUMBER
                                                                ;FROM THE AVAILABLE DRIVES
Ø14652  Ø25636                          RSDRVL

Ø14654  Ø13746  Ø2564Ø                  MOV     RSDRVH,-(SP)    ;PUT LOW DIVIDEND ON STACK
Ø1466Ø  ØØ5Ø46                          CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
                                                                ;IT ON STACK
Ø14662  Ø13746  ØØ152Ø                  MOV     DRMAP,-(SP)     ;PUSH DIVISOR ON STACK
Ø14666  ØØ4737  Ø2512Ø                  JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
Ø14672  ØØ5726                          TST     (SP)+           ;DISCARD THE REMAINDER. QUOTIENT IS
                                                                ;NOW ON TOP OF THE STACK
Ø14674  Ø126ØØ                          MOV     (SP)+,RØ
Ø14676  Ø2ØØ37  ØØ1264                  CMP     RØ,DRVPRS       ;MAKE SURE CORRECT MAPPING IS DONE
Ø147Ø2  ØØ1ØØ1                          BNE     3$
Ø147Ø4  ØØ53ØØ                          DEC     RØ              ;'QDRVE' CONTAINS RANDOM DRIVE NO.
Ø147Ø6  ØØ5Ø37  ØØ15Ø2          3$:     CLR     QDRV
Ø14712  116Ø37  ØØ1254  ØØ15Ø2          MOVB    PDR(RØ),QDRV
Ø1472Ø  1Ø5737  ØØ15Ø2                  TSTB    QDRV            ;TEST IF TYPE F DRIVE
Ø14724  1ØØØ16                          BPL     32$             ;NOT IF POSITIVE

Ø14726  142737  ØØØ2ØØ  ØØ15Ø2          BICB    #2ØØ,QDRV       ;CLEAR THE FLAG BIT
Ø14734  132737  ØØØØØ1  ØØ15Ø2          BITB    #1,QDRV         ;ODD OR EVEN DRIVE ADDRESS
Ø14742  ØØ14Ø4                          BEQ     31$

Ø14744  ØØ5737  Ø15632                  TST     ODDEVN          ;MUST HAVE BEEN AN ODD ONE
Ø1475Ø  ØØ173Ø                          BEQ     GEN1            ;INSURE THAT ODDS WHAT WE WANT
Ø14752  ØØØ4Ø3                          BR      32$

Ø14754  ØØ5737  Ø15632          31$:    TST     ODDEVN          ;MUST HAVE BEEN AN EVEN ONE
Ø1476Ø  ØØ1332                          BNE     22$             ;NO GOOD - TRy AGAIN

Ø14762  ØØ4737  Ø255Ø4          32$:    JSR     PC,$RAND        ;GENERATE A RANDOM NUMBER
Ø14766  Ø25642                          RSFUNL

Ø1477Ø  Ø13746  Ø25644                  MOV     RSFUNH,-(SP)    ;PUT LOW DIVIDEND ON STACK
Ø14774  ØØ5Ø46                          CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
                                                                ;IT ON STACK
Ø14776  Ø13746  ØØ1526                  MOV     FNMAP,-(SP)     ;PUSH DIVISOR ON STACK
Ø15ØØ2  ØØ4737  Ø2512Ø                  JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
Ø15ØØ6  ØØ5726                          TST     (SP)+           ;DISCARD THE REMAINDER. QUOTIENT IS
                                                                ;NOW ON TOP OF THE STACK
Ø15Ø1Ø  Ø21627  ØØØØØ3                  CMP     (SP),#3         ;MAKE SURE CORRECT FUNCTION IS SELCTD
Ø15Ø14  ØØ1ØØ1                          BNE     2Ø$
Ø15Ø16  ØØ5316                          DEC     (SP)
Ø15Ø2Ø  Ø12637  ØØ1512          2Ø$:    MOV     (SP)+,QFNC      ;THE FUNCTION THAT CAN BE PERFORMED

Ø15Ø24  ØØ4737  Ø255Ø4                  JSR     PC,$RAND        ;GENERATE A RANDOM NUMBER
```

```
015030  025646                          RSCYLL

015032  013746  025650                  MOV     RSCYLH,-(SP)      ;PUT LOW DIVIDEND ON STACK
015036  005046                          CLR     -(SP)             ;CLEAR HIGH DIVIDEND AND PUSH
                                                                  ;IT ON STACK
015040  013746  001522                  MOV     CYLMAP,-(SP)      ;PUSH DIVISOR ON STACK
015044  004737  025120                  JSR     PC,@#$DIV         ;GO TO THE 'DIVIDE' SUBROUTINE
015050  005726                          TST     (SP)+             ;DISCARD THE REMAINDER, QUOTIENT IS
                                                                  ;NOW ON TOP OF THE STACK
015052  012637  001504                  MOV     (SP)+,QCYL        ;(FROM 0-312)
015056  022737  000313  001504          CMP     #313,QCYL
015064  001002                          BNE     4$
015066  005337  001504                  DEC     QCYL              ;'QCYL' CONTAINS RANDOM CYLINDER NO,

015072                          4$:

015072  013746  025650                  MOV     RSCYLH,-(SP)      ;PUT LOW DIVIDEND ON STACK
015076  005046                          CLR     -(SP)             ;CLEAR HIGH DIVIDEND AND PUSH
                                                                  ;IT ON STACK
015100  013746  001524                  MOV     SECMAP,-(SP)      ;PUSH DIVISOR ON STACK
015104  004737  025120                  JSR     PC,@#$DIV         ;GO TO THE 'DIVIDE' SUBROUTINE
015110  005726                          TST     (SP)+             ;DISCARD THE REMAINDER, QUOTIENT IS
                                                                  ;NOW ON TOP OF THE STACK
015112  012637  001510                  MOV     (SP)+,QSEC        ;(BETWEEN 0-13/8)
015116  022737  000014  001510          CMP     #14,QSEC
015124  001002                          BNE     5$
015126  005337  001510                  DEC     QSEC              ;'QSEC' CONTAINS RANDOM SEC #

015132  022737  077777  025650  5$:     CMP     #77777,RSCYLH     ;SELECT SURFACE # RANDOMLY
015140  101005                          BHI     6$
015142  012737  000020  001506          MOV     #BIT4,QSUR        ;SURFACE 1
                                                                  ;R1 CONTAINS THE MAXM WORD COUNT BASED ON
                                                                  ;AVAILABLE DISK SPACE.
                                                                  ;R2 CONTAINS THE MAXM WORD COUNT BASED ON
                                                                  ;AVAILABLE MEMORY SPACE.
015150  005001                          CLR     R1
015152  000404                          BR      7$

015154  005037  001506          6$:     CLR     QSUR              ;SURFACE 0
015160  012701  000014                  MOV     #14,R1            ;14 SECTORS ON SUR 0

                                ;ASSUMING THAT ENOUGH MEMORY IS AVAILABLE THE MAXIMUM TRANSFER THAT CAN
                                ;TAKE PLACE IS 20K WORDS. IF THE RANDOM CYLINDER # > OR = TO 307 AND THE
                                ;SURFACE IS 1, CHANCES ARE THAT THE NUMBER OF WORDS TO BE TRANSFERRED MAY
                                ;BE GREATER THAN THE SPACE AVAILABLE ON THE DISK.  IN THAT CASE, THE WORDS
                                ;COUNT IS TO BE SELECTED IN SUCH AWAY THAT THIS OVERFLOW DOES NOT OCCUR.

015164  023727  001504  000306  7$:     CMP     QCYL,#306         ;IS THE RANDOM CYL # GREATER THAN 306?
015172  002003                          BGE     9$
015174  012701  177777          8$:     MOV     #177777,R1        ;IF YES, MAKE SURE THAT THE
015200  000431                          BR      21$               ;WORD COUNT IS SELECTED PROPERLY
                                                                  ;COMPUTE MAXM WORD COUNT BASED ON
                                                                  ;AVAILABLE DISK SPACE
015202  012700  000014          9$:     MOV     #14,R0            ;COMPUTE # OF SECTORS AVAILABLE ON
015206  163700  001510                  SUB     QSEC,R0           ;CYLINDERS SELECTED
015212  060001                          ADD     R0,R1
```

```
015214  012703  000312                  MOV     #312,R3           ;COMPUTE # OF SECTORS AVAILABLE
015220  163703  001504                  SUB     QCYL,R3           ;BEYOND THE CYLINDER SELECTED
015224  012746  000030                  MOV     #30,-(SP)                 ;;PUT THE MULTIPLER ON THE STACK
015230  010346                          MOV     R3,-(SP)                  ;;PUT THE MULTIPLICAND ON THE STACK
015232  004737  025006                  JSR     PC,@#$MULT        ;;CALL THE MULTPLY ROUTINE
015236  012616                          MOV     (SP)+,(SP)        ;;DISREGARD THE MSB'S
015240  012603                          MOV     (SP)+,R3                  ;;GET THE LSB'S OF THE PRODUCT
015242  060103                          ADD     R1,R3             ;COMPUTE TOTAL # OF SECTORS AVAILABLE
015244  012746  000400                  MOV     #400,-(SP)                ;;PUT THE MULTIPLER ON THE STACK
015250  010346                          MOV     R3,-(SP)                  ;;PUT THE MULTIPLICAND ON THE STACK
015252  004737  025006                  JSR     PC,@#$MULT        ;;CALL THE MULTPLY ROUTINE
015256  012616                          MOV     (SP)+,(SP)        ;;DISREGARD THE MSB'S
015260  012603                          MOV     (SP)+,R3                  ;;GET THE LSB'S OF THE PRODUCT
015262  010301                          MOV     R3,R1             ;COMPUTE TOTAL # OF WORDS-SPACE
                                                                  ;AVAILABLE ON DISK FROM THE SELECTED
                                                                  ;CYL #
                                                                  ;COMPUTE MAXM WORD COUNT BASED ON
                                                                  ;AVAILABLE MEMORY SPACE.
015264  004737  025504          21$:    JSR     PC,$RAND          ;GENERATE RANDOM NUMBER
015270  025652                          RSBAL

015272  013746  025654                  MOV     RSBAH,-(SP)       ;PUT LOW DIVIDEND ON STACK
015276  005046                          CLR     -(SP)             ;CLEAR HIGH DIVIDEND AND PUSH
                                                                  ;IT ON STACK
015300  013746  001530                  MOV     BAMAP,-(SP)       ;PUSH DIVISOR ON STACK
015304  004737  025120                  JSR     PC,@#$DIV         ;GO TO THE 'DIVIDE' SUBROUTINE
015310  005726                          TST     (SP)+             ;DISCARD THE REMAINDER, QUOTIENT IS
                                                                  ;NOW ON TOP OF THE STACK
015312  012637  001514                  MOV     (SP)+,QBUSAD      ;BE USED
015316  006337  001514                  ASL     QBUSAD
015322  063737  002052  001514          ADD     BASEBA,QBUSAD     ;FORM THE RANDOM BUS-ADDRESS
                                                                  ;BY ADDING RANDOM OFFSET TO
                                                                  ;THE BASE BUS-ADDRESS
015330  013703  002054                  MOV     MAXBA,R3          ;COMPUTE # OF WORDS THAT
015334  163703  001514                  SUB     QBUSAD,R3         ;CAN BE TRANSFERRED, USING THE
015340  000241                          CLC
015342  006003                          ROR     R3                ;ABOVE GENERATED BUS-ADDRESS WITHOUT
015344  010302                          MOV     R3,R2             ;CAUSING A NXM

015346  020201                  10$:    CMP     R2,R1             ;SELECT SMALLER OF THE TWO
015350  103401                          BLO     11$               ;WORD-COUNTS THAT WILL BE
                                                                  ;USED FOR GENERATING A RANDOM
015352  010103                          MOV     R1,R3             ;WORD COUNT)

                                                                  ;R3 CONTAINS THE MAXM WORD COUNT
                                                                  ;POSSIBLE. COMPUTE THE WORD COUNT
                                                                  ;MAPPING FACTOR FROM THIS.
015354                          11$:

015354  012746  177777                  MOV     #177777,-(SP)     ;PUT LOW DIVIDEND ON STACK
015360  005046                          CLR     -(SP)             ;CLEAR HIGH DIVIDEND AND PUSH
                                                                  ;IT ON STACK
015362  010346                          MOV     R3,-(SP)          ;PUSH DIVISOR ON STACK
015364  004737  025120                  JSR     PC,@#$DIV         ;GO TO THE 'DIVIDE' SUBROUTINE
015370  005726                          TST     (SP)+             ;DISCARD THE REMAINDER, QUOTIENT IS
                                                                  ;NOW ON TOP OF THE STACK
```

```
015372  005216                          INC     (SP)
015374  012637  001532                  MOV     (SP)+,WCMAP     ;WORD COUNT MAPPING FACTOR

015400  004737  025504                  JSR     PC,$RAND        ;GENERATE A RANDOM NUMBER
015404  025656                          RSWCL

015406  013746  025660                  MOV     RSWCH,-(SP)     ;PUT LOW DIVIDEND ON STACK
015412  005046                          CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
                                                                ;IT ON STACK
015414  013746  001532                  MOV     WCMAP,-(SP)     ;PUSH DIVISOR ON STACK
015420  004737  025120                  JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
015424  005726                          TST     (SP)+           ;DISCARD THE REMAINDER, QUOTIENT IS
                                                                ;NOW ON TOP OF THE STACK
015426  012637  001516                  MOV     (SP)+,QWRCNT    ;'QWRCNT' CONTAINS THE RANDOM
                                                                ;WORD COUNT THAT WILL BE USED
015432  005737  001516                  TST     QWRCNT          ;MAKE SURE THE WORD COUNT IS
015436  003004                          BGT     12$             ;NOT 0
015440  005437  001516                  NEG     QWRCNT          ;TAKE CARE OF ZERO AND NEG NMBRS
015444  005237  001516                  INC     QWRCNT
015450  113700  001502          12$:    MOVB    QDRV,R0
015454  000241                          CLC
015456  006000                          ROR     R0
015460  006000                          ROR     R0              ;POSITION THE DRIVE NUMBER IN
015462  006000                          ROR     R0              ;BITS 15,14,13
015464  006000                          ROR     R0
015466  013701  001504                  MOV     QCYL,R1
015472  000301                          SWAB    R1
015474  000241                          CLC
015476  006001                          ROR     R1              ;POSITION THE CYLINDER NUMBER
015500  006001                          ROR     R1              ;IN BITS 12-5
015502  006001                          ROR     R1
015504  050100                          BIS     R1,R0
015506  053700  001510                  BIS     QSEC,R0         ;R0 CONTAINS THE COMPLETE DISK
015512  053700  001506                  BIS     QSUR,R0         ;ADDRESS
015516  010025                          MOV     R0,(R5)+        ;INSERT RKDA IN THE PARAMETER TABLE
                                                                ;(FOR THE 8 COMMANDS)
                                                                ;WHICH FUNCTION?
                                                                ;0-READ CHECK, 1-READ, 2-WRITE
015520  022737  000001  001512          CMP     #1,QFNC
015526  001412                          BEQ     2$              ;READ
015530  003014                          BGT     14$             ;READ CHECK
015532  012725  000002          1$:     MOV     #2,(R5)+        ;WRITE FUNCTION CODE
015536  023727  025660  077777          CMP     RSWCH,#77777    ;SHOULD WRITE CHECK BE DONE
015544  101010                          BHI     15$             ;AFTER THE WRITE?
015546  052714  000040                  BIS     #BIT5,(R4)      ;SET FLAG IN KEY TO INDICATE
                                                                ;THAT WRITE CHECK SHOULD BE
                                                                ;DONE FOLLOWING THE WRITE
015552  000405                          BR      15$

015554  012725  000004          2$:     MOV     #4,(R5)+        ;READ FUNCTION CODE
015560  000402                          BR      15$

015562  012725  000012          14$:    MOV     #12,(R5)+       ;READ CHECK FUNCTION CODE

015566  013715  001516          15$:    MOV     QWRCNT,(R5)     ;INSERT THE WORD COUNT (RKWC)
015572  005425                          NEG     (R5)+           ;(2'S COMPLEMENT)
```

```
015574  013725  001514                  MOV     QBUSAD,(R5)+    ;INSERT THE BUS ADDRESS (RKBA) FOR
                                                                ;THIS COMMAND IN THE PARAMETER
                                                                ;TABLE
015600  053724  001502          16$:    BIS     QDRV,(R4)+      ;SET THE DRIVE NUMBER INSIDE
                                                                ;THE KEY FOR THIS COMMAND
015604  020427  001326                  CMP     R4,#KEY+20      ;GENERATED 8 COMMANDS IN
                                                                ;THE QUEUE?
015610  001402                          BEQ     17$
015612  000137  014632                  JMP     GEN1            ;IF NOT, GO BACK AND GENERATE
                                                                ;THE NEXT COMMAND AND THE
                                                                ;PARAMETERS (RKWC, BA, DA)
015616  005237  015632          17$:    INC     ODDEVN          ;CHANGE FROM ODD/ENEN TO EVEN/ODD
015622  042737  177776  015632          BIC     #177776,ODDEVN  ;KEEP ONLY ONE BIT
015630  000207                          RTS     PC              ;ALL 8 COMMANDS HAVE BEEN
                                                                ;GENERATED IN THE TASK-QUEUE
015632  000000                  ODDEVN: 0
```

```
                                        ;ENTER THIS CODE IF SW 9 HAS BEEN SET AND LOOPING HAS TO BE DONE ON ERROR
                                        ;CONTROL IS TRANFERRED TO THIS, ON RETURN FROM THE ERROR HANDLER '$ERROR'.


015634  004737  015714                  EXCRLUP: JSR    PC,CLRERR       ;WAIT FOR OTHER DRIVES TO GET DONE
                                                                        ;THEN ISSUE A CONTROL RESET
015640  010146                                  MOV     R1,-(SP)
015642  012701  001306                          MOV     #KEY,R1         ;CLEAR OUT THE COMMAND-KEYS
015646  042721  114220                  1$:     BIC     #114220,(R1)+
015652  020127  001326                          CMP     R1,#KEY+20
015656  001373                                  BNE     1$
015660  012601                                  MOV     (SP)+,R1
015662  004737  015676                          JSR     PC,CLRFLGS      ;CLEAR OUT THE VARIOUS FLAGS PERTAINING
                                                                                ;TO THE 8 COMMANDS IN THE Q
015666  012706  001100                          MOV     #STACK,SP       ;REESTABLISH STACK POINTER
015672  000137  010556                          JMP     QMNGER          ;START OVER AGAIN, PROCESS THE COMMANDS
                                                                        ;IN THE Q AGAIN. NOTE THAT ON LOOPING
                                                                        ;(ON EROR, WITH SW 9) AN ATTEMPT IS MADE
                                                                        ;TO RECREATE THE SET OF EVENTS WHICH LED TO
                                                                        ;ERROR.




                                        ;CLRFLGS
                                        ;THIS ROUTINE CLEARS OUT THE VARIOUS FLAGS USED FOR THE 8 COMMANDS IN THE QUEUE.

015676  012700  001426                  CLRFLGS: MOV    #BUSY,R0        ;CLEAR THE 8 BUSY FLAGS
015702  005020                          1$:     CLR     (R0)+
015704  020027  001462                          CMP     R0,#QSCNT+2     ;ALL DONE?
015710  001374                                  BNE     1$              ;NO
015712  000207                                  RTS     PC
```

```
                                        ;CLRERR
                                        ;THIS ROUTINE IS ENTERED WHEN A HARD ERROR HAS OCCURRED AND IT HAS TO BE
                                        ;CLEARED.  THE DRIVES THAT HAVE BEEN BUSY ARE CHECKED TO SEE IF THEIR R/W/S
                                        ;RDY BIT HAS SET.  WHEN R/W/S IS SET, CHECKING IS DONE FOR ANY ERROR. IF A
                                        ;ERROR OCCURED IT IS REPORTED, IF NOT, APPROPRIATE FLAGS ARE SET AND
                                        ;CLEARED FOR THAT DRIVE. AFTER ABOVE IS DONE FOR ALL DRIVES THAT HAVE BEEN
                                        ;SEEKING, A CONTROL RESET IS ISSUED TO CLEAR THE HARD ERROR.

015714  010446                          CLRERR: MOV     R4,-(SP)        ;SAVE R4, R4 ON THE STACK
015716  010546                                  MOV     R5,-(SP)
015720  005005                                  CLR     R5
015722  005077  163302                          CLR     @RKDA

015726  105765  001426                  1$:     TSTB    BUSY(R5)        ;WAS THIS DRIVE BUSY SEEKING?
015732  100035                                  BPL     4$              ;NO
015734  005037  001472                          CLR     TIMER
015740  032777  000100  163250          2$:     BIT     #RWS,@RKDS      ;R/W/S SET?
015746  001015                                  BNE     3$              ;YES
015750  005237  001472                          INC     TIMER           ;KEEP TIME
015754  001371                                  BNE     2$              ;WAIT FOR R/W/S RDY
015756  004737  022130                          JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
                                                                        ;TYPING SERIAL DRIVE #
015762  104004                                  ERROR   4               ;R/W/S READY DID NOT SET
                                                                        ;FOR THIS DRIVE, WAITED LONG ENOUGH.
015764  032777  001000  163224                  BIT     #SIN,@RKDS      ;SIN ERROR ON THIS DRIVE?
015772  001403                                  BEQ     3$
015774  004737  022130                          JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
                                                                        ;TYPING SERIAL DRIVE #
016000  104016                                  ERROR   16              ;SIN OCCURED ON THIS DRIVE

016002  116504  001426                  3$:     MOVB    BUSY(R5),R4     ;FORM THE ADDRESS OF THE
016006  042704  177760                          BIC     #177760,R4      ;KEY WHICH MADE THIS DRIVE
016012  062704  001306                          ADD     #KEY,R4 ;BUSY

016016  042714  010000                          BIC     #10000,(R4)     ;CLEAR HIGH PRIORITY BIT, IF SET
016022  105065  001426                          CLRB    BUSY(R5)        ;MAKE THIS DRIVE FREE, AVAILABLE


016026  062777  020000  163174          4$:     ADD     #20000,@RKDA    ;ADDRESS THE NEXT POSSIBLE DRIVE
016034  005205                                  INC     R5              ;INCREMENT COUNT
016036  022705  000010                          CMP     #10,R5          ;ALL DONE
016042  001331                                  BNE     1$              ;NO

016044  004737  020246                          JSR     PC,CRCMND       ;SAVE INFO ABOUT THE PAST & PRESENT COMAND
016050  104416                                  CON.RESET
016052  012605                                  MOV     (SP)+,R5        ;RESTORE R4,R5
016054  012604                                  MOV     (SP)+,R4
016056  000207                                  RTS     PC              ;RETURN
```

```
                                   ;CLRSIN
                                   ;THIS ROUTINE IS ENTERED WHEN THERE IS A 'SIN' ERROR.  AT TIME OF ENTRY
                                   ;RKDA CONTAINS THE DRIVE # THAT GAVE 'SIN'.  A DRIVE RESET IS DONE ON THAT
                                   ;DRIVE.  AFTER IT IS DONE, ROUTINE 'CLRHE' IS ENTERED, TO WAIT FOR THE
                                   ;OTHER DRIVES THAT HAVE BEEN DOING SEEKS.  WHEN ALL THE DRIVES GIVE
                                   ;'R/W/S RDY', A CONTROL RESET IS DONE, RETURN IS MADE BACK TO THIS
                                   ;ROUTINE-'CLRSIN'- AND FINALLY CONTROL IS TRANSFERRED BACK TO THE MAIN
                                   ;PROGRAM.

016060  017737  163144  001516  CLRSIN: MOV     @RKDA,QWRCNT    ;SAVE DISK ADDRESS
016066  004737  015714                  JSR     PC,CLRERR       ;GO, WAIT FOR OTHER DRIVES TO COMPLETE
                                                                ;THEIR SEEKS(IF THEY ARE DOING ANY)
                                                                ;THEN DO CON.RESET TO CLR THE EROR.
016072  013777  001516  163130          MOV     QWRCNT,@RKDA    ;ADDRESS THE DRIVE AGAIN
016100  004737  020222                  JSR     PC,DRCMND       ;SAVE INFO ABOUT THE PAST & PRESENT COMAND
016104  012777  000015  163110          MOV     #15,@RKCS       ;DO DRIVE RESET ON THE
                                                                ;DRIVE
                                                                ;INDICATED IN RKDA
016112  104417                          CON.RDY
016114  005037  001472                  CLR     TIMER
016120  032777  000100  163070  1$:     BIT     #RWS,@RKDS      ;WAIT FOR R/W/S RDY TO SET
016126  001015                          BNE     2$
016130  005237  001472                  INC     TIMER
016134  001371                          BNE     1$
016136  004737  022130                  JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
                                                                ;TYPING SERIAL DRIVE #
016142  104004                          ERROR   4               ;R/W/S RDY DID NOT SET AFTER
                                                                ;DOING DRIVE RESET, TIMED OUT
016144  032777  001000  163044          BIT     #SIN,@RKDS
016152  001403                          BEQ     2$
016154  004737  022130                  JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
                                                                ;TYPING SERIAL DRIVE #
016160  104016                          ERROR   16              ;A DRIVE RESET WAS DONE ON THIS DRIVE
                                                                ;TO CLEAR 'SIN', BUT 'SIN' DID NOT GET
                                                                ;CLEARED

016162  004737  020246          2$:     JSR     PC,CRCMND       ;SAVE INFO ABOUT THIS COMMAND
016166  104416                          CON.RESET               ;DO IT TO CLEAR OUT MASK F/FS
016170  000207                          RTS     PC              ;EXIT FROM THIS ROUTINE
```

```
                                   ;SINCNT
                                   ;THIS ROUTINE IS ENTERED WHEN A 'SIN' ERROR OCCURS.  THE 'SIN' COUNT FOR
                                   ;THE DRIVE GIVING 'SIN' IS INCREMENTED.  IF MORE THAN 5 'SIN' ERRORS
                                   ;OCCURRED THE DRIVE IS DESELECTED.  AT THE TIME OF ENTRY R1 CONTAINS THE
                                   ;DRIVE NUMBER THAT GAVE 'SIN' ERROR.
                                   ;CALL:  JSR     PC,SINCNT
                                   ;       -----           RETURN HERE IF SIN COUNT (MAXIMUM ALLOWABLE)
                                   ;                       WAS EXCEEDED.
                                   ;       -----           RETURN HERE IF TOTAL SIN COUNT LESS THAN MAXIMUM
                                   ;                       ALLOWABLE.


016172  105261  001622          SINCNT: INCB    SINCN(R1)       ;INCREMENT 'SIN' COUNT FOR THIS DRIVE
016176  122761  000005  001622          CMPB    #5,SINCN(R1)    ;5 ERRORS OCCURRED?
016204  101403                          BLOS    1$              ;YES
016206  062716  000002                  ADD     #2,(SP)         ;ADJUST PC FOR RETURN TO THE RIGHT POINT
016212  000207                          RTS     PC              ;RETURN

016214  000137  016220          1$:     JMP     DSELCT          ;5 ERRORS OCCURRED, GO DESELECT
```

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



 016220  012705  001253                 DSELCT: MOV     #PDR-1,R5
 016224  013702  001264                         MOV     DRVPRS,R2               ;NUMBER OF DRIVES BEING TESTED
 016230  062702  001253                         ADD     #PDR-1,R2               ;FOR END ADDRESS OF TABLE
 016234  005205                         1$:     INC     R5                      ;LOCATE THE DRIVE (TO BE
 016236  111503                                 MOVB    (R5),R3                 ;DESELECTED) IN THE TABLE
 016240  042703  177600                         BIC     #177600,R3              ;DROP THE F FLAG
 016244  020301                                 CMP     R3,R1                   ;IS THIS THE ONE
 016246  001403                                 BEQ     2$                      ;CONTAINING AVAILABLE DRIVES
 016250  020502                                 CMP     R5,R2                   ;FINISHED ?
 016252  103770                                 BLO     1$                      ;BR IF NOT
 016254  000472                                 BR      11$                     ;DRIVE WAS NOT FOUND IN TABLE, EXIT
 016256  111502                         2$:     MOVB    (R5),R2                 ;GET THE DRIVE NUMBER
 016260  020527  001264                 5$:     CMP     R5,#PDR+10              ;IS THIS DRIVE # THE LAST ENTRY IN TABLE?
 016264  001406                                 BEQ     4$                      ;YES
 016266  010504                                 MOV     R5,R4
 016270  005205                                 INC     R5                      ;IF NOT, TAKE OUT THIS DRIVE # FROM
 016272  112524                         3$:     MOVB    (R5)+,(R4)+             ;THE MIDDLE AND PUSH UP THE
 016274  022704  001263                         CMP     #PDR+7,R4               ;REST OF THE ENTRIES
 016300  001374                                 BNE     3$
 016302  105065  177777                 4$:     CLRB    -1(R5)                  ;CLEAR LAST ENTRY IN TABLE

                                                                                ;THE DRIVE # TYPED OUT WAS DESELECTED
                                                                                ;BECAUSE ERROR COUNT EXCEEDED THE
                                                                                ;MAXIMUM ALLOWABLE
 016306  104401  002336                         TYPE    ,MSG19                  ;TYPE "DRVE DROPPED"
 016312  010146                                 MOV     R1,-(SP)                ;PUSH DRIVE NUMBER ON STACK
 016314  104403                                 TYPOS                           ;TYPE IT ON THE TERMINAL
 016316     001                                 .BYTE   1
 016317     000                                 .BYTE   0
 016320  004737  026664                         JSR     PC,SNOTYP               ;GO TYPE OUT SERIAL NO OF THE DRIVE,
                                                                                ;IF SW 1 IS SET.
 016324  005337  001264                         DEC     DRVPRS                  ;DECREMENT THE TOTAL NUMBER OF
                                                                                ;DRIVES PRESENT
 016330  004737  022564                         JSR     PC,CHDPRS               ;CHECK IF ANY DRIVES PRESENT
                                                                                ;IF NONE GOT TO END OF PASS, $EOP

 016334  012746  177777                         MOV     #177777,-(SP)           ;PUT LOW DIVIDEND ON STACK
 016340  005046                                 CLR     -(SP)                   ;CLEAR HIGH DIVIDEND AND PUSH
                                                                                ;IT ON STACK
 016342  013746  001264                         MOV     DRVPRS,-(SP)            ;PUSH DIVISOR ON STACK
 016346  004737  025120                         JSR     PC,@#$DIV               ;GO TO THE 'DIVIDE' SUBROUTINE
 016352  005726                                 TST     (SP)+                   ;DISCARD THE REMAINDER, QUOTIENT IS
                                                                                ;NOW ON TOP OF THE STACK
 016354  012637  001520                         MOV     (SP)+,DRMAP             ;TO BE USED FOR GENERATING RANDOM
                                                                                ;DRIVE NUMBERS.
```

```
 016360  012704  001306                         MOV     #KEY,R4                 ;DESELECT THE COMMANDS
 016364  011405                         6$:     MOV     (R4),R5                 ;IN THE 'COMMAND Q' CORRESPONDING
 016366  042705  177770                         BIC     #177770,R5              ;TO THE DESELECTED DRIVE
 016372  020105                                 CMP     R1,R5
 016374  001002                                 BNE     7$
 016376  052714  104000                         BIS     #104000,(R4)            ;INDICATE COMMAND DESELECTED
 016402  005724                         7$:     TST     (R4)+                   ;(AND COMPLETED)
 016404  022704  001326                         CMP     #KEY+20,R4
 016410  001365                                 BNE     6$

 016412  105702                         8$:     TSTB    R2                      ;'F' TYPE DRIVE ?
 016414  100012                                 BPL     11$                     ;NO - JUST EXIT
 016416  032701  000001                         BIT     #1,R1                   ;ODD OR EVEN DRIVE NUMBER
 016422  001403                                 BEQ     9$
 016424  042701  000001                         BIC     #1,R1
 016430  000402                                 BR      10$
 016432  052701  000001                 9$:     BIS     #1,R1
 016436  000137  016220                 10$:    JMP     DSELCT                  ;DROP CORRESPONDING DRIVE


 016442  000137  010536                 11$:    JMP     BEGNEX                  ;GO RESTART EXERSISOR PART OF TEST
```

```
                                        ;CHKDRV
                                        ;THIS ROUTINE CHECKS FOR FATAL ERRORS OF THE DRIVE LIKE DPL, DRV
                                        ;WPS.  IF ANY ONE OF THESE ERRORS OCCUR THE DRIVE IS DESELECTED
                                        ;AND NO MORE FUNCTIONS WILL BE PERFORMED ON THAT DRIVE.  R1
                                        ;CONTAINS THE DRIVE NUMBER TO BE CHECKED.

                                        ;*NOTE 1: IN THE ERROR MESSAGE WHERE RKDA IS PRINTED OUT, IT GIVES
                                        ;*ONLY THE DRIVE NUMBER (NOT CYLINDER, SUFACE AND SECTOR).

                                        ;CALL:  JSR     PC,CHKDRV
                                        ;       ----- RETURN HERE IF ANY FATAL ERROR OCCURED
                                        ;       ----- RETURN HERE IF THERE WAS NO FATAL ERROR

016446  017746  162556                  CHKDRV: MOV     @RKDA,-(SP)     ;SAVE RKDA
016452  010146                                  MOV     R1,-(SP)        ;GET DRIVE #
016454  000241                                  CLC
016456  006016                                  ROR     (SP)
016460  006016                                  ROR     (SP)
016462  006016                                  ROR     (SP)
016464  006016                                  ROR     (SP)
016466  012677  162536                          MOV     (SP)+,@RKDA     ;ADDRESS THE DRIVE TO BE CHECKED
016472  032777  010000  162516                  BIT     #DPL,@RKDS      ;DRIVE POWER LOW?
016500  001403                                  BEQ     1$
016502  004737  022130                          JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE #
                                                                        ;FOR TYPING SERIAL NUMBER
016506  104035                                  ERROR   35              ;DRIVE POWER LO, *NOTE 1 ABOVE
016510  032777  002000  162500  1$:             BIT     #DRU,@RKDS      ;DRIVE
016516  001403                                  BEQ     2$
016520  004737  022130                          JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE #
                                                                        ;FOR TYPING SERIAL NUMBER
016524  104036                                  ERROR   36              ;DRIVE UNSAFE BIT IS SET
                                                                        ;*NOTE 1 ABOVE
016526  032777  000040  162462  2$:             BIT     #WPS,@RKDS      ;WRITE PROTECT SET?
016534  001403                                  BEQ     3$
016536  004737  022130                          JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE #
                                                                        ;FOR TYPING SERIAL NUMBER
016542  104037                                  ERROR   37              ;WPS SET, CHECK WRTE PROTECT SWTCH ON DRIVE
                                                                        ;*NOTE 1 ABOVE
016544  032777  000200  162444  3$:             BIT     #DRY,@RKDS      ;DRIVE READY CLEAR?
016552  001004                                  BNE     4$
016554  004737  022130                          JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE #
                                                                        ;FOR TYPING SERIAL NUMBER
016560  104034                                  ERROR   34              ;DRIVE READY CLEAR, SHOULD BE SET
                                                                        ;*NOTE 1 ABOVE
016562  000411                                  BR      5$

016564  032777  012040  162424  4$:             BIT     #12040,@RKDS    ;ANY ERROR?
016572  001005                                  BNE     5$              ;YES
016574  012677  162430                          MOV     (SP)+,@RKDA     ;RESTORE RKDA
016600  062716  000002                          ADD     #2,(SP)         ;ADJUST RETURN ADDRESS
016604  000207                                  RTS     PC

016606  000137  016220          5$:             JMP     DSELCT
```

```
                                        ;GENBUF
                                        ;THIS ROUTINE GENERRAES A BUFFER FULL OF RANDOM DATA WORDS. THIS BUFFER
                                        ;IS THEN USED TO WRITE DATA ON THE DISK.  AT THE TIME OF ENTRY, RKDA
                                        ;CONTAINS THE DISK ADDRESS WHERE WRITE WILL BE DONE. SEED WORDS USED FOR
                                        ;THE RANDOM NUMBERS ARE:
                                        ;       1) ABSOLUTE DISK ADDRESS (DRIVE #, CYL #, SEC #, SUR #) - $HINUM
                                        ;       2) COMPLEMENT OF THE ABOVE WORD
                                        ;CALL:  JSR     R5,GENBUF
                                        ;       X               ;X IS THE WORD COUNT (2'S COMPLEMENT)
                                        ;       Y               ;Y IS THE STARTING ADDRESS OF THE
                                                                ;MEMORY BUFFER.

016612  104414                          GENBUF: SAVREG                  ;SAVE REGISTERS
016614  016504  000002                          MOV     2(R5),R4        ;GET STARTING ADDRESS OF BUFFER
016620  011503                                  MOV     (R5),R3         ;GET WORD COUNT (# OF WORDS TO
                                                                        ;BE GENERATED)

016622  017702  162402                          MOV     @RKDA,R2
016626  010237  025664          1$:             MOV     R2,RSDTH        ;GET THIS RANDOM SEED
016632  010237  025662                          MOV     R2,RSDTL
016636  005137  025662                          COM     RSDTL           ;GET LO RANDOM SEED

016642  022703  177400                          CMP     #-400,R3        ;IF THE BUFFER IS MORE THAN
016646  003003                                  BGT     2$              ;ONE SECTOR (400 WORDS) LONG,
016650  010305                                  MOV     R3,R5           ;GENERATE THE BUFFER IN SUCH
016652  005003                                  CLR     R3              ;A WAY THAT EACH SECTOR
016654  000404                                  BR      3$              ;BEGINS WITH RANDOM DATA

016656  062703  000400          2$:             ADD     #400,R3         ;WORDS GENERATED USING THAT
016662  012705  177400                          MOV     #-400,R5        ;SECTOR ADDRESS AS THE RANDOM
                                                                        ;SEED
016666  010524                  3$:             MOV     R5,(R4)+        ;FIRST WORD OF EVERY SECTOR IS
                                                                        ;A WORD COUNT (2'S COMP) INDICATING  #
                                                                        ;OF WORDS ACTUALLY WRITTEN IN THAT SECTOR
016670  005205                                  INC     R5              ;ALL DONE?
016672  001427                                  BEQ     8$

016674  004737  025504          4$:             JSR     PC,$RAND        ;GENERATE DATA WORDS
016700  025662                                  RSDTL
016702  012737  000002  017012                  MOV     #2,RCNT
016710  013737  025664  017014                  MOV     RSDTH,RNUM
016716  000406                                  BR      6$
016720  005337  017012          5$:             DEC     RCNT
016724  001763                                  BEQ     4$
016726  013737  025662  017014                  MOV     RSDTL,RNUM
016734  005737  017014          6$:             TST     RNUM
016740  001767                                  BEQ     5$
016742  013724  017014                          MOV     RNUM,(R4)+      ;FILL THE BUFFER. DON'T USE
016746  005205                                  INC     R5              ;ALL DONE?
016750  001351                                  BNE     4$              ;NO

016752  005703                  8$:             TST     R3              ;ANY MORE DATA WORDS (FOR
016754  001412                                  BEQ     10$             ;REST OF THE SECTORS)?
                                                                        ;YES
016756  010246                                  MOV     R2,-(SP)
016760  042716  177760                          BIC     #177760,(SP)    ;(ABSOLUTE DISK ADDRESS & ITS
016764  022726  000013                  CMP     #13,(SP)+       ;COMPLEMENT) TO USE FOR
016770  001002                          BNE     9$              ;GENERATING DATA WORDS
016772  062702  000004                  ADD     #4,R2           ;OF THE NEXT BLOCK

016776  005202                  9$:     INC     R2              ;HI RANDOM SEED
017000  000712                          BR      1$

017002  104415                  10$:    RESREG                  ;RESTORE REGISTERS
017004  062705  000004                  ADD     #4,R5           ;ADJUST RETURN ADDRESS
017010  000205                          RTS     R5              ;RETURN

017012  000000                  RCNT:   0
017014  000000                  RNUM:   0
```

```
                                ;DATCHK
                                ;THIS ROUTINE IS ENTERED WHEN THE DATA THAT WAS READ FROM THE DISK IS TO
                                ;BE CHECKED.  AT THE TIME OF ENTRY R3 CONTAINS THE OFFSET OF  POINTER TO
                                ;THE ADDRESS OF THE PARAMETER LIST (RKCS,DA,WC,BA). DATA IS CHECKED IN
                                ;BLOCKS OF 1 SECTOR (400 WORDS).  EACH BLOCK IS GENERATED USING THE SECTOR
                                ;ADDRESS (AND ITS COMPLEMENT) AS RANDOM SEEDS.  WHEN A DATA MISCOMPARISON
                                ;OCCURS THE BUS ADDRESS, EXPECTED AND RECEIVED DATA ARE REPORTED.

017016  104414                  DATCHK: SAVREG                  ;SAVE R0-R5
017020  016303  002032                  MOV     PCMND(R3),R3    ;GET ADDRESS OF THE PARAMETER
                                                                ;TABLE
017024  016304  000004                  MOV     4(R3),R4        ;GET WORD COUNT (2'S COMP)
017030  016305  000006                  MOV     6(R3),R5        ;GET BUS ADDRESS
017034  011301                          MOV     (R3),R1         ;GET DISK ADDRESS

017036  010146                          MOV     R1,-(SP)
017040  004737  022076                  JSR     PC,CROTLF       ;ROTATE BITS 15,14,13 TO
                                                                ;0,1,2
017044  012602                          MOV     (SP)+,R2        ;POP OFF DRIVE # FROM THE STACK
017046  006302                          ASL     R2
017050  062702  001712                  ADD     #DATER,R2       ;FORM THE ADDRESS OF DATA ERROR COUNT
                                                                ;FOR THIS DRIVE
017054  012737  177764  001540          MOV     #-14,ECOUNT
017062  011337  025664                  MOV     (R3),RSDTH      ;CREATE RANDOM SEEDS TO
017066  013737  025664  025662          MOV     RSDTH,RSDTL     ;BE USED FOR CHECKING DATA
017074  005137  025662                  COM     RSDTL

017100  022704  177400          1$:     CMP     #-400,R4        ;DATA IS CHECKED IN 1 SECTOR
017104  003003                          BGT     2$              ;BLOCKS. EACH SECTOR IS GENERATED
017106  010403                          MOV     R4,R3           ;USING THAT SECTOR ADDRESS
017110  005004                          CLR     R4              ;AS THE RANDOM SEED
017112  000404                          BR      3$

017114  062704  000400          2$:     ADD     #400,R4
017120  012703  177400                  MOV     #-400,R3
017124  012500                  3$:     MOV     (R5)+,R0        ;SAVE THE FIRST WORD OF THE SECTOR.
                                                                ;FIRST WORD OF EVERY SECTOR IS A COUNT
                                                                ;(2'S COMP) INDICATING # OF WORDS ACTUALY
                                                                ;WRITTEN IN THAT SECTOR
017126  005200                          INC     R0              ;INCREMENT COUNT OF # OF WORDS  (WRITEN)
                                                                ;IN THE SECTOR
017130  005203                          INC     R3              ;INCRMENT COUNT OF DATA WORDS TO BE CHECKED
017132  001465                          BEQ     14$             ;BRANCH, IF DONE

017134  004737  025504          4$:     JSR     PC,$RAND        ;GENERATE RANDOM DATA WORD
017140  025662                          RSDTL
017142  012737  000002  017012          MOV     #2,RCNT
017150  013737  025664  017014          MOV     RSDTH,RNUM
017156  000406                          BR      10$
017160  005337  017012          9$:     DEC     RCNT
017164  001763                          BEQ     4$
017166  013737  025662  017014          MOV     RSDTL,RNUM
017174  005737  017014          10$:    TST     RNUM
017200  001767                          BEQ     9$

017202  005700                          TST     R0
```

```
017204  001401                          BEQ     5$
017206  005200                          INC     R0

017210  023715  017014          5$:     CMP     RNUM,(R5)       ;EXPCTD WORD = RECVD WORD?
017214  001431                          BEQ     8$              ;YES

017216  005700                          TST     R0
017220  001005                          BNE     6$
017222  005715                          TST     (R5)
017224  001425                          BEQ     8$
017226  005037  001164                  CLR     $REG1
017232  000403                          BR      7$

017234  013737  017014  001164  6$:     MOV     RNUM,$REG1      ;SAVE EXPCTD DATA WORD
017242  005212                  7$:     INC     (R2)            ;INCRMNT DATA EROR COUNT FOR THIS DRVE
017244  005737  001540                  TST     ECOUNT          ;STORE ONLY 12 (DEC) DATA ERRORS
017250  001413                          BEQ     8$              ;IF MORE EXIT
017252  010537  001162                  MOV     R5,$REG0        ;SAVE ERROR BUS ADDRESS
017256  011537  001166                  MOV     (R5),$REG2      ;SAVE ERROR DATA WORD
017262  010137  001170                  MOV     R1,$REG3
017266  004737  022134                  JSR     PC,GTSDRV       ;SAVE DRIVE #, FOR TYPING SERIAL #
017272  104023                          ERROR   23              ;DATA (COMPARISON) ERROR ON DOING
                                                                ;READ FROM DISK NORMALLY ONLY 12 DATA
                                                                ;ERRORS WILL BE REPORTED.  THROUGH
                                                                ;CHECKING WILL BE DONE, ERRORS
                                                                ;EXCEEDING 12 WON'T BE REPORTED.  IF
                                                                ;YOU WANT MORE, CHANGE 'ECOUNT', TO
                                                                ;WHATEVER # OF ERRORS YOU WANT REPORTED
017274  005237  001540                  INC     ECOUNT

017300  005725                  8$:     TST     (R5)+
017302  005203                          INC     R3              ;DONE CHECKING?
017304  001313                          BNE     4$

017306  005704                  14$:    TST     R4              ;ANY MORE SECTOR BLOCKS
017310  001427                          BEQ     17$             ;TO CHECK? IF NOT, EXIT

017312  010146                          MOV     R1,-(SP)
                                                                ;GET THE NEW RANDOM SEEDS
017314  042716  177760                  BIC     #177760,(SP)    ;(ABSOLUTE DISK ADDRESS & ITS COMPLEMENT)
017320  022726  000013                  CMP     #13,(SP)+       ;TO USE FOR GENERATING DATA WORDS
017324  001002                          BNE     15$             ;OF THE NEXT BLOCK
017326  062701  000004                  ADD     #4,R1

017332  005201                  15$:    INC     R1
017334  032777  000002  161656          BIT     #CSE,@RKER      ;IF THERE WAS A CSE THEN CHECK
017342  001403                          BEQ     16$             ;ONLY THOSE SECTORS THAT WERE READ
017344  020177  161660                  CMP     R1,@RKDA
017350  001407                          BEQ     17$
017352  010137  025664          16$:    MOV     R1,RSDTH
017356  010137  025662                  MOV     R1,RSDTL
017362  005137  025662                  COM     RSDTL
017366  000644                          BR      1$
017370  104415                  17$:    RESREG                  ;RESTORE R0-R5
017372  000207                          RTS     PC
```

```
                                        .SBTTL  ROUTINE TO SIZE MEMORY

                                ;;*************************************************************
                                ;*CALL:
                                ;*      JSR     PC,$SIZE
                                ;*      RETURN
                                ;*$LSTAD WILL CONTAIN:
                                ;*      WITH KT11 OPTION        -- LAST VIRTUAL ADDRESS OF THE LAST BANK
                                ;*      WITHOUT KT11 OPTION     -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
                                ;*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
                                ;*$KT11 IS THE MEMORY MANAGEMENT KEY
                                ;*BIT07 = 0 DON'T USE MEMORY MANAGEMENT
                                ;*      MUST BE SETUP BEFORE THE CALL
                                ;*BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
                                ;*      DETERMINED BY ROUTINE

017374  010046                  $SIZE:  MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
017376  010146                          MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
017400  010246                          MOV     R2,-(SP)        ;;SAVE R2 ON THE STACK
017402  010346                          MOV     R3,-(SP)        ;;SAVE R3 ON THE STACK
017404  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
017410  013746  000006                  MOV     @#ERRVEC+2,-(SP)
017414  010600                          MOV     SP,R0           ;;SAVE THE STACK POINTER
                                ;;SET THE ERRVEC PS TO THE PRESENT PS
017416  104400                          TRAP                    ;;PUSH OLD PSW AND PC ON STACK
017420  012637  000006                  MOV     (SP)+,@#ERRVEC+2        ;;SAVE THE PSW IN @#ERRVEC+2
017424  012701  003776                  MOV     #3776,R1        ;;SETUP ADDRESS
017430  105727                          TSTB    (PC)+           ;;USE MEMORY MANAGEMENT?
017432  000200                  $KT11:  .WORD   200             ;;SET TO USE MEMORY MANAGEMENT
017434  100062                          BPL     $CORE           ;;BR IF NO
017436  012737  017574  000004          MOV     #$KTNEX,@#ERRVEC ;;SET FOR TIMEOUT
017444  005737  177572                  TST     @#SR0           ;;KT11 ARE YOU THERE?
017450  052737  100000  017432          BIS     #100000,$KT11   ;;YES--SET KT11 KEY
017456  005046                          CLR     -(SP)           ;;INITIALIZE FOR "PAR" LOADING
017460  012702  172340                  MOV     #KIPAR0,R2      ;;ADDRESS OF FIRST "PAR"
017464  012703  000010                  MOV     #^D8,R3         ;;LOAD EIGHT "PAR,'S" AND EIGHT "PDR,'S"
017470  012762  077406  177740  1$:     MOV     #77406,-40(R2)  ;;PDR = 4K, UP, READ/WRITE
017476  011622                          MOV     (SP),(R2)+      ;;LOAD "PAR"
017500  062716  000200                  ADD     #200,(SP)       ;;UPDATE FOR NEXT "PAR"
017504  077307                          SOB     R3,1$           ;;LOOP UNTIL ALL EIGHT ARE LOADED
017506  012742  177600                  MOV     #177600,-(R2)   ;;SETUP KIPAR7 FOR I/O
017512  005042                          CLR     -(R2)           ;;SETUP KIPAR6 FOR TESTING
017514  012737  017532  000004          MOV     #2$,@#ERRVEC    ;;CATCH TIMEOUT IF NO SR3
017522  012737  000020  172516          MOV     #20,@#SR3       ;;ENABLE 22 BIT MODE
017530  000401                          BR      3$              ;;THIS PDP-11 HAS A SR3 REGISTER
017532  022626                  2$:     CMP     (SP)+,(SP)+     ;;CLEAN OFF THE STACK--NO SR3
017534  005237  177572          3$:     INC     @#SR0           ;;TURN ON MEMORY MANAGEMENT
017540  012737  017564  000004          MOV     #$KTOUT,@#ERRVEC ;;SET FOR TIME OUT
017546  005737  143776          4$:     TST     @#143776        ;;TRAP ON NON-EX-MEM
017552  062712  000040                  ADD     #40,(R2)        ;;MAKE A 1K STEP
017556  023712  172356                  CMP     @#KIPAR7,(R2)   ;;LAST ONE?
017562  101371                          BHI     4$              ;;NO--TRY IT
017564  011202                  $KTOUT: MOV     (R2),R2         ;;GET LAST BANK+1
017566  005037  177572                  CLR     @#SR0           ;;TURN OFF MEMORY MANAGEMENT
017572  000421                          BR      $SIZEX
017574  042737  100000  017432  $KTNEX: BIC     #100000,$KT11   ;;KT11 NON-EXISTENT
```

```
 017602  012737  017632  000004  $CORE:  MOV     #$CROUT,@#ERRVEC ;;SET FOR TIMEOUT
 017610  005002                          CLR     R2               ;;SET UP BANK
 017612  062701  004000          1$:     ADD     #4000,R1         ;;INCREMENT BY 1K
 017616  062702  000040                  ADD     #40,R2           ;;1K STEP
 017622  005711                          TST     (R1)             ;;TRAP ON TIME OUT
 017624  022701  177776                  CMP     #177776,R1       ;;LAST ONE
 017630  001370                          BNE     1$               ;;NO--TRY AGAIN
 017632  162701  004000          $CROUT: SUB     #4000,R1
 017636  162702  000040          $SIZEX: SUB     #40,R2           ;;DROP BACK
 017642  010006                          MOV     R0,SP            ;;RESTORE THE STACK
 017644  012637  000006                  MOV     (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
 017650  012637  000004                  MOV     (SP)+,@#ERRVEC
 017654  010137  017676                  MOV     R1,$LSTAD        ;;LAST ADDRESS
 017660  010237  017700                  MOV     R2,$LSTBK        ;;LAST BANK
 017664  012603                          MOV     (SP)+,R3         ;;RESTORE R3
 017666  012602                          MOV     (SP)+,R2         ;;RESTORE R2
 017670  012601                          MOV     (SP)+,R1         ;;RESTORE R1
 017672  012600                          MOV     (SP)+,R0         ;;RESTORE R0
 017674  000207                          RTS     PC
 017676  000000                  $LSTAD: .WORD   0                ;;CONTAINS THE LAST ADDRESS
 017700  000000                  $LSTBK: .WORD   0                ;;CONTAINS THE LAST BANK
```

```
                                 ;TYPDBO
                                 ;THIS ROUTINE CONVERTS A VIRTUAL ADDRESS TO PHYSICAL ADDRESS AND TYPES
                                 ;OUT THE 6 DIGIT PHYSICAL ADDRESS. R2 CONTAINS VIRTUAL ADDRESS AT THE TIME
                                 ;OF ENTRY.
                                 ;TYPE OUT IS INHIBITED IF SW 13 IS SET.

 017702  032777  020000  161230  TYPDBO: BIT     #SW13,@SWR       ;INHIBIT TYPEOUT?
 017710  001042                          BNE     2$               ;YES
 017712  010346                          MOV     R3,-(SP)
 017714  010446                          MOV     R4,-(SP)
 017716  010546                          MOV     R5,-(SP)

 017720  010246                          MOV     R2,-(SP)         ;PUSH VA ON STACK
 017722  004737  022076                  JSR     PC,CROTLF        ;ROTATE BITS 15,14,13 INTO 2,1,0
 017726  012603                          MOV     (SP)+,R3         ;FORM OFFSET TO BE USED
 017730  006303                          ASL     R3               ;FOR KIPAR

 017732  016304  172340                  MOV     KIPAR0(R3),R4    ;GET THE BASE PAGE ADDRESS FROM
                                                                  ;KIPAR
 017736  005003                          CLR     R3               ;ROTATE LEFT 6 TIMES (MULTIPLY
 017740  012705  177772                  MOV     #-6,R5           ;BY 100 OCTAL) TO GET THE
 017744  006104                  1$:     ROL     R4               ;BASE BUS ADDRESS (PHYSICAL)
 017746  005205                          INC     R5               ;R3 CONTAINS MSB-2 BITS
 017750  001375                          BNE     1$


 017752  010246                          MOV     R2,-(SP)         ;STRIP OFF TOP 3 BITS FROM  VA &

 017754  042716  160000                  BIC     #160000,(SP)     ;GET THE OFFSET INSIDE THE PAGE
 017760  062604                          ADD     (SP)+,R4         ;FORM THE ENTIRE PHYSICAL
 017762  005503                          ADC     R3               ;ADDRESS, R4 CONTAINS LOWER 16 BITS
                                                                  ;R3 CONTAINS TOP 2 BITS
 017764  010437  001162                  MOV     R4,$REG0         ;SAVE LOWER 16 BITS OF PA
 017770  010337  001164                  MOV     R3,$REG1         ;SAVE TOP 2 BITS OF PA
 017774  012746  001162                  MOV     #$REG0,-(SP)     ;PUSH POINTER TO PA ON STACK
 020000  004737  024372                  JSR     PC,@#$DB20       ;CONVERT THE 18 BIT BINARY
                                                                  ;ADDRESSS TO OCTAL ASCIZ NUMBERS ON RETURN
                                                                  ;POINTER TO THE FIRST ASCIZ CHARACTERS
                                                                  ;IS ON STACK
 020004  004737  024722                  JSR     PC,@#SUPRS       ;TYPE OUT THE OCTAL 6 DIGIT
                                                                  ;PHYSICAL ADDRESS.

 020010  012605                          MOV     (SP)+,R5
 020012  012604                          MOV     (SP)+,R4
 020014  012603                          MOV     (SP)+,R3

 020016  000207                  2$:     RTS     PC
```

```
                                        ;CHKCS
                                        ;THIS ROUTINE CHECKS IF BIT 15 OF RKCS WAS SET. IF IT WAS RETURN IS MADE TO
                                        ;THE ERROR MESSAGE FOLLOWING THE JSR CALL. IF NOT, THE ERROR MADE TO SKIP
                                        ;OVER THE ERROR MESSAGE.

 020020  005777  161176                 CHKCS:  TST     @RKCS           ;BIT 15 SET?
 020024  100073                                 BPL     COMRET          ;NO
 020026  004737  021740                         JSR     PC,GT4RG        ;YES, GET RKCS, ER, DS, DA
 020032  000207                                 RTS     PC              ;RETURN TO THE ERROR MESSAGE

                                        ;CHKDA
                                        ;THIS ROUTINE CHECKS IF RKDA INCREMENTED CORRECTLY. IF NOT, RETURN IS MADE
                                        ;TO THE ERROR MESSAGE FOLLOWING THE JSR CALL. IF YES, RETURN IS MADE TO
                                        ;SKIP OVER THE ERROR MESSAGE.
                                        ;AT THE TIME OF ENTRY, R2 CONTAINS THE EXPECTED RKDA.

 020034  020277  161170                 CHKDA:  CMP     R2,@RKDA        ;DID RKDA INCREMENT CORRECTLY?
 020040  001465                                 BEQ     COMRET          ;YES
 020042  010237  001162                         MOV     R2,$REG0        ;GET EXPCTD RKDA
 020046  017737  161156  001164                 MOV     @RKDA,$REG1     ;GET RKDA RECVD
 020054  000207                                 RTS     PC              ;RETURN TO THE ERROR MESSAGE

                                        ;CHKBA
                                        ;THIS ROUTINE CHECKS IF RKBA INCREMENTED CORRECTLY. IF NOT, RETURN IS MADE
                                        ;TO THE ERROR MESSAGE FOLLOWING THE JSR CALL. IF YES, RETURN IS MADE TO
                                        ;SKIP OVER THE ERROR MESSAGE.
                                        ;AT THE TIME OF ENTRY, R3 CONTAINS THE WORD COUNT (# OF WORDS TRANSFERRED)
                                        ;R4 CONTAINS THE BUS ADDRESS WHERE THE TRANSFER STARTED.

 020056  000241                         CHKBA:  CLC
 020060  006103                                 ROL     R3              ;FORM THE EXPCTD BUS ADDRESS
 020062  060304                                 ADD     R3,R4
 020064  000241                                 CLC
 020066  006003                                 ROR     R3
 020070  020477  161132                         CMP     R4,@RKBA        ;DID RKBA INCREMENT CORRECTLY?
 020074  001447                                 BEQ     COMRET          ;YES
 020076  010437  001162                         MOV     R4,$REG0        ;GET EXPCTD RKBA
 020102  000207                                 RTS     PC              ;RETURN TO THE EROR MESAGE

 020104  017737  161116  001164                 MOV     @RKBA,$REG1     ;GET RKBA RECVD
                                        ;CHKMEX
                                        ;THIS ROUTINE CHECKS THAT RKBA OVERFLOWED AND MEX BIT WAS SETIN RKCS (BIT 4)
                                        ;IF RKBA OVERFLOWED CORRECTLY, THE RETURN ADDRESS IS ADJUSTED TO SKIP THE
                                        ;ERROR MESSAGE ON RETURN.

 020112  017746  161104                 CHKMEX: MOV     @RKCS,-(SP)     ;GET RKCS
 020116  042716  177717                         BIC     #177717,(SP)    ;GET MEX BITS 4,5
 020122  022726  000020                         CMP     #BIT4,(SP)+     ;CHECK BIT 4 SET?
 020126  001432                                 BEQ     COMRET          ;YES, OK
 020130  004737  021740                         JSR     PC,GT4RG        ;SAVE RKCS,ER,DS,DA
 020134  000207                                 RTS     PC              ;RETURN
```

```
                                        ;CHKWC
                                        ;THIS ROUTINE CHECKS IF RKWC OVERFLOWED CORRECTLY AFTER A DATA TRANSFER
                                        ;IF IT DID NOT, RETURN IS MADE TO THE ERROR MESSAGE. IF IT DID, RETURN IS
                                        ;MADE TO SKIP OVER THE ERROR MESSAGE.

 020136  005777  161062                 CHKWC:  TST     @RKWC           ;RKWC OVERFLOWED?
 020142  001424                                 BEQ     COMRET          ;YES
 020144  017737  161060  001162                 MOV     @RKDA,$REG0
 020152  017737  161046  001164                 MOV     @RKWC,$REG1
 020160  000207                                 RTS     PC              ;RETURN TO THE EROR MESAGE



                                        ;CHKRWS
                                        ;THIS ROUTINE CHECKS IF R/W/S RDY BIT IN RKDS IS SET. IF IT IS NOT SET RETURN
                                        ;IS MADE TO THE ERROR MESSAGE FOLLOWING THE JSR CALL. IF IT IS, THE RETURN
                                        ;ADDRESS IS ADJUSTED TO SKIP OVER THE ERROR MESSAGE ON RERTURN.

 020162  032777  000100  161026         CHKRWS: BIT     #RWS,@RKDS      ;RWS RDY SET?
 020170  001011                                 BNE     COMRET          ;YES
 020172  004737  021740                         JSR     PC,GT4RG        ;GET RKCS, ER, DS, DA
 020176  000207                                 RTS     PC              ;RETURN TO THE ERROR MESSAGE


                                        ;CHKCRDY
                                        ;THIS ROUTINE CHECKS IF CONTROL READY BIT IN RKCS IS SET. IF IT IS NOT,
                                        ;RETURN IS MADE TO THE ERROR MESSAGE FOLLOWING THE JSR CALL. IF IT IS,
                                        ;RETURN ADDRESS IS ADJUSTED TO SKIP OVER THE ERROR MESSAGE.

 020200  105777  161016                 CHKCRDY: TSTB   @RKCS           ;CONTROL READY SET?
 020204  100403                                 BMI     COMRET          ;YES
 020206  004737  021740                         JSR     PC,GT4RG        ;GET RKCS, ER, DS, DA
 020212  000207                                 RTS     PC              ;RETURN TO THE EROR MESAGE

 020214  062716  000002                 COMRET: ADD     #2,(SP)         ;ADJUST RETURN ADDRESS TO SKIP OVER MESAGE
 020220  000207                                 RTS     PC
```

```
                                        ;THIS ROUTINE KEEPS A HISTORY OF THE COOMANDS THAT ARE BEING EXECUTED
                                        ;ON THE RK11. 'PRSFNC' CONTAINS INFORMATION ABOUT THE PRESENT COMMAND
                                        ;WHICH IS ABOUT TO BE INITIATED. 'PSTFNC' CONTAINS INFORMATION ABOUT THE
                                        ;COMMAND THAT WAS EXECUTED BEFORE THIS NEW ONE. THERE ARE MULTIPLE POINTS
                                        ;OF ENTRY DEPENDING ON THE TYPE OF COMMAND BEING PRESENTLY INITIATED:

                                        ;DRCMND - ENTERED WHEN A DRIVE RESET IS BEING INITIATED. DRIVE # IS SAVED
                                        ;IN BITS 0-2 OF 'PRSCMND' AND BIT 8 IS SET.

                                        ;CRCMND - ENTERED WHEN A CONTROL RESET IS BEING INITIATED. BIT 14 OF
                                        ;'PRSCMND' IS SET.

                                        ;POSCMND - ENTERED WHEN A POSITIONING SEEK IS BEING INITIATED. BITS 0-2
                                        ;CONTAIN THE DRIVE NUMBER ON WHICH THE POSITIONING SEEK WAS DONE. ALSO
                                        ;BIT 7 IS SET.

                                        ;IN ALL ABOVE CASES BIT 15 OF 'PRSCMND' IS SET.

                                        ;FNCMND - ENTERED WHEN A COMMAND OTHER THAN ANY ONE OF THE ABOVE IS BEING
                                        ;INITIATED (EX: READ, WRITE, ETC).
                                        ;THE OFFSET TO THE COMMAND KEY (BASE=KEY) IS SAVED IN BITS 0-3 OF 'PSRCMND'.

                                        ;IT SHOULD BE NOTED THAT CONTENTS OF 'PRSFNC' ARE PUSHED INTO 'PSTFNC'
                                        ;AND SAVED, BEFORE PUTTING INFO ABOUT THE PRESENT COMMAND IN 'PRSFNC'.

                                        ;R0 CONTAINS ADDRESS OF THE COMMAND KEY, AT THE TIME OF ENTRY.

 020222  012737  100400  001512  DRCMND: MOV     #BIT15+BIT8,QFNC
 020230  017746  160774                  MOV     @RKDA,-(SP)             ;SAVE DRIVE #
 020234  004737  022076                  JSR     PC,CROTLF
 020240  052637  001512                  BIS     (SP)+,QFNC
 020244  000424                          BR      P2

 020246  012737  140000  001512  CRCMND: MOV     #BIT15+BIT14,QFNC
 020254  000420                          BR      P2

 020256  011037  001512          POSCMND:        MOV     (R0),QFNC
 020262  042737  177770  001512          BIC     #177770,QFNC            ;GET DRIVE NO.
 020270  052737  100200  001512          BIS     #BIT15+BIT7,QFNC
 020276  000407                          BR      P2

 020300  005037  001512          FNCMND: CLR     QFNC
 020304  010046                  P1:     MOV     R0,-(SP)
 020306  162716  001306                  SUB     #KEY,(SP)
 020312  052637  001512                  BIS     (SP)+,QFNC

 020316  013737  001462  001464  P2:     MOV     PRSFNC,PSTFNC
 020324  013737  001512  001462          MOV     QFNC,PRSFNC
 020332  000207                          RTS     PC
```

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

 020334  010046                  HISTRY: MOV     R0,-(SP)
 020336  010146                          MOV     R1,-(SP)

 020340  012700  001462                  MOV     #PRSFNC,R0
 020344  104401  020654                  TYPE    ,MH1
 020350  104401  020700                  TYPE    ,MH3
 020354  104401  020670                  TYPE    ,MH2
 020360  005710                  6$:     TST     (R0)
 020362  100053                          BPL     3$              ;READ, READ CHECK, WRITE, WRITE CHECK, SEEK
 020364  105710                          TSTB    (R0)
 020366  100427                          BMI     2$              ;POSITIONING (SEEK)
 020370  032710  040000                  BIT     #BIT14,(R0)
 020374  001014                          BNE     1$              ;CONTROL RESET

 020376  104401  020404                  TYPE    ,65$            ;;TYPE ASCIZ STRING
 020402  000410                          BR      64$             ;;GET OVER THE ASCIZ
                                 ;;65$:   .ASCIZ  /DRESET ON DRV /
 020424                          64$:
 020424  000425                          BR      7$

 020426                          1$:
 020426  104401  020434                  TYPE    ,67$            ;;TYPE ASCIZ STRING
 020432  000404                          BR      66$             ;;GET OVER THE ASCIZ
                                 ;;67$:   .ASCIZ  /CRESET/
 020444                          66$:
 020444  000463                          BR      4$

 020446                          2$:
 020446  104401  020454                  TYPE    ,69$            ;;TYPE ASCIZ STRING
 020452  000412                          BR      68$             ;;GET OVER THE ASCIZ
                                 ;;69$:   .ASCIZ  /POSITIONING DRIVE /
 020500                          68$:
 020500  011046                  7$:     MOV     (R0),-(SP)
 020502  042716  177770                  BIC     #177770,(SP)    ;TYPE DRIVE NO.
 020506  104402                          TYPOC
 020510  000441                          BR      4$

 020512  011001                  3$:     MOV     (R0),R1
 020514  016101  002032                  MOV     PCMND(R1),R1    ;GO TYPE OUT THE FUNCTION
 020520  016104  000002                  MOV     2(R1),R4        ;BEING PERFORMED
 020524  004737  021644                  JSR     PC,TYPFN
 020530  104401  020536                  TYPE    ,71$            ;;TYPE ASCIZ STRING
 020534  000403                          BR      70$             ;;GET OVER THE ASCIZ
                                 ;;71$:   .ASCIZ  <15><12>/DA=/
 020544                          70$:
 020544  011146                          MOV     (R1),-(SP)      ;TYPE OUT DISK ADDRESS
 020546  104402                          TYPOC
 020550  104401  020556                  TYPE    ,73$            ;;TYPE ASCIZ STRING
 020554  000403                          BR      72$             ;;GET OVER THE ASCIZ
```

```
                                      ;;73$:  .ASCIZ  /  BA=/
020564                                72$:
020564  016146  000006                        MOV     6(R1),-(SP)
020570  104402                                TYPOC
020572  104401  020600                        TYPE    ,75$            ;;TYPE ASCIZ STRING
020576  000403                                BR      74$             ;;GET OVER THE ASCIZ
                                      ;;75$:  .ASCIZ  /  WC=/
020606                                74$:
020606  016146  000004                        MOV     4(R1),-(SP)
020612  104402                                TYPOC

020614  020027  001464                4$:     CMP     R0,#PSTFNC
020620  001410                                BEQ     5$
020622  005720                                TST     (R0)+
020624  104401  020654                        TYPE    ,MH1
020630  104401  020703                        TYPE    ,MH4
020634  104401  020670                        TYPE    ,MH2
020640  000647                                BR      6$
020642  104401  001213                5$:     TYPE    ,$CRLF
020646  012601                                MOV     (SP)+,R1
020650  012600                                MOV     (SP)+,R0
020652  000207                                RTS     PC
020654  005015  052506  041516        MH1:    .ASCIZ  <15><12>/FUNCTION /
020662  044524  047117  000040
020670  042440  051122  051117        MH2:    .ASCIZ  / ERROR /
020676  000040
020700  052101     000                MH3:    .ASCIZ  /AT/
020703     120  044522  051117        MH4:    .ASCIZ  /PRIOR TO/
020710  052040  000117
                                              .EVEN
```

```
                                      ;STATSTC
                                      ;AT THE TIME OF ENTRY R1 CONTAINS THE DRIVE NUMBER FOR WHICH THE STATISTIC
                                      ;IS TO BE OBTAINED.  R5 CONTAINS THE POINTER TO THE PARAMETER TABLE, FOR
                                      ;THE COMMAND EXECUTED ON THE ABOVE DRIVE. R4 CONTAINS THE FUNCTION CODE
                                      ;(WRITE, READ, ETC) FOR WHICH STATISTICS  ARE TO BE TAKEN.

020714  010046                        STATSTC:MOV     R0,-(SP)        ;PUSH R0, R2, R3 ONTO THE
020716  010246                                MOV     R2,-(SP)        ;STACK
020720  010346                                MOV     R3,-(SP)

020722  005002                                CLR     R2
020724  005701                                TST     R1              ;DRIVE 0?
020726  001404                                BEQ     2$              ;FORM THE OFFSET FOR THE
020730  062702  000004                1$:     ADD     #4,R2           ;'WORDS XFERRED COUNTS'-
020734  005301                                DEC     R1              ;NWRTL, NRDL
020736  001374                                BNE     1$
020740  016500  000004                2$:     MOV     4(R5),R0        ;GET WORD COUNT (RKWC) FROM
020744  005400                                NEG     R0              ;THE PARAMETER TABLE

020746  005777  160250                        TST     @RKCS           ;ANY ERROR DURING THE XFER?
020752  100004                                BPL     3$
                                                                      ;YES,
020754  017703  160244                        MOV     @RKWC,R3        ;GET THE # OF WORDS THAT
020760  005403                                NEG     R3              ;WERE ACTUALLY X-FERRED
020762  160300                                SUB     R3,R0

020764  022704  000002                3$:     CMP     #2,R4           ;WRITE FUNCTION?
020770  001005                                BNE     5$
                                                                      ;YES, ADD THE # OF WORDS
020772  060062  001732                4$:     ADD     R0,NWRTL(R2)    ;XFERRED (WRITE)
020776  005562  001734                        ADC     NWRTH(R2)       ;NOTE IT'S 2-WORD COUNT LO, HI
021002  000404                                BR      6$

021004  060062  001772                5$:     ADD     R0,NRDL(R2)     ;ADD THE # OF WORDS READ
                                                                      ;NOTE THAT WRT CHK,
                                                                      ;READ CHK ARE ALSO CONS-
                                                                      ;IDERED TO BE 'READ'
021010  005562  001774                        ADC     NRDH(R2)        ;CARRY OVER TO THE HI WORD

021014  012603                        6$:     MOV     (SP)+,R3        ;POP R3,R2,R0 FROM THE STACK
021016  012602                                MOV     (SP)+,R2
021020  012600                                MOV     (SP)+,R0
021022  000207                                RTS     PC
```

```
                                ;REPSTAT
                                ;THIS ROUTINE REPORTS ERROR STATISTICS AND DATA-TRANSFER STATISTICS.

021024  104401  002377          REPSTA: TYPE    ,MSG26
021030  013700  001264                  MOV     DRVPRS,R0
021034  012701  001254                  MOV     #PDR,R1

021040  104401  001213          1$:     TYPE    ,$CRLF
021044  112102                          MOVB    (R1)+,R2
021046  042702  177770                  BIC     #177770,R2
021052  010246                          MOV     R2,-(SP)
021054  104403                          TYPOS
021056     003                          .BYTE   3
021057     000                          .BYTE   0
021060  104401  002662                  TYPE    ,BLNKS3

021064  005004                          CLR     R4
021066  010203                          MOV     R2,R3
021070  001404                          BEQ     3$
021072  062704  000004          2$:     ADD     #4,R4
021076  005303                          DEC     R3
021100  001374                          BNE     2$

021102  104401  002664          3$:     TYPE    ,BLNKS1
021106  010446                          MOV     R4,-(SP)
021110  062716  001732                  ADD     #NWRTL,(SP)
021114  004737  024512                  JSR     PC,$DB2D
021120  004737  024722                  JSR     PC,SUPRS
021124  104401  002664                  TYPE    ,BLNKS1
021130  010446                          MOV     R4,-(SP)
021132  062716  001772                  ADD     #NRDL,(SP)
021136  004737  024512                  JSR     PC,$DB2D
021142  004737  024722                  JSR     PC,SUPRS

021146  006302                          ASL     R2

021150  104401  002664                  TYPE    ,BLNKS1
021154  016246  001652                  MOV     CSECN(R2),-(SP)
021160  104405                          TYPDS

021162  104401  002664                  TYPE    ,BLNKS1
021166  016246  001632                  MOV     WCECN(R2),-(SP)
021172  104405                          TYPDS

021174  104401  002664                  TYPE    ,BLNKS1
021200  016246  001712                  MOV     DATER(R2),-(SP)
021204  042716  100000                  BIC     #100000,(SP)    ;DONT TYPE A NEGATIVE NO.
021210  104405                          TYPDS

021212  104401  002664                  TYPE    ,BLNKS1
021216  016246  001562                  MOV     HECN(R2),-(SP)
021222  104405                          TYPDS

021224  005300                          DEC     R0              ;FINISHED WITH THE DRIVES ?
021226  001304                          BNE     1$              ;BR IF NOT
021230  104401  002474                  TYPE    ,MSG26A         ;REST OF SUMMARY MESSAGE
```

```
021234  013700  001264                  MOV     DRVPRS,R0       ;NUMBER OF DRIVES
021240  012701  001254                  MOV     #PDR,R1         ;'DRIVES PRESENT' TABLE ADDRESS
021244  104401  001213          4$:     TYPE    ,$CRLF          ;CR-LF
021250  112102                          MOVB    (R1)+,R2        ;DRIVE ADDRESS
021252  042702  177770                  BIC     #177770,R2      ;LEAVE ONLY DRIVE NUMBER
021256  010246                          MOV     R2,-(SP)        ;PUT ON STACK FOR TYPEOUT
021260  104403                          TYPOS                   ;TYPE IT IN OCTAL
021262     003                          .BYTE   3               ;TYPE 3 CHARACTERS
021263     000                          .BYTE   0               ;SUPRESS LEADING ZEROS
021264  104401  002662                  TYPE    ,BLNKS3         ;3 BLANKS
021270  006302                          ASL     R2              ;CONVERT TO A WORD TABLE INDEX
021272  104401  002664                  TYPE    ,BLNKS1
021276  016246  001602                  MOV     SKECN(R2),-(SP)
021302  104405                          TYPDS


021304  104401  002664                  TYPE    ,BLNKS1
021310  016246  001672                  MOV     ABORT(R2),-(SP)
021314  104405                          TYPDS

021316  006202                          ASR     R2
021320  104401  002664                  TYPE    ,BLNKS1
021324  116246  001622                  MOVB    SINCN(R2),-(SP)
021330  104405                          TYPDS

021332  005300                          DEC     R0
021334  001343                          BNE     4$
021336  004737  026556                  JSR     PC,TIMTYP       ;TYPE THE TIME
021342  000207                          RTS     PC
```

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

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

021344  005046                          STATUS: CLR     -(SP)           ;DROP PRIORITY AND WAIT FOR INT
021346  012746  021354                          MOV     #1$,-(SP)       ;RETURN FOR RTI
021352  000002                                  RTI
                                                                        ;NOTE THAT THE INTERRUPTS ARE ALLOWED ONLY
                                                                        ;AT CERTAIN PLACES IN THE PROGRAM, BECAUSE
                                                                        ;IT MAKES TROUBLESHOOTING OF FALIURES EASY.
                                                                        ;OTHER PLACES WHERE INTERRUPTS ARE ALLOWED
                                                                        ;TO TAKE PLACE:
                                                                        ;'CHFAFN',  FIRST INTERRUPT AFTER ISSUING
                                                                        ;A SEEK FUNCTION.

021354  005237  001460                  1$:     INC     QSCNT
021360  001456                                  BEQ     QEROR
021362  105777  157634                          TSTB    @RKCS
021366  100514                                  BMI     CNOBSY

021370  005037  001466                          CLR     CICNT
021374  012737  177761  001470                  MOV     #-17,CICNT1

021402  105737  001534                  CBSY:   TSTB    INTFLG          ;FOR A NON-SEEK COMAND:
                                                                        ;INTFLG- BIT 7 IS SET, BITS 0-3 CONTAIN
                                                                        ;OFFSET TO THE COMMAND KEY (FROM KEY),
                                                                        ;FOR WHICH THIS INTERUPT IS EXPCTD.
                                                                        ;WHEN THE INTERUPT OCCURS & 'INTHND' IS
                                                                        ;ENTERED 'INTFLG' IS CLEARED.
021406  001507                                  BEQ     SEXIT
021410  005237  001466                          INC     CICNT
021414  001372                                  BNE     CBSY
021416  005237  001470                          INC     CICNT1
021422  001367                                  BNE     CBSY

                                                                        ;TIMED OUT WHILE WAITING FOR THE INTRUPT.
                                                                        ;ONE OF THE COMMANDS DID NOT INTERRUPT
021424  113700  001534                  NIEROR: MOVB    INTFLG,R0
021430  042700  177760                          BIC     #177760,R0
021434  010003                                  MOV     R0,R3
021436  062700  001306                          ADD     #KEY,R0
021442  011037  001172                          MOV     (R0),SREG4
021446  042737  177770  001172                  BIC     #177770,SREG4
```

```
021454  013737  001172  001250                  MOV     SREG4,SRDRV     ;GET DRIVE #, FOR TYPING SERIAL #

021462  104421  002245                          TYPMSG  ,MSG15          ;PRINT 'DRVE # DIDN'T INTRUPT AFTER'
021466  016305  002032                          MOV     PCMND(R3),R5
021472  016504  000002                          MOV     2(R5),R4
021476  004737  021644                          JSR     PC,TYPFN
021502  004737  021740                          JSR     PC,GT4RG
021506  104025                                  ERROR   25              ;COMMAND TYPED OUT IN EROR MESAGE DID
                                                                        ;NOT INTERUPT ON COMPLETION.
021510  052710  104000                          BIS     #BIT15+BIT11,(R0)       ;INDICATE THAT FUNCTION IS ABORTED
021514  000444                                  BR      SEXIT


021516  005037  001460                  QEROR:  CLR     QSCNT           ;REESTABLISH COUNT
021522  004737  021740                          JSR     PC,GT4RG
021526  104026                                  ERROR   26              ;ALL 8 COMMANDS SHOULD BE DONE BY NOW, TIMED
                                                                        ;OUT. THE PROGRAM IS WAITING FOR ONE OF THE
                                                                        ;COMMANDS IN THE Q TO BE FINISHED AND THIS
                                                                        ;DID NOT HAPPEN OR FOR SOME OTHER REASON THE
                                                                        ;'FINISHED' FLAG (BIT 15) OF ONE OF THE 8
                                                                        ;COMMAND KEYS WAS NOT SET. VARIOUS FLAGS 'POS'(-7)
                                                                        ;'BUSY'(-7), 'KEY'(-8) CONTAIN INFORMATION
                                                                        ;ABOUT THE STATUS OF THE SYSTEM.

021530  032777  020000  157402                  BIT     #SW13,@SWR      ;INHIBIT TYPEOUT?
021536  001024                                  BNE     2$              ;YES
021540  104401  002305                          TYPE,   MSG16
021544  012700  001306                          MOV     #KEY,R0
021550  012701  001426                          MOV     #BUSY,R1
021554  012702  177770                          MOV     #-10,R2
021560  104401  001213                  1$:     TYPE    ,$CRLF
021564  012046                                  MOV     (R0)+,-(SP)     ;TYPE OUT CONTENTS OF ALL KEYS
021566  104402                                  TYPOC                   ;KEY-KEY8
021570  104401  002662                          TYPE    ,BLNKS3
021574  005046                                  CLR     -(SP)
021576  112116                                  MOVB    (R1)+,(SP)      ;TYPE OUT CONTENTS OF ALL BUSY FLAGS
021600  104403                                  TYPOS                   ;BUSY-BUSY7
021602     003                                  .BYTE   3
021603     000                                  .BYTE   0
021604  005202                                  INC     R2              ;DONE?
021606  001364                                  BNE     1$              ;NO

021610  004737  015714                  2$:     JSR     PC,CLRERR       ;MAKE SURE THERE IS NO HEAD MOVEMENT ON
                                                                        ;ANY DRIVE & THEN DO CONTROL RESET
021614  000137  010536                          JMP     BEGNEX          ;GO, BAK AND CONTINUE

021620  005004                          CNOBSY: CLR     R4
021622  005204                                  INC     R4
021624  001376                                  BNE     .-2
021626  013746  001244                  SEXIT:  MOV     PPRLVL,-(SP)
021632  012746  021640                          MOV     #RTIPC7,-(SP)   ;RETURN FOR RTI ***********
021636  000002                                  RTI

021640  000137  010556                  RTIPC7: JMP     QMNGER
```

```
                                        ;TYPFN
                                        ;ROUTINE TO TYPE OUT THE FUNCTION (READ,WRITE,ETC) ;R4 CONTAINS THE
                                        ;FUNCTION CODE AT THE TIME OF ENTRY.
                                        ;SW 13, IF SET INHIBITS TYPEOUT.

 021644  032777  020000  157266  TYPFN:  BIT     #SW13,@SWR      ;INHIBIT TYPEOUT?
 021652  001031                          BNE     5$              ;YES
 021654  020427  000002                  CMP     R4,#2           ;WRITE?
 021660  001002                          BNE     1$
 021662  104401  002133                  TYPE    ,MSG6
 021666  022704  000004          1$:     CMP     #4,R4           ;READ?
 021672  001002                          BNE     2$
 021674  104401  002141                  TYPE    ,MSG7
 021700  022704  000012          2$:     CMP     #12,R4          ;READ CHECK?
 021704  001002                          BNE     3$
 021706  104401  002156                  TYPE    ,MSG9
 021712  022704  000006          3$:     CMP     #6,R4           ;WRITE CHECK?
 021716  001002                          BNE     4$
 021720  104401  002146                  TYPE    ,MSG8
 021724  022704  000010          4$:     CMP     #10,R4          ;SEEK?
 021730  001002                          BNE     5$
 021732  104401  002201                  TYPE    ,MSG11
 021736  000207                  5$:     RTS     PC
```

```
                                        ;GT4RG
                                        ;GET CONTENTS OF RKCS, RKER, RKDS, RKDAA

 021740  017737  157264  001170  GT4RG:  MOV     @RKDA,$REG3
 021746  017737  157250  001162  GT3RG:  MOV     @RKCS,$REG0
 021754  017737  157240  001164          MOV     @RKER,$REG1
 021762  017737  157230  001166          MOV     @RKDS,$REG2
 021770  000207                          RTS     PC


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

 021772  004737  021746          GETINF: JSR     PC,GT3RG
 021776  010046                          MOV     R0,-(SP)
 022000  010146                          MOV     R1,-(SP)
 022002  010246                          MOV     R2,-(SP)
 022004  012700  001200                  MOV     #$REG6+2,R0
 022010  017701  157214                  MOV     @RKDA,R1
 022014  010102                          MOV     R1,R2
 022016  042702  177760                  BIC     #177760,R2
 022022  010240                          MOV     R2,-(R0)
 022024  006201                          ASR     R1
 022026  006201                          ASR     R1
 022030  006201                          ASR     R1
 022032  006201                          ASR     R1
 022034  010102                          MOV     R1,R2
 022036  042702  177776                  BIC     #177776,R2
 022042  010240                          MOV     R2,-(R0)
 022044  006201                          ASR     R1
 022046  010102                          MOV     R1,R2
 022050  042702  177400                  BIC     #177400,R2
 022054  010240                          MOV     R2,-(R0)
 022056  000301                          SWAB    R1
 022060  042701  177770                  BIC     #177770,R1
 022064  010140                          MOV     R1,-(R0)
 022066  012602                          MOV     (SP)+,R2
 022070  012601                          MOV     (SP)+,R1
 022072  012600                          MOV     (SP)+,R0
 022074  000207                          RTS     PC
```

```
                                 ;CROTLF
                                 ;CALL:  MOV     #NO,-(SP)        ;PUSH NO. TO BE ROTATED ON STACK
                                 ;       JSR     PC,CROTLF
                                 ;THIS ROUTINE ROTATES BITS 15, 14, 13 OF A WORD INTO BITS 2, 1, 0. THE
                                 ;REST OF THE BITS OF THE ROTATED WORD ARE CLEARED.


022076  042766  017777  000002   CROTLF: BIC     #17777,2(SP)
022104  000241                           CLC
022106  006166  000002                   ROL     2(SP)
022112  006166  000002                   ROL     2(SP)
022116  006166  000002                   ROL     2(SP)
022122  006166  000002                   ROL     2(SP)
022126  000207                           RTS     PC


                                 ;RG4SDRV
                                 ;CALL:  JSR     PC,RG4SDRV
                                 ;THIS ROUTINE GETS THE CONTENTS OF RKDS, RKER, RKCS, RKDA. THEN
                                 ;IT SAVES THE DRIVE NUMBER FROM RKDA IN 'SRDRV'.
022130  004737  021740           RG4SDR: JSR     PC,GT4RG         ;GET RKCS, ER, DS, DA


                                 ;GTSDRV
                                 ;CALL:  JSR     PC,GTSDRV
                                 ;THIS ROUTINE EXTRACTS THE DRIVE # FROM RKDA (BITS 15,14,13) AND SAVES
                                 ;IT IN "SRDRV" (BITS 0,1,2)

022134  017746  157070           GTSDRV: MOV     @RKDA,-(SP)      ;GET BITS 15,14,13 FROM RKDA
022140  004737  022076                   JSR     PC,CROTLF
022144  012637  001250                   MOV     (SP)+,SRDRV      ;SAVE THE DRIVE #
022150  000207                           RTS     PC
```

```
                                         .SBTTL  DRV.RESET - DRIVE RESET ROUTINE
                                 ;DRV.RESET - DRIVE RESET ROUTINE
                                 ;IF R/W/S RDY DOES NOT SET WITHIN A CERTAIN TIME OF DOING DRIVE RESET
                                 ;AN ERROR IS REPORTED.

022152  005037  022262           DR.RST: CLR     TIMOUT
022156  013777  001502  157044           MOV     QDRV,@RKDA
022164  012777  000015  157030           MOV     #15,@RKCS
022172  104417                           CON.RDY
022174  032777  000100  157014   1$:     BIT     #100,@RKDS       ;DID R/W/S RDY SET?
022202  001026                           BNE     2$               ;YES
022204  012746  177760                   MOV     #-20,-(SP)       ;NO, WAIT FOR R/W/S
022210  005216                           INC     (SP)
022212  001376                           BNE     .-2
022214  005726                           TST     (SP)+
022216  005237  022262                   INC     TIMOUT
022222  001364                           BNE     1$
022224  032777  020000  156706           BIT     #SW13,@SWR       ;INHIBIT TYPEOUT?
022232  001012                           BNE     2$               ;YES
022234  104401  001213                   TYPE    ,$CRLF           ;TIMED OUT, R/W/S RDY DID NOT SET
022240  104401  027644                   TYPE    ,EM4             ;REPORT ERROR
022244  104401  002206                   TYPE    ,MSG12
022250  011646                           MOV     (SP),-(SP)
022252  162716  000002                   SUB     #2,(SP)
022256  104402                           TYPOC
022260  000002                   2$:     RTI
022262  000000                   TIMOUT: 0
```

```
                                     .SBTTL  CON.RESET - CONTROL RESET ROUTINE
                             ;CON.RESET
                             ;CONTROL RESET ROUTINE
                             ;CON.RDY
                             ;CONTROL READY ROUTINE

022264  012777  000001  156730  CN.RST: MOV     #1,@RKCS
022272  005037  001472          CN.RDY: CLR     TIMER
022276  105777  156720          1$:     TSTB    @RKCS           ;DID CONTROL RDY SET?
022302  100451                          BMI     2$              ;YES
022304  012746  177750                  MOV     #-30,-(SP)      ;WAIT FOR CNTRL RDY
022310  005216                          INC     (SP)
022312  001376                          BNE     .-2
022314  005726                          TST     (SP)+
022316  005237  001472                  INC     TIMER
022322  001365                          BNE     1$
022324  032777  020000  156606          BIT     #SW13,@SWR      ;INHIBIT TYPEOUT?
022332  001035                          BNE     2$              ;YES
022334  104401  002206                  TYPE    ,MSG12          ;CNTRL RDY DID NOT SET, REPORT ERROR
022340  011646                          MOV     (SP),-(SP)
022342  162716  000002                  SUB     #2,(SP)
022346  104402                          TYPOC
022350  104401  022356                  TYPE    ,65$            ;;TYPE ASCIZ STRING
022354  000421                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/CONTROLLER NOT READY -   RKCS=/
022420                          64$:
022420  017746  156576                  MOV     @RKCS,-(SP)
022424  104402                          TYPOC
022426  000002                  2$:     RTI
```

```
                                        .SBTTL  TYPMSG - TYPE MESSAGE ROUTINE (SW13)
                                ;TYPMSG
                                ;THIS ROUTINE IS USED FOR MESSAGE TYPEOUTS. IF SW 13 IS SET THE TYPEOUT
                                ;IS SKIPPED.
                                ;CALL:  TYPMSG
                                ;       POINTER                         ;POINTER TO THE ASCII MESSAGE STRING

022430  032777  020000  156502  TY.MSG: BIT     #SW13,@SWR      ;INHIBIT TYPEOUT?
022436  001005                          BNE     2$              ;YES
022440  017637  000000  022450          MOV     @(SP),1$        ;GET POINTER TO ASCII STRING
022446  104401                          TYPE
022450  000000                  1$:     .WORD   0

022452  062716  000002          2$:     ADD     #2,(SP)         ;ADJUST RETURN ADDRESS TO SKIP OVER POINTER
022456  000002                          RTI
```

```
                                        .SBTTL  KWSRVE - KW11L CLOCK SERVICE ROUTINE
                                ;THIS ROUTINE SERVICES THE INTERRUPT FROM THE KW11L LINE CLOCK
                                ;AND KEEPS TRACK OF ELAPSED TIME.
                                ;KWCOUNT- CONTAINS CYCLES (PER SECOND) (2'S COMPLEMENT)
                                ;KWSEC- CONTAINS SECONDS (2'S COMPLEMENT)
                                ;KWMIN- CONTAINS MINUTES (2'S COMPLEMENT)
                                ;KWHR- CONTAINS HOURS (2'S COMPLEMENT)

022460  005237  001560          KWSRVE: INC     KWCOUNT         ;COUNT 60 CPS
022464  001401                          BEQ     1$              ;OVERFLOWED?
022466  000002                          RTI
022470  012737  177704  001560  1$:     MOV     #-60.,KWCOUNT   ;RESET 60 CPS COUNT
022476  005237  001556                  INC     KWSEC           ;COUNT SECONDS
022502  001401                          BEQ     2$              ;OVERFLOWED?
022504  000002                          RTI                     ;RETURN
022506  012737  177704  001556  2$:     MOV     #-60.,KWSEC     ;RESET "SECONDS" COUNT
022514  005237  001554                  INC     KWMIN           ;COUNT MINUTES
022520  001005                          BNE     3$              ;OVERFLOWED?
022522  012737  177704  001554          MOV     #-60.,KWMIN     ;RESET "MINUTES" COUNT
022530  005237  001552                  INC     KWHR            ;COUNT HOURS

022534  000002                  3$:     RTI                     ;RETURN



                                ;WATIME
                                ;ROUTINE PROVIDES SOME WAITING TIME.

022536  013746  022560          WATIME: MOV     2$,-(SP)        ;COUNTER VALUE
022542  005237  022562          1$:     INC     3$              ;COUNT
022546  001375                          BNE     1$              ;HANG IN THERE UNTIL COUNT WRAPS AROUND
022550  005216                          INC     (SP)            ;COUNT AGAIN
022552  001373                          BNE     1$              ;GO THROUGH MINOR LOOP AGAIN
022554  005726                          TST     (SP)+           ;RESTORE THE STACK POINTER
022556  000207                          RTS     PC              ;RETURN
022560  177730                  2$:     .WORD   177730          ;VALUE FOR APPROX 15 SEC DELAY
022562  000000                  3$:     .WORD   0               ;'MINOR' LOOP COUNTER
```

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

022564  005737  001264          CHDPRS: TST     DRVPRS          ;ANY DRIVES PRESENT?
022570  001401                          BEQ     1$              ;NO
022572  000207                          RTS     PC              ;YES, EXIT
022574  104401  002225          1$:     TYPE    ,MSG14          ;NO, GIVE A MESSAGE
022600  004737  022536                  JSR     PC,WATIME       ;KILL SOME TIME
022604  012706  001100                  MOV     #STACK,SP       ;REINITIALIZE STACK
022610  000400                          BR      $EOP            ;GO TO END OF PASS ROUTINE

```

```
                                        .SBTTL  END OF PASS ROUTINE

                                        ;;*************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO QMNGER

022612                                  $EOP:
022612  000004                                  SCOPE
022614  005037  001102                          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
022620  005237  001100                          INC     $PASS           ;;INCREMENT THE PASS NUMBER
022624  042737  100000  001100                  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
022632  005327                                  DEC     (PC)+           ;;LOOP?
022634  000001                          $EOPCT: .WORD   1
022636  003013                                  BGT     $DOAGN          ;;YES
022640  012737                                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
022642  000001                          $ENDCT: .WORD   1
022644  022634                                  $EOPCT
022646  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
022652  001405                                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
022654  000005                                  RESET                   ;;CLEAR THE WORLD
022656  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
022660  000240                                  NOP                     ;;SAVE ROOM
022662  000240                                  NOP                     ;;FOR
022664  000240                                  NOP                     ;;ACT11
022666                                  $DOAGN:
022666  000137                                  JMP     @(PC)+          ;;RETURN
022670  010556                          $RTNAD: .WORD   QMNGER
```

```
                                        .SBTTL  TTY INPUT ROUTINE

                                        ;;*************************************************************
                                        .ENABL  LSB

                                        ;;*************************************************************
                                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                        ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                        ;*WHEN OPERATING IN TTY FLAG MODE.
022672  022737  000176  001140          $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
022700  001074                                  BNE     15$             ;;BRANCH IF NO
022702  105777  156236                          TSTB    @$TKS           ;;CHAR THERE?
022706  100071                                  BPL     15$             ;;IF NO, DON'T WAIT AROUND
022710  117746  156232                          MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
022714  042716  177600                          BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
022720  022726  000007                          CMP     #7,(SP)+        ;;IS IT A CONTROL G?
022724  001062                                  BNE     15$             ;;NO, RETURN TO USER
022726  123727  001134  000001                  CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
022734  001456                                  BEQ     15$             ;;BRANCH IF YES

022736  104401  023417                          TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
022742  104401  023424                  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
022746  013746  000176                          MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
022752  104402                                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
022754  104401  023435                          TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
022760  005046                          19$:    CLR     -(SP)           ;;CLEAR COUNTER
022762  005046                                  CLR     -(SP)           ;;THE NEW SWR
022764  105777  156154                  7$:     TSTB    @$TKS           ;;CHAR THERE?
022770  100375                                  BPL     7$              ;;IF NOT TRY AGAIN

022772  117746  156150                          MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
022776  042716  177600                          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII



023002  021627  000025                  9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
023006  001005                                  BNE     10$             ;;BRANCH IF NOT
023010  104401  023412                          TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
023014  062706  000006                  20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
023020  000757                                  BR      19$             ;;LET'S TRY IT AGAIN


023022  021627  000015                  10$:    CMP     (SP),#15        ;;IS IT A <CR>?
023026  001022                                  BNE     16$             ;;BRANCH IF NO
023030  005766  000004                          TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
023034  001403                                  BEQ     11$             ;;BRANCH IF YES
023036  016677  000002  156074                  MOV     2(SP),@SWR      ;;SAVE NEW SWR
023044  062706  000006                  11$:    ADD     #6,SP           ;;CLEAR UP STACK
023050  104401  001213                  14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
023054  123727  001135  000001                  CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
023062  001003                                  BNE     15$             ;;BRANCH IF NOT
023064  012777  000100  156052                  MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
023072  000002                          15$:    RTI                     ;;RETURN
023074  004737  024322                  16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
023100  021627  000060                          CMP     (SP),#60        ;;CHAR < 0?
```

```
023104  002420                                  BLT     18$             ;;BRANCH IF YES
023106  021627  000067                          CMP     (SP),#67        ;;CHAR > 7?
023112  003015                                  BGT     18$             ;;BRANCH IF YES
023114  042726  000060                          BIC     #60,(SP)+       ;;STRIP-OFF ASCII
023120  005766  000002                          TST     2(SP)           ;;IS THIS THE FIRST CHAR
023124  001403                                  BEQ     17$             ;;BRANCH IF YES
023126  006316                                  ASL     (SP)            ;;NO, SHIFT PRESENT
023130  006316                                  ASL     (SP)            ;;   CHAR OVER TO MAKE
023132  006316                                  ASL     (SP)            ;;   ROOM FOR NEW ONE.
023134  005266  000002                  17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
023140  056616  177776                          BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
023144  000707                                  BR      7$              ;;GET THE NEXT ONE
023146  104401  001212                  18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
023152  000720                                  BR      20$             ;;SIMULATE CONTROL-U
                                        .DSABL  LSB



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

023154  011646                          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
023156  016666  000004  000002                  MOV     4(SP),2(SP)     ;;SAVE THE PS
023164  105777  155754                  1$:     TSTB    @$TKS           ;;WAIT FOR
023170  100375                                  BPL     1$              ;;A CHARACTER
023172  117766  155750  000004                  MOVB    @$TKB,4(SP)     ;;READ THE TTY
023200  042766  177600  000004                  BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
023206  026627  000004  000023                  CMP     4(SP),#23       ;;IS IT A CONTROL-S?
023214  001013                                  BNE     3$              ;;BRANCH IF NO
023216  105777  155722                  2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
023222  100375                                  BPL     2$              ;;LOOP UNTIL ITS THERE
023224  117746  155716                          MOVB    @$TKB,-(SP)     ;;GET CHARACTER
023230  042716  177600                          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
023234  022627  000021                          CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
023240  001366                                  BNE     2$              ;;IF NOT DISCARD IT
023242  000750                                  BR      1$              ;;YES, RESUME
023244  026627  000004  000140          3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
023252  002407                                  BLT     4$              ;;BRANCH IF YES
023254  026627  000004  000175                  CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
023262  003003                                  BGT     4$              ;;BRANCH IF YES
023264  042766  000040  000004                  BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
023272  000002                          4$:     RTI                     ;;GO BACK TO USER
                                        ;;*************************************************************
                                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                        ;*CALL:
                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

023274  010346                          $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
023276  012703  023402                  1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
023302  022703  023412                  2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
```

```
023306  101405                                  BLOS    4$              ;;BR IF YES
023310  104410                                  RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
023312  112613                                  MOVB    (SP)+,(R3)      ;;GET CHARACTER
023314  122713  000177                  10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
023320  001003                                  BNE     3$              ;;SKIP IF NOT
023322  104401  001212                  4$:     TYPE    ,$QUES          ;;TYPE A '?'
023326  000763                                  BR      1$              ;;CLEAR THE BUFFER AND LOOP
023330  111337  023400                  3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
023334  104401  023400                          TYPE    ,9$
023340  122723  000015                          CMPB    #15,(R3)+       ;;CHECK FOR RETURN
023344  001356                                  BNE     2$              ;;LOOP IF NOT RETURN
023346  105063  177777                          CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
023352  104401  001214                          TYPE    ,$LF            ;;TYPE A LINE FEED
023356  012603                                  MOV     (SP)+,R3        ;;RESTORE R3
023360  011646                                  MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
023362  016666  000004  000002                  MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
023370  012766  023402  000004                  MOV     #$TTYIN,4(SP)
023376  000002                                  RTI                     ;;RETURN
023400     000                          9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
023401     000                                  .BYTE   0               ;;TERMINATOR
023402  000010                          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
023412  052536  005015     000          $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
023417     136  006507  000012          $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
023424  005015  053523  020122          $MSWR:  .ASCIZ  <15><12>/SWR = /
023432  020075     000
023435     040  047040  053505          $MNEW:  .ASCIZ  /  NEW = /
023442  036440  000040
```

```
                                        .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                        ;*CHANGE IT TO BINARY.
                                        ;*CALL:
                                        ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                        ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                        ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT

023446  011646                          $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
023450  016666  000004  000002                  MOV     4(SP),2(SP)     ;;INPUT NUMBER
023456  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
023460  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
023462  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
023464  104411                          1$:     RDLIN                   ;;READ AN ASCIZ LINE
023466  012600                                  MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
023470  005001                                  CLR     R1              ;;CLEAR DATA WORD
023472  005002                                  CLR     R2
023474  112046                          2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
023476  001412                                  BEQ     3$              ;;IF ZERO GET OUT
023500  006301                                  ASL     R1              ;;*2
023502  006102                                  ROL     R2
023504  006301                                  ASL     R1              ;;*4
023506  006102                                  ROL     R2
023510  006301                                  ASL     R1              ;;*8
023512  006102                                  ROL     R2
023514  042716  177770                          BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
023520  062601                                  ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
023522  000764                                  BR      2$              ;;LOOP
023524  005726                          3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
023526  010166  000012                          MOV     R1,12(SP)       ;;SAVE THE RESULT
023532  010237  023546                          MOV     R2,$HIOCT
023536  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
023540  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
023542  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
023544  000002                                  RTI                     ;;RETURN
023546  000000                          $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
```

```
                                        .SBTTL  READ A DECIMAL NUMBER FROM THE TTY

                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
                                        ;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
                                        ;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
                                        ;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
                                        ;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
                                        ;*POSITIVE 32767 TO NEGATIVE 32768.
                                        ;*CALL:
                                        ;*      RDDEC                   ;;READ A DECIMAL NUMBER
                                        ;*      RETURN HERE             ;;NUMBER IS ON TOP OF THE STACK
                                        ;

023550  011646                          $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
023552  016666  000004  000002                  MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
023560  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
023562  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
023564  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
023566  104411                          1$:     RDLIN                   ;;READ AN ASCIZ LINE
023570  012600                                  MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
023572  010037  023716                          MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
023576  005046                                  CLR     -(SP)           ;;CLEAR DATA WORD
023600  005002                                  CLR     R2              ;;SIGN SET POSITIVE
023602  122710  000055                          CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
023606  001001                                  BNE     2$              ;;BR IF NO MINUS SIGN
023610  112002                                  MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
023612  112001                          2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
023614  001424                                  BEQ     3$              ;;GET OUT IF ZERO
023616  122701  000060                          CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
023622  003032                                  BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
023624  122701  000071                          CMPB    #'9,R1
023630  002427                                  BLT     5$
023632  032716  170000                          BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
023636  001024                                  BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
023640  006316                                  ASL     (SP)            ;;*2
023642  011646                                  MOV     (SP),-(SP)      ;;SAVE FOR LATER
023644  006316                                  ASL     (SP)            ;;*4
023646  006316                                  ASL     (SP)            ;;*8.
023650  062616                                  ADD     (SP)+,(SP)      ;;*10.
023652  102416                                  BVS     5$              ;;OVERFLOW ISN'T ALLOWED
023654  162701  000060                          SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
023660  060116                                  ADD     R1,(SP)         ;;ADD IN THIS DIGIT
023662  102412                                  BVS     5$              ;;OVERFLOW ISN'T ALLOWED
023664  000752                                  BR      2$              ;;LOOP
023666  005702                          3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
023670  001401                                  BEQ     4$              ;;BR IF NO
023672  005416                                  NEG     (SP)            ;;YES--NEGATE THE NUMBER
023674  012666  000012                  4$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
023700  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
023702  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
023704  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
023706  000002                                  RTI                     ;;RETURN

023710  005726                          5$:     TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
023712  105010                                  CLRB    (R0)            ;;SET A TERMINATOR
```

```
023714  104401                          TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
023716  000000                  6$:     .WORD   0               ;;POINTER GOES HERE
023720  104401  001212                  TYPE    ,$QUES          ;;"?" "CR" &"LF"
023724  000720                          BR      1$              ;;TRY AGAIN
```

```
                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE


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

023726                          $TYPDS:
023726  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
023730  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
023732  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
023734  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
023736  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
023740  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
023744  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
023750  100004                          BPL     1$              ;;BR IF INPUT IS POS.
023752  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
023754  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
023762  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
023764  012703  024142                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
023770  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
023774  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
023776  016001  024132                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
024002  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
024004  002402                          BLT     4$              ;;BR IF DONE
024006  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
024010  000774                          BR      3$
024012  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
024014  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
024016  001002                          BNE     5$              ;;FALL THROUGH IF 0
024020  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
024022  100407                          BMI     7$              ;;BR IF YES
024024  106316                  5$:     ASLB    (SP)            ;;MSD?
024026  103003                          BCC     6$              ;;BR IF NO
024030  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
024036  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
024042  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
024046  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
024050  005720                          TST     (R0)+           ;;JUST INCREMENTING
024052  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
024056  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
024060  003002                          BGT     8$              ;;GO TO EXIT
024062  010502                          MOV     R5,R2           ;;GET THE LSD
024064  000764                          BR      6$              ;;GO CHANGE TO ASCII
024066  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
024070  100003                          BPL     9$              ;;BR IF NO
024072  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
024100  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
024102  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
024104  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
024106  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
```

```
024110  012601                            MOV     (SP)+,R1            ;;POP STACK INTO R1
024112  012600                            MOV     (SP)+,R0            ;;POP STACK INTO R0
024114  104401  024142                    TYPE    ,$DBLK              ;;NOW TYPE THE NUMBER
024120  016666  000002  000004            MOV     2(SP),4(SP)         ;;ADJUST THE STACK
024126  012616                            MOV     (SP)+,(SP)
024130  000002                            RTI                         ;;RETURN TO USER
024132  023420                    $DTBL:  10000.
024134  001750                            1000.
024136  000144                            100.
024140  000012                            10.
024142  000004                    $DBLK:  .BLKW   4
```

```
                                  .SBTTL  TYPE ROUTINE

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

024152  105737  001157            $TYPE:  TSTB    $TPFLG              ;;IS THERE A TERMINAL?
024156  100002                            BPL     1$                  ;;BR IF YES
024160  000000                            HALT                        ;;HALT HERE IF NO TERMINAL
024162  000407                            BR      3$                  ;;LEAVE
024164  010046                    1$:     MOV     R0,-(SP)            ;;SAVE R0
024166  017600  000002                    MOV     @2(SP),R0           ;;GET ADDRESS OF ASCIZ STRING
024172  112046                    2$:     MOVB    (R0)+,-(SP)         ;;PUSH CHARACTER TO BE TYPED ONTO STACK
024174  001005                            BNE     4$                  ;;BR IF IT ISN'T THE TERMINATOR
024176  005726                            TST     (SP)+               ;;IF TERMINATOR POP IT OFF THE STACK
024200  012600                    60$:    MOV     (SP)+,R0            ;;RESTORE R0
024202  062716  000002            3$:     ADD     #2,(SP)             ;;ADJUST RETURN PC
024206  000002                            RTI                         ;;RETURN
024210  122716  000011            4$:     CMPB    #HT,(SP)            ;;BRANCH IF <HT>
024214  001430                            BEQ     8$
024216  122716  000200                    CMPB    #CRLF,(SP)          ;;BRANCH IF NOT <CRLF>
024222  001006                            BNE     5$
024224  005726                            TST     (SP)+               ;;POP  <CR><LF> EQUIV
024226  104401                            TYPE                        ;;TYPE A CR AND LF
024230  001213                            $CRLF
024232  105037  024366                    CLRB    $CHARCNT            ;;CLEAR CHARACTER COUNT
024236  000755                            BR      2$                  ;;GET NEXT CHARACTER
024240  004737  024322            5$:     JSR     PC,$TYPEC           ;;GO TYPE THIS CHARACTER
024244  123726  001156            6$:     CMPB    $FILLC,(SP)+        ;;IS IT TIME FOR FILLER CHARS.?
024250  001350                            BNE     2$                  ;;IF NO GO GET NEXT CHAR.
024252  013746  001154                    MOV     $NULL,-(SP)         ;;GET # OF FILLER CHARS. NEEDED
                                                                      ;;AND THE NULL CHAR.
024256  105366  000001            7$:     DECB    1(SP)               ;;DOES A NULL NEED TO BE TYPED?
024262  002770                            BLT     6$                  ;;BR IF NO--GO POP THE NULL OFF OF STACK
024264  004737  024322                    JSR     PC,$TYPEC           ;;GO TYPE A NULL
024270  105337  024366                    DECB    $CHARCNT            ;;DO NOT COUNT AS A COUNT
024274  000770                            BR      7$                  ;;LOOP

                                  ;HORIZONTAL TAB PROCESSOR

024276  112716  000040            8$:     MOVB    #' ,(SP)            ;;REPLACE TAB WITH SPACE
024302  004737  024322            9$:     JSR     PC,$TYPEC           ;;TYPE A SPACE
024306  132737  000007  024366            BITB    #7,$CHARCNT         ;;BRANCH IF NOT AT
024314  001372                            BNE     9$                  ;;TAB STOP
024316  005726                            TST     (SP)+               ;;POP SPACE OFF STACK
```

```
024320  000724                            BR      2$                  ;;GET NEXT CHARACTER
024322  105777  154622           $TYPEC:  TSTB    @$TPS               ;;WAIT UNTIL PRINTER IS READY
024326  100375                            BPL     $TYPEC
024330  116677  000002  154614            MOVB    2(SP),@$TPB         ;;LOAD CHAR TO BE TYPED INTO DATA REG.
024336  122766  000015  000002            CMPB    #CR,2(SP)           ;;IS CHARACTER A CARRIAGE RETURN?
024344  001003                            BNE     1$                  ;;BRANCH IF NO
024346  105037  024366                    CLRB    $CHARCNT            ;;YES--CLEAR CHARACTER COUNT
024352  000406                            BR      $TYPEX              ;;EXIT
024354  122766  000012  000002   1$:      CMPB    #LF,2(SP)           ;;IS CHARACTER A LINE FEED?
024362  001402                            BEQ     $TYPEX              ;;BRANCH IF YES
024364  105227                            INCB    (PC)+               ;;COUNT THE CHARACTER
024366  000000                   $CHARCNT:.WORD   0                   ;;CHARACTER COUNT STORAGE
024370  000207                   $TYPEX:  RTS     PC

```

```
                                 .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE


                                 ;;****************************************************************
                                 ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
                                 ;*UNSIGNED OCTAL ASCIZ NUMBER.
                                 ;*CALL
                                 ;*       MOV     #PNTR,-(SP)         ;;POINTER TO LOW WORD OF BINARY NUMBER
                                 ;*       JSR     PC,@#$DB20          ;;CALL THE ROUTINE
                                 ;*       RETURN                      ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK


024372  104414                   $DB20:   SAVREG                      ;;SAVE ALL REGISTERS
024374  016601  000002                    MOV     2(SP),R1            ;;PICKUP THE POINTER TO LOW WORD
024400  012705  024511                    MOV     #$OCTVL+13.,R5      ;;POINTER TO DATA TABLE
024404  012704  000014                    MOV     #12.,R4             ;;DO ELEVEN CHARACTERS
024410  012703  177770                    MOV     #^C7,R3             ;;MASK
024414  012100                            MOV     (R1)+,R0            ;;LOWER WORD
024416  012101                            MOV     (R1)+,R1            ;;HIGH WORD
024420  005002                            CLR     R2                  ;;TERMINATOR
024422  110245                   1$:      MOVB    R2,-(R5)            ;;PUT CHARACTER IN DATA TABLE
024424  010002                            MOV     R0,R2               ;;GET THIS DIGIT
024426  005304                            DEC     R4                  ;;COUNT THIS CHARACTER
024430  003007                            BGT     3$                  ;;BR IF NOT THE LAST DIGIT
024432  001405                            BEQ     2$                  ;;BR IF IT IS THE LAST DIGIT
024434  005205                            INC     R5                  ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
024436  010566  000002                    MOV     R5,2(SP)            ;;ASCIZ CHAR. & PUT IT ON THE STACK
024442  104415                            RESREG                      ;;RESTORE ALL REGISTERS
024444  000207                            RTS     PC                  ;;RETURN TO USER
024446  006203                   2$:      ASR     R3                  ;;POSITION THE MASK FOR THE LAST DIGIT
024450  006001                   3$:      ROR     R1                  ;;POSITION THE BINARY NUMBER FOR
024452  006000                            ROR     R0                  ;;       THE NEXT OCTAL DIGIT
024454  006001                            ROR     R1
024456  006000                            ROR     R0
024460  006001                            ROR     R1
024462  006000                            ROR     R0
024464  040302                            BIC     R3,R2               ;;MASK OUT ALL JUNK
024466  062702  000060                    ADD     #'0,R2              ;;MAKE THIS CHAR. ASCII
024472  000753                            BR      1$                  ;;GO PUT IT IN THE DATA TABLE
024474  000016                   $OCTVL:  .BLKB   14.                 ;;RESERVE DATA TABLE
```

```
                                        .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

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


024512  104414                          $DB2D:  SAVREG                  ;;SAVE REGISTERS
024514  016602  000002                          MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
024520  012700  024672                          MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
024524  010066  000002                          MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
024530  012201                                  MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
024532  012202                                  MOV     (R2)+,R2
024534  012737  000012  024610                  MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
024542  012704  024622                          MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
024546  012705  024624                          MOV     #$TNPWR+2,R5
024552  005003                          1$:     CLR     R3              ;;CLEAR PARTIAL
024554  161401                          2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
024556  005602                                  SBC     R2
024560  161502                                  SUB     (R5),R2
024562  002402                                  BLT     3$              ;;BR IF TEN POWER TO LARGE
024564  005203                                  INC     R3              ;;ADD 1 TO PARTIAL
024566  000772                                  BR      2$              ;;LOOP
024570  062401                          3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
024572  005502                                  ADC     R2
024574  062402                                  ADD     (R4)+,R2
024576  022525                                  CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
024600  052703  000060                          BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
024604  110320                                  MOVB    R3,(R0)+        ;;SAVE IT
024606  005327                                  DEC     (PC)+           ;;DONE?
024610  000000                          4$:     .WORD   0
024612  001357                                  BNE     1$              ;;BR IF NO
024614  105020                                  CLRB    (R0)+           ;;TERMINATOR
024616  104415                                  RESREG                  ;;RESTORE REGISTERS
024620  000207                                  RTS     PC              ;;RETURN
024622  145000                          $TNPWR: 145000                  ;;1.0E09
024624  035632                                  35632
024626  160400                                  160400                  ;;1.0E08
024630  002765                                  2765
024632  113200                                  113200                  ;;1.0E07
024634  000230                                  230
024636  041100                                  041100                  ;;1.0E06
024640  000017                                  17
024642  103240                                  103240                  ;;1.0E05
024644  000001                                  1
024646  023420                                  23420                   ;;1.0E04
024650  000000                                  0
024652  001750                                  1750                    ;;1.0E03
024654  000000                                  0
024656  000144                                  144                     ;;1.0E02
```

```
024660  000000                                  0
024662  000012                                  12                      ;;1.0E01
024664  000000                                  0
024666  000001                                  1                       ;;1.0E00
024670  000000                                  0
024672  000014                          $DECVL: .BLKB   12.             ;;RESERVE STORAGE FOR ASCIZ STRING
```

```
                                .SBTTL  SUPRS - TYPE NUMERICAL ASCIZ STRING, REPLACE LEADING 0'S BY BLANKS
                                .SBTTL  SUPRSL - TYPE NUMERICAL ASCIZ STRING, LEFT JUSTIFY
                                ;NOT FROM SYSMAC

024706  010046                  SUPRSL: MOV     R0,-(SP)          ;SAVE R0
024710  005037  024776                  CLR     SUP2
024714  016600  000004                  MOV     4(SP),R0
024720  000405                          BR      SUP1

024722  010046                  SUPRS:  MOV     R0,-(SP)          ;SAVE R0
024724  016600  000004                  MOV     4(SP),R0          ;PICKUP THE POINTER
024730  010037  024776                  MOV     R0,SUP2           ;SAVE FOR TYPING
024734                          SUP1:
024734  105710                  1$:     TSTB    (R0)              ;TERMINATOR?
024736  001406                          BEQ     2$                ;BR IF YES
024740  122710  000060                  CMPB    #'0,(R0)          ;IS THIS AN ASCII "0"?
024744  001006                          BNE     4$                ;NO
024746  112720  000040                  MOVB    #40,(R0)+         ;REPLACE IT WITH "BLANK"
024752  000770                          BR      1$
024754  005300                  2$:     DEC     R0                ;BACKUP BY 1
024756  112710  000060                  MOVB    #'0,(R0)          ;ASCII "0"
024762  005737  024776          4$:     TST     SUP2              ;LEFT JUSTIFY?
024766  001002                          BNE     5$                ;NO
024770  010037  024776                  MOV     R0,SUP2           ;YES
024774  104401                  5$:     TYPE                      ;GO TYPE
024776  000000                  SUP2:   .WORD   0
025000  012600                          MOV     (SP)+,R0          ;RESTORE R0
025002  012616                          MOV     (SP)+,(SP)        ;RESTORE THE STACK
025004  000207                          RTS     PC                ;RETURN
```

```
                                .SBTTL  INTEGER MULTIPLY ROUTINE


                                ;;***********************************************************
                                ;*CALL
                                ;*      MOV     MULTIPLER,-(SP)
                                ;*      MOV     MULTIPLICAND,-(SP)
                                ;*      JSR     PC,@#$MULT
                                ;*      RETURN  ;;PRODUCT IS ON THE STACK
                                ;*
                                ;*      STACK   PRODUCT
                                ;*      -----   -------
                                ;*      TOP     LSB'S
                                ;*      +2      MSB'S

025006                          $MULT:
025006  010046                          MOV     R0,-(SP)          ;;PUSH R0 ON STACK
025010  010146                          MOV     R1,-(SP)          ;;PUSH R1 ON STACK
025012  010246                          MOV     R2,-(SP)          ;;PUSH R2 ON STACK
025014  005046                          CLR     -(SP)             ;;CLEAR THE SIGN KEY
025016  016601  000012                  MOV     12(SP),R1         ;;GET THE MULTIPLICAND
025022  100002                          BPL     1$                ;;BR IF PLUS
025024  005216                          INC     (SP)              ;;SET THE SIGN KEY
025026  005401                          NEG     R1                ;;MAKE THE MULTIPLICAND POSTIVE
025030  016602  000014          1$:     MOV     14(SP),R2         ;;GET THE MULTIPLIER
025034  100002                          BPL     2$                ;;BR IF PLUS
025036  005316                          DEC     (SP)              ;;UPDATE THE SIGN KEY
025040  005402                          NEG     R2                ;;MAKE THE MULTIPLIER POSTIVE
025042  012746  000021          2$:     MOV     #17.,-(SP)        ;;SET THE LOOP COUNT
025046  005000                          CLR     R0                ;;SETUP FOR THE MULTIPLY LOOP
025050  103001                  3$:     BCC     4$                ;;DON'T ADD IF MULTIPLICAND = 0
025052  060200                          ADD     R2,R0
025054  006000                  4$:     ROR     R0                ;;POSITION THE PARITIAL PRODUCT AND
025056  006001                          ROR     R1                ;;THE MULTIPLICAND
025060  005316                          DEC     (SP)              ;;HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
025062  001372                          BNE     3$                ;;BR IF NO
025064  022616                          CMP     (SP)+,(SP)        ;;SHOULD PRODUCT BE NEGATIVE?
025066  001403                          BEQ     5$                ;;GO TO EXIT IF NO
025070  005400                          NEG     R0                ;;YES--SO MAKE IT SO
025072  005401                          NEG     R1
025074  005600                          SBC     R0
025076  005726                  5$:     TST     (SP)+             ;;CLEAR SIGN INFO, OFF OF STACK
025100  010066  000012                  MOV     R0,12(SP)         ;;PUT THE PRODUCT ON THE STACK (MSB'S)
025104  010166  000010                  MOV     R1,10(SP)         ;;LSB'S
025110  012602                          MOV     (SP)+,R2          ;;POP STACK INTO R2
025112  012601                          MOV     (SP)+,R1          ;;POP STACK INTO R1
025114  012600                          MOV     (SP)+,R0          ;;POP STACK INTO R0
025116  000207                          RTS     PC
```

```
                                .SBTTL  INTEGER DIVIDE ROUTINE

                                ;*CALL:
                                ;*      MOV     LOW DIVIDEND,-(SP)      ;THE HIGH DIVIDEND MUST BE < 1/2
                                ;*      MOV     HIGH DIVIDEND,-(SP)     ;       AS LARGE AS THE DIVISOR
                                ;*      MOV     DIVISOR,-(SP)
                                ;*      JSR     PC,$DIV
                                ;*      RETURN                          ;QUOTIENT & REMAINDER ARE ON THE STACK
                                ;*      "V"=0   IMPLIES NO ERROR
                                ;*      "V"=1   IMPLIES ERROR OCCURRED
                                ;*              "C"=0   DIVIDE OVERFLOW OCCURRED
                                ;*              "C"=1   ATTEMPTED TO DIVIDE BY ZERO
                                ;*
                                ;*
                                ;*      STACK   NO ERROR        OVERFLOW        DIVIDE BY ZERO
                                ;*      -----   --------        --------        --------------
                                ;*      TOP     REMAINDER       ALL ZEROS       ALL ONES
                                ;*      +2      QUOTIENT        ALL ZEROS       ALL ONES

025120  013746  000034          $DIV:   MOV     34,-(SP)        ;SAVE CURRENT TRAP VECTOR
025124  012737  025134  000034          MOV     #1$,34          ;SET UP TRAP VECTOR
025132  104400                          TRAP
025134  012716  025156          1$:     MOV     #2$,(SP)        ;REPLACE NEW PC
025140  016637  000004  000034          MOV     4(SP),34        ;RESTORE OLD TRAP VECT
025146  016666  000002  000004          MOV     2(SP),4(SP)     ;SAVE PSW
025154  000002                          RTI                     ;RESTORE PSW

025156  042716  000017          2$:     BIC     #17,(SP)        ;STRIP AWAY CONDITION CODES
025162  010046                          MOV     R0,-(SP)        ;PUSH R0 ON STACK
025164  010146                          MOV     R1,-(SP)        ;PUSH R1 ON STACK
025166  010246                          MOV     R2,-(SP)        ;PUSH R2 ON STACK
025170  010346                          MOV     R3,-(SP)        ;PUSH R3 ON STACK
025172  005046                          CLR     -(SP)           ;SAVE A PLACE FOR SIGNS
025174  012746  000021                  MOV     #17.,-(SP)      ;SETUP THE ITERATION COUNTER
025200  016601  000024                  MOV     24(SP),R1       ;PICKUP THE DIVIDEND
025204  016600  000022                  MOV     22(SP),R0
025210  100005                          BPL     3$              ;CHECK THE SIGN
025212  105366  000003                  DECB    3(SP)           ;KEEP TRACK OF THE SIGN
025216  005400                          NEG     R0              ;AND NEGATE THE ORIGINAL
025220  005401                          NEG     R1              ;NUMBER
025222  005600                          SBC     R0
025224  016602  000020          3$:     MOV     20(SP),R2       ;PICKUP THE DIVISOR
025230  022702  000001                  CMP     #1,R2           ;IF THE DIVISOR IS 1 SKIP THE REST
025234  001463                          BEQ     13$             ;YES
025236  005702                          TST     R2
025240  002407                          BLT     4$              ;CHECK THE SIGN
025242  003011                          BGT     5$              ;DIVISOR OF 0 IS A NO-NO
025244  052766  000003  000014          BIS     #3,14(SP)       ;SET "V" & "C"
025252  012700  177777                  MOV     #-1,R0          ;SET REMAINDER TO ALL ONES
025256  000424                          BR      9$              ;EXIT
025260  005266  000002          4$:     INC     2(SP)           ;KEEP TRACK OF DIVISORS SIGN
025264  000401                          BR      6$
025266  005402                  5$:     NEG     R2              ;NEGATE THE ORIGINAL NUMBER
025270  000241                  6$:     CLC                     ;CLEAR "C"
025272  000405                          BR      8$              ;START FORMING QUOTIENT
025274  006100                  7$:     ROL     R0              ;POSITION MSB'S
```

```
025276  010003                          MOV     R0,R3           ;COPY
025300  060203                          ADD     R2,R3           ;COMPARE DIVIDEND & DIVISOR
025302  103001                          BCC     8$              ;BR IF DIVIDEND > DIVISOR
025304  010300                          MOV     R3,R0           ;REMAINDER AFTER THIS LOOP
025306  006101                  8$:     ROL     R1              ;QUOTIENT BIT ENTERS HERE
025310  005316                          DEC     (SP)            ;DONE?
025312  001370                          BNE     7$              ;BR IF NO
025314  005701                          TST     R1              ;OVERFLOW?
025316  100005                          BPL     10$             ;BR IF NO
025320  052766  000002  000014          BIS     #2,14(SP)       ;SET "V" IN RETURN STATUS WORD
025326  005000                          CLR     R0              ;SET REMAINDER TO ALL ZEROS
025330  010001                  9$:     MOV     R0,R1           ;COPY REMAINDER INTO QUOTIENT
025332  005726                  10$:    TST     (SP)+           ;CLEAR COUNTER FROM STACK
025334  005716                          TST     (SP)            ;REMAINDER SIGN CORRECTION NEEDED?
025336  002004                          BGE     11$             ;BR IF NO
025340  005400                          NEG     R0              ;NEGATE REMAINDER
025342  105066  000001                  CLRB    1(SP)           ;CLEAR SIGN
025346  005316                          DEC     (SP)            ;BUT DON'T FORGET QUOTIENT
025350  005726                  11$:    TST     (SP)+           ;QUOTIENT SIGN CORRECTION NEEDED?
025352  001401                          BEQ     12$             ;BR IF NO
025354  005401                          NEG     R1              ;NEGATE QUOTIENT
025356  010166  000020          12$:    MOV     R1,20(SP)       ;RETURN QUOTIENT AND
025362  010066  000016                  MOV     R0,16(SP)       ;REMAINDER TO USER
025366  012603                          MOV     (SP)+,R3        ;POP STACK INTO R3
025370  012602                          MOV     (SP)+,R2        ;POP STACK INTO R2
025372  012601                          MOV     (SP)+,R1        ;POP STACK INTO R1
025374  012600                          MOV     (SP)+,R0        ;POP STACK INTO R0
025376  012666  000002                  MOV     (SP)+,2(SP)     ;SETUP TO RETURN CONDITION CODES
025402  000002                          RTI                     ;RETURN
025404  022626                  13$:    CMP     (SP)+,(SP)+     ;POP THE STACK
025406  000763                          BR      12$
```

```
                                        .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES

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

025410                                  $SAVREG:
025410  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
025412  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
025414  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
025416  010346                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
025420  010446                                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
025422  010546                                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
025424  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
025430  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
025434  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
025440  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
025444  000002                                  RTI

                                        ;*RESTORE R0-R5
                                        ;*CALL:
                                        ;*      RESREG
025446                                  $RESREG:
025446  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
025452  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
025456  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
025462  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
025466  012605                                  MOV     (SP)+,R5        ;;POP STACK INTO R5
025470  012604                                  MOV     (SP)+,R4        ;;POP STACK INTO R4
025472  012603                                  MOV     (SP)+,R3        ;;POP STACK INTO R3
025474  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
025476  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
025500  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
025502  000002                                  RTI
```

```
                                        .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
                                        ;*CALL:
                                        ;*      JSR     PC,$RAND                ;CALL THE ROUTINE
                                        ;*      RETURN                          ;RETURN HERE THE RANDOM
                                        ;*                                      ;NUMBER WILL BE IN
                                        ;*                                      ;$HINUM,$LONUM
025504                                  $RAND:
025504  010046                                  MOV     R0,-(SP)                ;PUSH R0 ON STACK
025506  010146                                  MOV     R1,-(SP)                ;PUSH R1 ON STACK
025510  010246                                  MOV     R2,-(SP)                ;PUSH R2 ON STACK
025512  010346                                  MOV     R3,-(SP)                ;PUSH R3 ON STACK
025514  010446                                  MOV     R4,-(SP)                ;PUSH R4 ON STACK
025516  017604  000012                          MOV     @12(SP),R4              ;GET POINTER TO THE SAVED SEEDS
                                                                                ;FOR GENERATING THIS RANDOM NUMBER
025522  011400                                  MOV     (R4),R0                 ;GET LO NUMBER SEED
025524  016401  000002                          MOV     2(R4),R1                ;GET HIGH NUMBER SEED
025530  012703  177771                          MOV     #-7,R3                  ;SET SHIFT COUNT
025534  005002                                  CLR     R2                      ;ZERO R2
025536  006300                          1$:     ASL     R0                      ;SHIFT R0 LEFT AND
025540  006101                                  ROL     R1                      ;ROTATE CARRY INTO R1 AND
025542  006102                                  ROL     R2                      ;ROTATE CARRY INTO R2
025544  005203                                  INC     R3                      ;CHECK FOR DONE
025546  001373                                  BNE     1$                      ;CONTINUE SHIFT LOOP
025550  061400                                  ADD     (R4),R0                 ;ADD NUMBER TO MAKE X 129
025552  005501                                  ADC     R1                      ;PROPOGATE CARRY
025554  066401  000002                          ADD     2(R4),R1                ;ADD NUMBER TO MAKE X 129
025560  005502                                  ADC     R2                      ;PROPOGATE CARRY
025562  062700  001057                          ADD     #1057,R0                ;ADD LOW CONSTANT
025566  005501                                  ADC     R1                      ;PROPOGATE CARRY
025570  005502                                  ADC     R2                      ;PROPOGATE CARRY
025572  062701  047401                          ADD     #47401,R1               ;ADD HIGH CONSTANT
025576  005502                                  ADC     R2                      ;PROPOGATE CARRY
025600  062702  000006                          ADD     #6,R2                   ;ADD HIGHEST CONSTART
025604  060200                                  ADD     R2,R0                   ;REPRIME R0 WITH HIGHEST DIGIT
025606  005501                                  ADC     R1                      ;PROPOGATE CARRY
025610  010014                                  MOV     R0,(R4)                 ;SAVE R0=$LONUM (FOR USE NXT TIME)
025612  010164  000002                          MOV     R1,2(R4)                ;SAVE R1=$HINUM (FOR USE NXT TIME)
025616  012604                                  MOV     (SP)+,R4
025620  012603                                  MOV     (SP)+,R3                ;POP STACK INTO R3
025622  012602                                  MOV     (SP)+,R2                ;POP STACK INTO R2
025624  012601                                  MOV     (SP)+,R1                ;POP STACK INTO R1
025626  012600                                  MOV     (SP)+,R0                ;POP STACK INTO R0
025630  062716  000002                          ADD     #2,(SP)                 ;ADJUST SP FOR CORRECT RETURN
025634  000207                                  RTS     PC                      ;RETURN
025636  123456                          RSDRVL: 123456                          ;RANDOM SEED FOR DRIVE SELECTION (LO)
025640  176543                          RSDRVH: 176543                          ; "   "  (HI)
025642  001201                          RSFUNL: 1201                            ;RANDOM SEED FOR FUNCTION
025644  062465                          RSFUNH: 62465                           ; "   "  (HI)
025646  176105                          RSCYLL: 176105                          ;RANDOM SEED FOR CYLINDER (LO)
025650  174532                          RSCYLH: 174532                          ; "   "  (HI)
025652  157650                          RSBAL:  157650                          ;RANDOM SEED FOR BUS "ADDRESS (LO)
025654  030753                          RSBAH:  30753                           ; "   "  (HI)
025656  131547                          RSWCL:  131547                          ;RANDOM SEED FOR WORD COUNT (LO)
025660  032070                          RSWCH:  32070                           ; "   "  (HI)
025662  123456                          RSDTL:  123456                          ;RANDOM SEED FOR DATA (LO)
025664  176543                          RSDTH:  176543                          ; "   "  (HI)
```

```
                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

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

025666  017646  000000          $TYPOS: MOV     @(SP),-(SP)         ;;PICKUP THE MODE
025672  116637  000001  026111          MOVB    1(SP),$0FILL        ;;LOAD ZERO FILL SWITCH
025700  112637  026113                  MOVB    (SP)+,$OMODE+1      ;;NUMBER OF DIGITS TO TYPE
025704  062716  000002                  ADD     #2,(SP)             ;;ADJUST RETURN ADDRESS
025710  000406                          BR      $TYPON
025712  112737  000001  026111  $TYPOC: MOVB    #1,$0FILL           ;;SET THE ZERO FILL SWITCH
025720  112737  000006  026113          MOVB    #6,$OMODE+1         ;;SET FOR SIX(6) DIGITS
025726  112737  000005  026110  $TYPON: MOVB    #5,$OCNT            ;;SET THE ITERATION COUNT
025734  010346                          MOV     R3,-(SP)            ;;SAVE R3
025736  010446                          MOV     R4,-(SP)            ;;SAVE R4
025740  010546                          MOV     R5,-(SP)            ;;SAVE R5
025742  113704  026113                  MOVB    $OMODE+1,R4         ;;GET THE NUMBER OF DIGITS TO TYPE
025746  005404                          NEG     R4
025750  062704  000006                  ADD     #6,R4               ;;SUBTRACT IT FOR MAX. ALLOWED
025754  110437  026112                  MOVB    R4,$OMODE           ;;SAVE IT FOR USE
025760  113704  026111                  MOVB    $0FILL,R4           ;;GET THE ZERO FILL SWITCH
025764  016605  000012                  MOV     12(SP),R5           ;;PICKUP THE INPUT NUMBER
025770  005003                          CLR     R3                  ;;CLEAR THE OUTPUT WORD
025772  006105                  1$:     ROL     R5                  ;;ROTATE MSB INTO "C"
025774  000404                          BR      3$                  ;;GO DO MSB
025776  006105                  2$:     ROL     R5                  ;;FORM THIS DIGIT
026000  006105                          ROL     R5
026002  006105                          ROL     R5
026004  010503                          MOV     R5,R3
026006  006103                  3$:     ROL     R3                  ;;GET LSB OF THIS DIGIT
026010  105337  026112                  DECB    $OMODE              ;;TYPE THIS DIGIT?
026014  100016                          BPL     7$                  ;;BR IF NO
026016  042703  177770                  BIC     #177770,R3          ;;GET RID OF JUNK
026022  001002                          BNE     4$                  ;;TEST FOR 0
026024  005704                          TST     R4                  ;;SUPPRESS THIS 0?
026026  001403                          BEQ     5$                  ;;BR IF YES
```

```
026030  005204                  4$:     INC     R4                  ;;DON'T SUPPRESS ANYMORE 0'S
026032  052703  000060                  BIS     #'0,R3              ;;MAKE THIS DIGIT ASCII
026036  052703  000040          5$:     BIS     #' ,R3              ;;MAKE ASCII IF NOT ALREADY
026042  110337  026106                  MOVB    R3,8$               ;;SAVE FOR TYPING
026046  104401  026106                  TYPE    ,8$                 ;;GO TYPE THIS DIGIT
026052  105337  026110          7$:     DECB    $OCNT               ;;COUNT BY 1
026056  003347                          BGT     2$                  ;;BR IF MORE TO DO
026060  002402                          BLT     6$                  ;;BR IF DONE
026062  005204                          INC     R4                  ;;INSURE LAST DIGIT ISN'T A BLANK
026064  000744                          BR      2$                  ;;GO DO THE LAST DIGIT
026066  012605                  6$:     MOV     (SP)+,R5            ;;RESTORE R5
026070  012604                          MOV     (SP)+,R4            ;;RESTORE R4
026072  012603                          MOV     (SP)+,R3            ;;RESTORE R3
026074  016666  000002  000004          MOV     2(SP),4(SP)         ;;SET THE STACK FOR RETURNING
026102  012616                          MOV     (SP)+,(SP)
026104  000002                          RTI                         ;;RETURN
026106     000                  8$:     .BYTE   0                   ;;STORAGE FOR ASCII DIGIT
026107     000                          .BYTE   0                   ;;TERMINATOR FOR TYPE ROUTINE
026110     000                  $OCNT:  .BYTE   0                   ;;OCTAL DIGIT COUNTER
026111     000                  $0FILL: .BYTE   0                   ;;ZERO FILL SWITCH
026112  000000                  $OMODE: .WORD   0                   ;;NUMBER OF DIGITS TO TYPE
```

```
                                        .SBTTL  ERROR HANDLER ROUTINE

                                        ;*SW15=1        HALT ON ERROR
                                        ;*SW13=1        INHIBIT ERROR TYPEOUTS
                                        ;*SW12=1        TYPE OUT THE ERROR HISTORY, THE FUNCTION THAT
                                        ;               WAS BEING PERFORMED ON RK AT THE TIME OF ERROR AND
                                        ;               THE FUNCTION PERFORMED PRIOR TO THAT.
                                        ;*NOTE THIS SWITCH OPTION (12) IS MEANINGFUL ONLY FOR ERRORS OCCURING IN THE
                                        ;*EXERCISER PART OF THE PROGRAM.
                                        ;*SW11=1        DUMP OUT ALL RK REGISTERS
                                        ;*SW10=1        BELL ON ERROR
                                        ;*SW09=1        LOOP ON ERROR
                                        ;*SW03=1        TYPE OUT TIME AT WHICH ERROR OCCURED
                                        ;*SW02=1        DROP THE DRIVE AFTER MAXM ERORS ON THIS DRIVE
                                        ;*SW01=1        TYPE OUT THE SERIAL NUMBER OF THE ERRORING DRIVE
                                        ;*GO TO $ERRTYP ON ERROR


026114  104407                          $ERROR: CKSWR                   ;LOOK FOR A 'CONTROL G'
026116  105237  001103                          INCB    $ERFLG          ;SET THE ERROR FLAG
026122  001774                                  BEQ     $ERROR          ;DON'T LET THE FLAG GO TO ZERO
026124  013777  001102  153010                  MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
026132  032777  002000  153000                  BIT     #SW10,@SWR      ;BELL ON ERROR?
026140  001402                                  BEQ     1$              ;NO - SKIP
026142  104401  001206                          TYPE    ,$BELL          ;RING BELL
026146  005237  001112                  1$:     INC     $ERTTL          ;COUNT THE NUMBER OF ERRORS

026152  032777  000004  152760                  BIT     #SW2,@SWR       ;COUNT # OF ERORS & DROP DRVE?
026160  001415                                  BEQ     3$              ;NO
                                                                        ;YES
026162  010146                                  MOV     R1,-(SP)        ;SAVE R1
026164  013701  001250                          MOV     SRDRV,R1        ;GET ERRORING DRIVE #
026170  100410                                  BMI     2$              ;IF (SRDRV)= -1, SKIP(BECAUSE THE
                                                                        ;EROR WAS NOT ATTRIBUTABLE TO ANY
                                                                        ;SPECIFIC DRIVE)
026172  105261  001542                          INCB    ERDRV(R1)       ;COUNT # OF ERORS ON THIS DRIVE
026176  126127  001542  000003                  CMPB    ERDRV(R1),#3    ;# OF ERORS GREATER THAN ALLOWABLE?
026204  101402                                  BLOS    2$              ;NO
026206  000137  016220                          JMP     DSELCT          ;DROP THE DRIVE

026212  012601                          2$:     MOV     (SP)+,R1        ;RESTORE R1

026214  011637  001116                  3$:     MOV     (SP),$ERRPC     ;GET ADDRESS OF ERROR INSTRUCTION
026220  162737  000002  001116                  SUB     #2,$ERRPC
026226  005046                                  CLR     -(SP)
026230  117716  152662                          MOVB    @$ERRPC,(SP)    ;STRIP AND SAVE THE ERROR ITEM CODE
026234  121627  000100                          CMPB    (SP),#100       ;FORM THE CO4$ECT ITEMB IF THIS IS AN
026240  002402                                  BLT     4$              ;EROR MESAGE EQUAL OR ABOVE 100.
026242  162716  000040                          SUB     #40,(SP)        ;NOTE THERE R 2 CLASSES OF ERRORS:
                                                                        ;1) $ITEMB'S BELOW 100
                                                                        ;2) $ITEMB'S ABOVE 100
                                                                        ;SUBTRACTION FACTOR HAS TOBE SUCH THAT
                                                                        ;THE CO4$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
```

```
026246  012637  001114                  4$:     MOV     (SP)+,$ITEMB
026252  032777  020000  152660                  BIT     #SW13,@SWR      ;SKIP TYPEOUT IF SET
026260  001012                                  BNE     5$              ;SKIP TYPEOUTS
026262  004737  026370                          JSR     PC,@#$ERRTYP    ;GO TO USER ERROR ROUTINE
026266  104401  001213                          TYPE    ,$CRLF

026272  032777  010000  152640                  BIT     #SW12,@SWR      ;TYPE ERROR HISTORY?
026300  001402                                  BEQ     5$              ;NO
026302  004737  020334                          JSR     PC,HISTRY       ;YES

026306  005777  152626                  5$:     TST     @SWR            ;HALT ON ERROR
026312  100002                                  BPL     6$              ;SKIP IF CONTINUE
026314  000000                                  HALT                    ;HALT ON ERROR!
026316  104407                                  CKSWR                   ;LOOK FOR A 'CONTROL G'
026320  032777  001000  152612          6$:     BIT     #SW09,@SWR      ;LOOP ON ERROR SWITCH SET?
026326  001411                                  BEQ     7$              ;BR IF NO
026330  123727  001114  000040                  CMPB    $ITEMB,#40      ;THERE R 37 ERROR MESSAGES IN CLASS 1
026336  103011                                  BHIS    8$
026340  013746  001244                          MOV     PPRLVL,-(SP)    ;LOCK OUT ALL INTERUPTS ON RETURN
                                                                        ;FROM THIS EROR HANDLER, IF THE EROR
                                                                        ;IS IN EXERCISER & LOOPING IS TO
                                                                        ;BE DONE
026344  005726                                  TST     (SP)+
026346  012716  015634                          MOV     #EXCRLUP,(SP)   ;IF THIS ERROR CALL WAS FROM EXERCISER
                                                                        ;PART OF THE PROGRAM, GO TO 'EXCRLUP'
                                                                        ;OTHERWISE RETURN THRU '$LUPERR'

026352  012737  177777  001250          7$:     MOV     #-1,SRDRV       ;RESET SERIAL NO FLAG
                                                                        ;IF (SRDRV)=-1, THEN THE SERIAL
                                                                        ;NO OF THE DRIVE WILL NOT BE
                                                                        ;TYPED OUT.
                                                                        ;OTHERWISE, SERIAL NO FOR THE DRIVE
                                                                        ;# IN 'SRDRV' WILL BE TYPED.
026360  000002                                  RTI
026362  013716  001110                  8$:     MOV     $LPERR,(SP)     ;FUDGE RETURN FOR LOOPING
026366  000771                                  BR      7$              ;RETURN
```

```
                                        .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

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


026370    104401    001213              $ERRTYP:TYPE      ,$CRLF              ;"CARRIAGE RETURN" & "LINE FEED"
026374    010046                                  MOV       R0,-(SP)            ;SAVE R0
026376    005000                                  CLR       R0                  ;PICKUP THE ITEM INDEX
026400    153700    001114                        BISB      @#$ITEMB,R0
026404    001004                                  BNE       1$                  ;IF ITEM NUMBER IS ZERO, JUST
026406    013746    001116                        MOV       $ERRPC,-(SP)        ;TYPE THE PC OF THE ERROR
026412    104402                                  TYPOC
026414    000425                                  BR        5$                  ;GET OUT
026416    005300                        1$:       DEC       R0                  ;ADJUST THE INDEX SO THAT IT WILL
026420    006300                                  ASL       R0                  ;WORK FOR THE ERROR TABLE
026422    006300                                  ASL       R0
026424    006300                                  ASL       R0
026426    062700    002666                        ADD       #$ERRTB,R0          ;FORM TABLE POINTER
026432    012037    026442                        MOV       (R0)+,2$            ;PICKUP "ERROR MESSAGE" POINTER
026436    001404                                  BEQ       4$                  ;SKIP TYPEOUT IF NO POINTER
026440    104401                                  TYPE                          ;TYPE THE "ERROR MESSAGE"
026442    000000                        2$:       .WORD     0                   ;"ERROR MESSAGE" POINTER GOES HERE
026444    104401    001213              3$:       TYPE      ,$CRLF              ;"CARRIAGE RETURN" & "LINE FEED"
026450    032777    004000    152462    4$:       BIT       #SW11,@SWR          ;DUMP OUT RK REGISTERS?
026456    001404                                  BEQ       5$
026460    004737    026730                        JSR       PC,DMPREG           ;GO TYPE OUT RK REGISTERS
026464    104401    001213                        TYPE      ,$CRLF

026470    012037    026500              5$:       MOV       (R0)+,6$            ;PICKUP "DATA HEADER" POINTER
026474    001404                                  BEQ       7$                  ;SKIP TYPEOUT IF 0
026476    104401                                  TYPE                          ;TYPE THE "DATA HEADER"
026500    000000                        6$:       .WORD     0                   ;"DATA HEADER" POINTER GOES HERE
026502    104401    001213                        TYPE      ,$CRLF              ;"CARRIAGE RETURN" & "LINE FEED"
026506    011000                        7$:       MOV       (R0),R0             ;PICKUP "DATA TABLE" POINTER
026510    001406                                  BEQ       9$

                                                                                ;TYPE AN OCTAL NUMBER
026512    013046                        8$:       MOV       @(R0)+,-(SP)        ;SAVE @(R0)+ FOR TYPEOUT
026514    104402                                  TYPOC                         ;GO TYPE--OCTAL ASCII(ALL DIGITS)
026516    104401    002663                        TYPE      ,BLNKS2             ;TYPE TWO(2) SPACES
026522    005710                                  TST       (R0)                ;IS THERE ANOTHER NUMBER?
026524    001372                                  BNE       8$                  ;BR IF YES

026526    012600                        9$:       MOV       (SP)+,R0            ;RESTORE R0
026530    104401    001213                        TYPE      ,$CRLF              ;"CARRIAGE RETURN" & "LINE FEED"
026534    123727    001114    000040              CMPB      $ITEMB,#40          ;SKIP TIME & SERIAL # FOR
                                                                                ;NON-EXERCISER ERORS
026542    103004                                  BHIS      10$
026544    004737    026556                        JSR       PC,TIMTYP           ;TYPE OUT TIME IF SW 3 IS SET
026550    004737    026664                        JSR       PC,SNOTYP           ;GO TYPE OUT THE SERIAL # OF THE
                                                                                ;ERRORING DRIVE, IF SW 1 IS SET.
026554    000207                        10$:      RTS       PC                  ;RETURN
```

```
                                        ;TIMTYP
                                        ;THIS ROUTINE TYPES THE TIME IN HOURS:MIN:SECS IF SW 3 IS SET
                                        ;SW 3 SHOULD NOT BE SET IF KW11L IS NOT PRESENT.

026556    032777    000010    152354    TIMTYP: BIT       #SW3,@SWR           ;IS SW 3 SET?
026564    001434                                  BEQ       4$                  ;IF NOT SKIP TYPING TIME
026566    104401    002652                        TYPE      ,MSG29              ;'TIME'
026572    104401    002662                        TYPE      ,BLNKS3
026576    104414                                  SAVREG                        ;SAVE R0-R4
026600    012700    001552                        MOV       #KWHR,R0            ;INITIALIZE POINTER
026604    012001                                  MOV       (R0)+,R1
026606    000404                                  BR        2$
026610    012001                        1$:       MOV       (R0)+,R1            ;TYPE OUT
026612    001402                                  BEQ       2$
026614    062701    000074                        ADD       #60.,R1
026620    010137    026660              2$:       MOV       R1,5$
026624    012746    026660                        MOV       #5$,-(SP)           ;HOURS:MINS:SECS
026630    004737    024512                        JSR       PC,@#$DB2D          ;CONVERT TO ASCIZ STRING
026634    004737    024706                        JSR       PC,@#$UPRSL         ;GO TYPE
026640    020027    001560                        CMP       R0,#KWSEC+2         ;ALL DONE?
026644    001403                                  BEQ       3$
026646    104401    002334                        TYPE      ,MSG18
026652    000756                                  BR        1$
026654    104415                        3$:       RESREG                        ;RESTORE R0-R4
026656    000207                        4$:       RTS       PC                  ;RETURN
026660    000000    000000              5$:       .WORD     0,0
```

```
                                      ;SNOTYP
                                      ;THIS ROUTINE TYPES OUT THE SERIAL NUMBER OF THE ERRORING DRIVE, IF SW 1
                                      ;IS SET. NOTE THAT THE SERIAL NUMBER IS TYPED OUT ONLY WHEN THE DRIVE
                                      ;CAN BE IDENTIFIED POSITIVELY, AS THE ONE WHICH GAVE THE ERROR. IF THE
                                      ;ERROR CANNOT BE ATTRIBUTED TO ANY SPECIFIC DRIVE < (SRDRV)= -1> THEN
                                      ;THE SERIAL NUMBER IS NOT TYPED OUT.

 026664  032777  000002  152246       SNOTYP: BIT     #SW1,@SWR       ;TYPE OUT SERIAL #?
 026672  001415                               BEQ     2$              ;NO
 026674  010146                               MOV     R1,-(SP)        ;SAVE R1
 026676  013701  001250                       MOV     SRDRV,R1        ;GET ERRORING DRIVE #
 026702  006301                               ASL     R1              ;IF (SRDRV)= -1, SKIP (BECAUSE
 026704  100407                               BMI     1$              ;THE ERROR WAS NOT ATTRIBUTABLE
                                                                      ;TO A SPECIFIC DRIVE)
 026706  104401  001213                       TYPE    ,$CRLF
 026712  104401  002326                       TYPE    ,MSG17          ;TYPE "SR. NO:"
 026716  016146  001266                       MOV     SRNO(R1),-(SP)  ;GET THE SERIAL #
 026722  104405                               TYPDS                   ;TYPE IT OUT (DECIMAL)

 026724  012601                       1$:     MOV     (SP)+,R1        ;RESTORE R1
 026726  000207                       2$:     RTS     PC              ;RETURN



                                      ;DMPREG
                                      ;THIS ROUTINE DUMPS OUT ALL RK11 REGISTERS WHEN SW 11 IS SET AND AN ERROR OCCURS.

 026730                               DMPREG:
 026730  104401  026736                       TYPE    ,65$            ;;TYPE ASCIZ STRING
 026734  000441                               BR      64$             ;;GET OVER THE ASCIZ
                                      ;;65$:  .ASCIZ  <15><12>/  PC      RKDS     RKER     RKCS     RKWC     RKBA     RKDA     RKDB/<
 027040                               64$:
 027040  013746  001116                       MOV     $ERRPC,-(SP)
 027044  104402                               TYPOC
 027046  104401  002663                       TYPE    ,BLNKS2
 027052  010046                               MOV     R0,-(SP)
 027054  012700  001216                       MOV     #RKDS,R0
 027060  013046                       1$:     MOV     @(R0)+,-(SP)
 027062  104402                               TYPOC
 027064  104401  002663                       TYPE    ,BLNKS2
 027070  020027  001232                       CMP     R0,#RKDB
 027074  003771                               BLE     1$
 027076  012600                               MOV     (SP)+,R0
 027100  000207                               RTS     PC
```

```
                                      .SBTTL  SCOPE HANDLER ROUTINE

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

 027102                               $SCOPE:
 027102  104407                               CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
 027104  032777  040000  152026       1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
 027112  001047                               BNE     $OVER           ;;YES IF SW14=1
                                      ;#####START OF CODE FOR THE XOR TESTER#####
 027114  000416                       SXTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                      ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
 027116  013746  000004                       MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 027122  012737  027142  000004               MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
 027130  005737  177060                       TST     @#177060        ;;TIME OUT ON XOR?
 027134  012637  000004                       MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 027140  000421                               BR      $SVLAD          ;;GO TO THE NEXT TEST
 027142  022626                       5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
 027144  012637  000004                       MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 027150  000407                               BR      7$              ;;LOOP ON THE PRESENT TEST
 027152                               6$:;;#####END OF CODE FOR THE XOR TESTER#####
 027152  105737  001103               2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 027156  001412                               BEQ     $SVLAD          ;;BR IF NO
 027160  032777  001000  151752               BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
 027166  001404                               BEQ     4$              ;;BR IF NO
 027170  013737  001110  001106       7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 027176  000415                               BR      $OVER
 027200  105037  001103               4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
 027204  105237  001102               $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
 027210  011637  001106                       MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 027214  011637  001110                       MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 027220  005037  001204                       CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 027224  112737  000001  001115               MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 027232  013777  001102  151702       $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 027240  013716  001106                       MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 027244  000002                               RTI                     ;;FIXES PS
```

```
                                        .SBTTL  TRAP DECODER

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

027246  010046                          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
027250  016600  000002                          MOV     2(SP),R0        ;;GET TRAP ADDRESS
027254  005740                                  TST     -(R0)           ;;BACKUP BY 2
027256  111000                                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
027260  006300                                  ASL     R0              ;;POSITION FOR INDEXING
027262  016000  027302                          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
027266  000200                                  RTS     R0              ;;GO TO ROUTINE


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

027270  011646                          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
027272  016666  000004  000002                  MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
027300  000002                                  RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------
027302  027270                          $TRPAD: .WORD   $TRAP2
027304  024152                                  $TYPE   ;;CALL=TYPE       TRAP+1(104401)  TTY TYPEOUT ROUTINE
027306  025712                                  $TYPOC  ;;CALL=TYPOC      TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
027310  025666                                  $TYPOS  ;;CALL=TYPOS      TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
027312  025726                                  $TYPON  ;;CALL=TYPON      TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
027314  023726                                  $TYPDS  ;;CALL=TYPDS      TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

027316  022742                                  $GTSWR  ;;CALL=GTSWR      TRAP+6(104406)  GET SOFT-SWR SETTING

027320  022672                                  $CKSWR  ;;CALL=CKSWR      TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
027322  023154                                  $RDCHR  ;;CALL=RDCHR      TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
027324  023274                                  $RDLIN  ;;CALL=RDLIN      TRAP+11(104411) TTY TYPEIN STRING ROUTINE
027326  023446                                  $RDOCT  ;;CALL=RDOCT      TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
027330  023550                                  $RDDEC  ;;CALL=RDDEC      TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY
027332  025410                                  $SAVREG ;;CALL=SAVREG     TRAP+14(104414) SAVE R0-R5 ROUTINE
027334  025446                                  $RESREG ;;CALL=RESREG     TRAP+15(104415) RESTORE R0-R5 ROUTINE
027336  022264                                  CN.RST  ;;CALL=CON.RESET          TRAP+16(104416) CONTROL RESET ROUTINE
027340  022272                                  CN.RDY  ;;CALL=CON.RDY    TRAP+17(104417) WAIT FOR CONTROL READY
027342  022152                                  DR.RST  ;;CALL=DRV.RESET          TRAP+20(104420) DRIVE RESET ROUTINE
027344  022430                                  TY.MSG  ;;CALL=TYPMSG     TRAP+21(104421) TYPE MESSAGE ROUTINE, SW13

```

```
                                        .SBTTL  POWER DOWN AND UP ROUTINES

                                        ;;*************************************************************
                                        ;POWER DOWN ROUTINE
027346  012737  027512  000024          $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
027354  012737  000340  000026                  MOV     #340,@#PWRVEC+2 ;;PRIO:7
027362  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
027364  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
027366  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
027370  010346                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
027372  010446                                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
027374  010546                                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
027376  017746  151536                          MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
027402  010637  027516                          MOV     SP,$SAVR6       ;;SAVE SP
027406  012737  027420  000024                  MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
027414  000000                                  HALT
027416  000776                                  BR      .-2             ;;HANG UP


                                        ;;*************************************************************
                                        ;POWER UP ROUTINE
027420  012737  027512  000024          $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
027426  013706  027516                          MOV     $SAVR6,SP       ;;GET SP
027432  005037  027516                          CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
027436  005237  027516                  1$:     INC     $SAVR6          ;;WAIT FOR THE INC
027442  001375                                  BNE     1$              ;;OF  WORD
027444  012677  151470                          MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
027450  012605                                  MOV     (SP)+,R5        ;;POP STACK INTO R5
027452  012604                                  MOV     (SP)+,R4        ;;POP STACK INTO R4
027454  012603                                  MOV     (SP)+,R3        ;;POP STACK INTO R3
027456  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
027460  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
027462  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
027464  012737  027346  000024                  MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
027472  012737  000340  000026                  MOV     #340,@#PWRVEC+2 ;;PRIO:7
027500  104401                                  TYPE                    ;REPORT THE POWER FAILURE
027502  027520                          $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
027504  012716                                  MOV     (PC)+,(SP)      ;;RESTART AT PFSTRT
027506  003366                          $PWRAD: .WORD   PFSTRT          ;;RESTART ADDRESS
027510  000002                                  RTI
027512  000000                          $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
027514  000776                                  BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
027516  000000                          $SAVR6: 0                       ;;PUT THE SP HERE
027520  005015  047520  042527          $POWER: .ASCIZ  <15><12>"POWER"
027526  000122
                                                .EVEN
                                        ;ERROR MESSAGES

        027530  051105  051117  047440  EM1:    .ASCIZ  /EROR ON WRITE/
        027536  020116  051127  052111
        027544  000105
        027546  052101  046505  052120  EM2:    .ASCIZ  /ATEMPT TO INITIATE FUNCTION ON 'BUSY' DRVE/
        027554  052040  020117  047111
        027562  052111  040511  042524
        027570  043040  047125  052103
        027576  047511  020116  047117
        027604  023440  052502  054523
        027612  020047  051104  042526
        027620     000
        027621     103  052116  047522  EM3:    .ASCIZ  /CNTROL RDY NOT SET/
        027626  020114  042122  020131
        027634  047516  020124  042523
        027642  000124
        027644  051057  053457  051457  EM4:    .ASCIZ  "/R/W/S RDY NOT SET"
        027652  051040  054504  047040
        027660  052117  051440  052105
        027666     000
        027667     103  052116  047522  EM5:    .ASCIZ  /CNTROL RDY NOT SET AFTER 1ST INTRUPT ON ISSUING SEEK/
        027674  020114  042122  020131
        027702  047516  020124  042523
        027710  020124  043101  042524
        027716  020122  051461  020124
        027724  047111  051124  050125
        027732  020124  047117  044440
        027740  051523  044525  043516
        027746  051440  042505  000113
        027754  051127  047117  020107  EM6:    .ASCIZ  /WRONG BITS IN RKCS, EXPCT SEEK/
        027762  044502  051524  044440
        027770  020116  045522  051503
        027776  020054  054105  041520
        030004  020124  042523  045505
        030012     000
        030013     047  052502  054523  EM7:    .ASCIZ  /'BUSY' FLAG CLEAR ON INTRUPTING DRVE/
        030020  020047  046106  043501
        030026  041440  042514  051101
        030034  047440  020116  047111
        030042  051124  050125  044524
        030050  043516  042040  053122
        030056  000105
        030060  050047  051517  052111  EM10:   .ASCIZ  /'POSITIONING' FLAG FOR INTRUPTING DRVE CLEAR/
        030066  047511  044516  043516
        030074  020047  046106  043501
        030102  043040  051117  044440
        030110  052116  052522  052120
        030116  047111  020107  051104
        030124  042526  041440  042514
        030132  051101     000
        030135     047  051105  023522  EM11:   .ASCIZ  /'ERR'OR SET AFTER 1ST INTERRUPT ON ISSUING SEEK/
        030142  051117  051440  052105
        030150  040440  052106  051105
        030156  030440  052123  044440
        030164  052116  051105  052522
```

```
        030172  052120  047440  020116
        030200  051511  052523  047111
        030206  020107  042523  045505
        030214     000
        030215     123  050103  051440  EM12:   .ASCIZ  /SCP SET AFTER 1ST INTRUPT ON ISSUING SEEK/
        030222  052105  040440  052106
        030230  051105  030440  052123
        030236  044440  052116  052522
        030244  052120  047440  020116
        030252  051511  052523  047111
        030260  020107  042523  045505
        030266     000
        030267     103  052116  047522  EM13:   .ASCIZ  /CNTROL RDY NOT SET AFTER SEEK DONE INTRUPT/
        030274  020114  042122  020131
        030302  047516  020124  042523
        030310  020124  043101  042524
        030316  020122  042523  045505
        030324  042040  047117  020105
        030332  047111  051124  050125
        030340  000124
        030342  047111  051124  050125  EM14:   .ASCIZ  /INTRUPTING DRVE (SEEK DONE) WAS NOT 'BUSY'/
        030350  044524  043516  042040
        030356  053122  020105  051450
        030364  042505  020113  047504
        030372  042516  020051  040527
        030400  020123  047516  020124
        030406  041047  051525  023531
        030414     000
        030415     122  053457  051457  EM15:   .ASCIZ  "R/W/S READY NOT SET FOR INTRUPTING DRVE (SEEK DONE)"
        030422  051040  040505  054504
        030430  047040  052117  051440
        030436  052105  043040  051117
        030444  044440  052116  052522
        030452  052120  047111  020107
        030460  051104  042526  024040
        030466  042523  045505  042040
        030474  047117  024505     000
        030501     123  047111  042440  EM16:   .ASCIZ  /SIN EROR/
        030506  047522  000122
        030512  042447  051122  047447  EM17:   .ASCIZ  /'ERR'OR ON DOING SEEK/
        030520  020122  047117  042040
        030526  044517  043516  051440
        030534  042505  000113
        030540  041523  020120  044504  EM20:   .ASCIZ  /SCP DID NOT SET AFTER SEEK WAS DONE/
        030546  020104  047516  020124
        030554  042523  020124  043101
        030562  042524  020122  042523
        030570  045505  053440  051501
        030576  042040  047117  000105
        030604  047523  052106  042440  EM21:   .ASCIZ  /SOFT EROR/
        030612  047522  000122
        030616  040504  040524  024040  EM23:   .ASCIZ  /DATA (COMPARISON) EROR/
        030624  047503  050115  051101
        030632  051511  047117  020051
        030640  051105  051117     000
        030645     103  052116  047522  EM24:   .ASCIZ  /CNTROL RDY CLR ON INTRUPT AFTER RK FUNCTION/
```

```
030652  020114  042122  020131
030660  046103  020122  047117
030666  044440  052116  052522
030674  052120  040440  052106
030702  051105  051040  020113
030710  052506  041516  044524
030716  047117     000
030721     123  052524  045503  EM26:   .ASCIZ  /STUCK IN LOOP,8 COMANDS SHLDBE DONE BY NOW/
030726  044440  020116  047514
030734  050117  034054  041440
030742  046517  047101  051504
030750  051440  046110  041104
030756  020105  047504  042516
030764  041040  020131  047516
030772  000127
030774  052101  050115  020124  EM27:   .ASCIZ  /ATMPT TO DO WRITE BEFORE WRT CHK/
031002  047524  042040  020117
031010  051127  052111  020105
031016  042502  047506  042522
031024  053440  052122  041440
031032  045510     000
031035     101  046524  052120  EM30:   .ASCIZ  /ATMPT TO REEXECUTE COMMAND-IN PROGRESS OR ALREADY FINISHED/
031042  052040  020117  042522
031050  054105  041505  052125
031056  020105  047503  046515
031064  047101  026504  047111
031072  050040  047522  051107
031100  051505  020123  051117
031106  040440  051114  040505
031114  054504  043040  047111
031122  051511  042510  000104
031130  043047  047125  052103  EM31:   .ASCIZ  /'FUNCTION IN PROGRES' FLG FOR INTRUPTING DRIVE ISN'T SET/
031136  047511  020116  047111
031144  050040  047522  051107
031152  051505  020047  046106
031160  020107  047506  020122
031166  047111  051124  050125
031174  044524  043516  042040
031202  044522  042526  044440
031210  047123  052047  051440
031216  052105     000
031221     125  042516  050130  EM32:   .ASCIZ  /UNEXPCTED DRIVE INTRUPTED/
031226  052103  042105  042040
031234  044522  042526  044440
031242  052116  052522  052120
031250  042105     000
031253     125  042516  050130  EM33:   .ASCIZ  /UNEXPCTD FUNCTION CODE IN RKCS AFTER INTRUPT/
031260  052103  020104  052506
031266  041516  044524  047117
031274  041440  042117  020105
031302  047111  051040  041513
031310  020123  043101  042524
031316  020122  047111  051124
031324  050125  000124
031330  051104  042526  051040  EM34:   .ASCIZ  /DRVE RDY CLEAR/
031336  054504  041440  042514
```

```
031344  051101     000
031347     104  053122  020105  EM35:   .ASCIZ  /DRVE POWER LO/
031354  047520  042527  020122
031362  047514     000
031365     104  053122  020105  EM36:   .ASCIZ  /DRVE UNSAFE/
031372  047125  040523  042506
031400     000
031401     127  051520  051440  EM37:   .ASCIZ  /WPS SET/
031406  052105     000
031411     111  052116  051105  EM101:  .ASCIZ  /INTERUPT DIDN'T OCUR AFTER WRTE/
031416  050125  020124  044504
031424  047104  052047  047440
031432  052503  020122  043101
031440  042524  020122  051127
031446  042524     000
031451     047  051105  023522  EM102:  .ASCIZ  /'ERR'OR SET/
031456  051117  051440  052105
031464     000
031465     122  042113  020101  EM103:  .ASCIZ  /RKDA INCRMENTED WRONG/
031472  047111  051103  042515
031500  052116  042105  053440
031506  047522  043516     000
031513     122  041113  020101  EM104:  .ASCIZ  /RKBA INCRMENTED WRONG/
031520  047111  051103  042515
031526  052116  042105  053440
031534  047522  043516     000
031541     122  053513  020103  EM105:  .ASCIZ  /RKWC DIDN'T OVRFLO TO 0/
031546  044504  047104  052047
031554  047440  051126  046106
031562  020117  047524  030040
031570     000
031571     115  054105  041040  EM106:  .ASCIZ  /MEX BITS WRONG/
031576  052111  020123  051127
031604  047117  000107
031610  051127  042524  041440  EM110:  .ASCIZ  /WRTE CHK EROR/
031616  045510  042440  047522
031624  000122
```

```
                                        ;DATA HEADERS
031626  020040  041520  020040  DH1:     .ASCIZ  /  PC      RKCS     RKER     RKDS     RKDA/
031634  020040  051040  041513
031642  020123  020040  051040
031650  042513  020122  020040
031656  051040  042113  020123
031664  020040  051040  042113
031672  000101

031674  020040  041520  020040  DH2:     .ASCIZ  /  PC     DRV#/
031702  020040  051104  021526
031710     000

031711     040  050040  020103  DH21:    .ASCIZ  /  PC      RKCS     RKER     RKDS   DRIVE     CYL      SUR      SEC/
031716  020040  020040  045522
031724  051503  020040  020040
031732  045522  051105  020040
031740  020040  045522  051504
031746  020040  042040  044522
031754  042526  020040  020040
031762  054503  020114  020040
031770  020040  052523  020122
031776  020040  020040  042523
032004  000103

032006  020040  041520  020040  DH23:    .ASCIZ  /  PC      RKBA    EXPCT    RECVD    RKDA/
032014  020040  045522  040502
032022  020040  020040  054105
032030  041520  020124  020040
032036  042522  053103  020104
032044  020040  045522  040504
032052     000
032053     040  050040  020103  DH25:    .ASCIZ  /  PC      RKCS    RKER     RKDS     RKDA   DRVE#/
032060  020040  020040  045522
032066  051503  020040  051040
032074  042513  020122  020040
032102  051040  042113  020123
032110  020040  051040  042113
032116  020101  020040  051104
032124  042526  000043

032130  020040  041520  020040  DH27:    .ASCIZ  /  PC      KEY     FNCTN CODE/
032136  020040  045440  054505
032144  020040  020040  047106
032152  052103  020116  047503
032160  042504     000
032163     040  050040  020103  DH103:   .ASCIZ  /  PC     EXPCT    RECVD/
032170  020040  042440  050130
032176  052103  020040  051040
032204  041505  042126     000

032211     040  050040  020103  DH30:    .ASCIZ  /  PC     KEY/
032216  020040  045440  054505
032224     000
032225     040  050040  020103  DH105:   .ASCIZ  /  PC      RKDA    RKWC/
032232  020040  020040  045522
```

```
032240  040504  020040  051040
032246  053513  000103
032252  020040  041520  020040  DH110:   .ASCIZ  /  PC      RKCS     RKER     RKBA     RKDA/
032260  020040  051040  041513
032266  020123  020040  051040
032274  042513  020122  020040
032302  051040  041113  020101
032310  020040  051040  042113
032316  000101


                                         .EVEN

032320  001116  001162  001164  DT1:     .WORD    $ERRPC,$REG0,$REG1,$REG2,$REG3,0
032326  001166  001170  000000

032334  001116  001162  000000  DT2:     .WORD    $ERRPC,$REG0,0

032342  001116  001162  001164  DT21:    .WORD    $ERRPC,$REG0,$REG1,$REG2,$REG3,$REG4,$REG5,$REG6,0
032350  001166  001170  001172
032356  001174  001176  000000
032364  001116  001162  001164  DT25:    .WORD    $ERRPC,$REG0,$REG1,$REG2,$REG3,$REG4,0
032372  001166  001170  001172
032400  000000
032402  001116  001162  001164  DT103:   .WORD    $ERRPC,$REG0,$REG1,0
032410  000000

                                ;THIS IS THE DATA BUFFER USED TO WRITE THE RANDOM PATTERNS ON THE
                                ;DISK AT THE BEGINING. 400 (OCTAL) WORDS ARE WRITTEN AT A TIME, THUS
                                ;THIS BUFFER IS 400/8 WORDS LONG.

032412                          DBUF:
032412  000240                  PGEND:   NOP
        000001                           .END
```

```
$SCOPE  027102          $SWR   = 143000         $TPFLG  001157          $TTYIN  023402          $TYPOS  025666
$SETUP= 000115          $SWRMK= 000000          $TPS    001150          $TYPDS  023726          $XTSTR  027114
$SIZE   017374          $TKB    001146          $TRAP   027246          $TYPE   024152          $$GET4= 000000
$SIZEX  017636          $TKS    001144          $TRAP2  027270          $TYPEC  024322          $0FILL  026111
$STUP = 177777          $TN    = 000010         $TRP   = 000022         $TYPEX  024370          .      = 032414
$SVLAD  027204          $TNPWR  024622          $TRPAD  027302          $TYPOC  025712
$SVPC = 000220          $TPB    001152          $TSTNM  001102          $TYPON  025726

. ABS.  032414     000

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DZRKHF,DZRKHF/LI:ME/NL:MC:MD:CND/SOL/NSQ_DZRKHF.P11
 RUN-TIME: 64 47 1 SECONDS
 RUN-TIME RATIO: 297/114=2.6
 CORE USED:  30K  (59 PAGES)
```