RK11/05F/J

PERFORMANCE EXERCISER

MD-11-DZRKH-G

EP-DZRKH-G-DL-A

FICHE 1 OF 1

AUG 1977

digital

MADE IN USA

E0F1DZM9ADSEQ	00010000	770712	PDP10 411	37HDR1DZRKHGSEQ	00010000	770712

.REM %

IDENTIFICATION

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

TABLE OF CONTENTS

1.0 ABSTRACT

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

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

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

2.0 REQUIREMENTS

2.1 EQUIPMENT

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

2.2 PRELIMINARY PROGRAMS

SINCE THIS IS A HIGH-LEVEL EXERCISER PROGRAM THE CONTROLLER AND THE DRIVE SHOULD BE FREE OF BASIC FAULTS. IT IS 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/RK05 DYNAMIC TESTS
C. RK05 UTILITY PACKAGE (IF NEEDED)

2.3 EXECUTION TIME

THIS VARIES FROM 30 TO 90 MINUTES FOR A PASS. IT SHOULD BE NOTED THAT THIS IS AN EXERCISER LEVEL PROGRAM AND SHOULD BE PREFERRABLY RUN FOR A LONG PERIOD OF TIME.
NOTE: THE FIRST PASS IS A SHORT (5-20 MINUTE) PASS TO SERVE AS A QUICK VERIFY OF THE RK SUBSYSTEM.

3.0 STARTING ADDRESS

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

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

4.0 PROGRAM CONTROL MODES AND OPERATOR ACTION

PAPER TAPE LOADING
RKDP DUMP MODE
RKDP CHAIN MODE
ACT11

4.1 PAPER TAPE LOADING

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

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

4.1.3 LOAD ADDRESS 200

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

4.1.5 THE PROGRAM IDENTIFIES ITSELF

MAINDEC-11-DZRKH-G

RK11/RK05 PERFORMANCE EXERCISER

THEN IT PROCEEDS TO TEST THE DRIVES.

4.2 RKDP DUMP MODE

4.2.1 THE PROGRAM IS LOADED BY THE RKDP MONITOR.

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

4.2.3 THE PROGRAM IDENTIFIES ITSELF AND PRINTS OUT:

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

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

4.3 RKDP CHAIN MODE

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

'DRIVE 'N' NOT TESTED'

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

4.4 ACT11 MODE

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

5.0 DRIVE SELECTION

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

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

6.0 SWITCH OPTIONS

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' WHENEVER THE PROGRAM ENTERS THE SCOPE ROUTINE OR BEGINS A NEW TEST. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

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

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

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

6.1 SW<15>

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

6.2 SW<13>

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

6.3 SW<12>

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

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

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

6.4 SW<11>

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

6.5 SW<09>

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

6.6 SW<08>

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

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

THE ERROR STATISTICS GIVE THE NUMBER OF ERRORS THAT HAVE OCCURRED

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

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

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

6.7 SW<06>

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

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

LO LIMIT?
HI LIMIT?

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

THIS SWITCH COULD BE QUITE USEFUL IN DETERMINING WHETHER THE PROBLEM IS WITHIN THE RK11 OR OUTSIDE (IN MEMORY). NORMALLY, IF THE PROBLEM IS WITHIN THE RK11, ERRORS WILL KEEP ON 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<05>

THIS SWITCH PROVIDES THE USER A CAPABILITY TO HALT THE PROGRAM AT A KNOWN POINT. THE HALT IS DONE AFTER THE CURRENT SET OF EIGHT COMMANDS IN THE QUEUE HAVE BEEN EXECUTED. THE "HALT" IS LOCATED AT THE BEGINNING OF THE'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 200 OR RESTARTED AT 210.

6.9 SW<04>

THIS SWITCH PROVIDES THE USER WITH AN ABILITY TO SKIP THE TIME CONSUMING REWRITE OF ALL THE DISKS WHEN THE PROGRAM IS RESTARTED AT 210. THIS SWITCH CAN BE USED ONLY WHEN RESTARTING THE PROGRAM AT 210 WITH SW 4 SET. ON RESTARTING THE PROGRAM AT 210, THE INITIAL BOUNDARY CONDITION TESTS (TST1-TST7) ARE SKIPPED. IF SWITCH 4 IS SET, THE REWRITE OF ALL THE DISKS (WHICH WOULD HAVE BEEN NORMALLY DONE) IS ALSO SKIPPED. THE USER IS CAUTIONED TO USE THIS SWITCH CAREFULLY. THE DISKS SHOULD HAVE BEEN WRITTEN WITH RANDOM PATTERNS AT LEAST ONCE BEFORE RESTARTING THE PROGRAM AT 210. IT SHOULD BE NOTED THAT TESTS 1-7 WRITE ON CYLINDERS 0,1. ON RESTART, THE STATISTICS COLLECTED SO FAR ARE SAVED.

6.10 SW<03>

THIS SWITCH ALLOWS THE 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 AR. 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 LOCPING 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.

```
587
588                             .TITLE  MD-11-DZRKHG, RK11-RK05 PERFORMANCE EXERCISER
589                             ;*COPYRIGHT (C) 1973,1977
590                             ;*DIGITAL EQUIPMENT CORP.
591                             ;*MAYNARD, MASS. 01754
592                             ;*
593                             ;*PROGRAM BY JIM KAPADIA
594                             ;*
595                             ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
596                             ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
597                             ;*
598                             ;*REVISED BY GEORGE GALLANT,TOM SAWYER   -  MARCH 1976
599                             ;*REVISED BY CHUCK HESS - AUGUST 1976
600                             .SBTTL  OPERATIONAL SWITCH SETTINGS
601                             ;*
602                             ;*      SWITCH                  USE
603                             ;*      ------          --------------------
604                             ;*        15            HALT ON ERROR
605                             ;*        14            LOOP ON TEST
606                             ;*        12            TYPE OUT ERROR HISTORY
607                             ;*        10            BELL ON ERROR
608                             ;*         9            LOOP ON ERROR
609                             ;*         8            TYPE OUT ERROR AND TRANSFER DATA STATISTICS
610                             ;*         6            SELECT BUS ADDRESS LIMITS FOR DISK DATA TRANSFERS
611                             ;*         5            HALT BEFORE DOING NEXT SET OF COMMANDS(GENERQ)
612                             ;*         4            DO NOT REWRITE THE DISKS ON RESTART AT 21C
613                             ;*         3            TYPE OUT ELAPSED TIME AT ERROR
614                             ;*         2            DROP DRIVE AFTER MAXM ERORS ON THIS DRIVE
615                             ;*         1            TYPE SERIAL NUMBER OF ERRORING DRIVE
616                             ;*         0            IF SW8=1, ONLY TYPE ELAPSED TIME
617                             ;*        11            DUMP OUT ALL RK11 REGISTERS ON ERROR
618
619
620                             ;*      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
                                        $SVPC=.                 ;SAVE PC
                                        .=46
                                        $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                                        .=52
                                        .WORD   0               ;;2)SET LOC.52 TO ZERO
                                        .=$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 REGISTERS
                                
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
```

```
                                RKCS:   .WORD   177404
                                RKWC:   .WORD   177406
                                RKBA:   .WORD   177410
                                RKDA:   .WORD   177412
                                RKDB:   .WORD   177416
                                KWLS:   .WORD   177546  ;STATUS REGISTER FOR KW11L

                                PCNTR:  .WORD   250.

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

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

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


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
```

```
                        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 KEYS (TAKING KEY AS BASE).  KEY #= OFFSET(0-3)/2 + 1

                        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.

                        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.

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

                        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)

                        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


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


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

                        TIMER:  .WORD   0
                        ERCODE: .WORD   0
                        DRVPTR: .WORD   0
                        DRVCNT: .WORD   0


                        QDRV:   .WORD   0       ;TEMPORARY REGISTERS USED BY 'GENRQ'
                        QCYL:   .WORD   0       ;ROUTINE TO STORE VARIOUS PARAMETERS
                        QSUR:   .WORD   0       ;OF A COMMAND AS THEY ARE GENERATED.
                        QSEC:   .WORD   0
                        QFNC:   .WORD   0
                        QBUSAD: .WORD   0
                        QWRCNT: .WORD   0
```

```
1040
1041
1042                                    ;THIS TABLE CONTAINS VARIOUS MAPPING FACTORS TO BE USED
1043                                    ;FOR GENERATING RANDOM PARAMETERS FROM RANDOM NUMBERS
1044
1045  001520  000000                    DRMAP:   .WORD   0       ;MAPPING FACTOR FOR GENERATING RANDOM DRIVE NUMBER
1046  001522  000000                    CYLMAP:  .WORD   0       ;MAPPING FACTOR FOR CYLINDER
1047  001524  000000                    SECMAP:  .WORD   0       ;MAPPING FACTOR FOR SECTOR
1048  001526  000000                    FNMAP:   .WORD   0       ;MAPPING FACTOR FOR FUNCTION
1049  001530  000000                    BAMAP:   .WORD   0       ;MAPPING FACTOR FOR BUS ADDRESS
1050  001532  000000                    WCMAP:   .WORD   0       ;MAPPING FACTOR FOR WORD COUNT
1051
1052
1053                                    ;THESE TWO FLAGS CORRESPOND TO THE 2 INTERRUPT HANDLERS (RK11) USED
1054                                    ;IN THIS PROGRAM. WHEN THE INTERRUPT HANDLER IS ENTERED THE FLAG IS
1055                                    ;CLEARED OR SET.
1056
1057  001534      000                   INTFLG: .BYTE   0       ;FOR 'INTHND ', CLEARED ON ENTERING HANDLER
1058  001535      000                   INT1FL: .BYTE   0       ;FOR 'INT1SK', SET ON ENTERING HANDLER
1059
1060  001536  000000                    SAVKEY: .WORD   0
1061  001540  000000                    ECOUNT: .WORD   0
1062
1063                                    ;THIS TABLE CONTAINS COUNTS FOR THE NUMBER OF OF ERRORS OCCURING ON A
1064                                    ;DRIVE (NOTE: ONLY THOSE ERRORS WHICH ARE POSITIVELY ATTRIBUTABLE TO A
1065                                    ;SPECIFIC DRIVE). THE COUNT  KEPT ONLY IF SWITCH 2 IS SET, WHEN THE COUNT
1066                                    ;REACHES THE MAXIMUM ALLOWABLE (USUALLY 3) THE DRIVE IS DROPPED FROM
1067                                    ;TESTING AND IS TAKEN OUT OF THE DRIVE SELECTION TABLE.
1068
1069  001542  000010                    ERDRV:   .BLKB   10      ;COUNT FOR DRIVES 0-7
1070
1071  001552  000000                    KWHR:    .WORD   0       ;COUNTS HOURS (2'S COMPLEMENT)
1072  001554  000000                    KWMIN:   .WORD   0       ;COUNTS MINUTES (2'S COMPLEMENT)
1073  001556  000000                    KWSEC:   .WORD   0       ;COUNTS SECONDS (2'S COMPLEMENT)
1074  001560  000000                    KWCOUNT: .WORD   0       ;COUNTS CPS FROM KW11L (2'S COMPLMNT)
1075
1076                                    ;THIS TABLE CONTAINS COUNTS FOR HARD ERRORS ON A PARTICULAR DRIVE.
1077                                    ;EX HECN2 WILL CONTAIN THE TOTAL NUMBER OF HARD ERRORS THAT OCCURED ON
1078                                    ;DRIVE 2
1079
1080  001562  000010                    HECN:    .BLKW   10      ;DRIVE 0-7 HARD ERROR COUNTS
1081
1082                                    ;THIS TABLE CONTAINS COUNTS FOR SEEK ERRORS
1083                                    ;ON A PARTICULAR DRIVE.
1084
1085  001602  000010                    SKECN:   .BLKW   10      ;DRIVE 0-7 SEEk ERROR COUNTS
1086
1087
1088                                    ;THIS TABLE CONTAINS COUNTS FOR SIN ERRORS ON A
1089                                    ;PARTICULAR DRIVE
1090
1091  001622  000010                    SINCN:   .BLKB   10      ;DRIVE 0-7 SIN COUNTS
1092
1093                                    ;THIS TABLE CONTAINS COUNTS FOR WRITE CHECK ERRORS
1094                                    ;THAT OCCURED ON A PARTICULAR DRIVE
1095
```

```
1096  001632  000010                WCECN:  .BLKW   10      ;WCE COUNT FOR DRIVES 0-7
1097
1098
1099                                ;THIS TABLE CONTAINS COUNTS FOR CHECK SUM ERROR THAT
1100                                ;OCCURED ON A PARTICULAR DRIVE
1101
1102  001652  000010                CSECN:  .BLKW   10      ;CSE COUNT FOR DRIVES 0-7
1103
1104
1105                                ;THIS TABLE CONTAINS COUNT OF NUMBER OF FUNCTIONS
1106                                ;THAT WERE ABORTED ON A PARTICULAR DRIVE. A
1107                                ;FUNCTION IS ABORTED ONLY AFTER DOING RETRIES
1108
1109  001672  000010                ABORT:  .BLKW   10      ;ABORT COUNT FOR DRIVES 0-7
1110
1111                                ;COUNTS FOR NUMBER OF DATA ERRORS THAT OCCURED ON INDIVIDUAL DRIVES.
1112
1113  001712  000010                DATER:  .BLKW   10      ;DRIVES 0-7
```

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

```
1170  002222  036501     000
1171  002225     015  047012  020117  MSG14:  .ASCIZ  <15><12>/NO DRVS PRSNT/
1172  002232  051104  051526  050040
1173  002240  051522  052116     000
1174  002245     015  042012  053122  MSG15:  .ASCIZ  <15><12>/DRVE # DIDN'T INTERUPT AFTER /
1175  002252  020105  020043  044504
1176  002260  047104  052047  044440
1177  002266  052116  051105  050125
1178  002274  020124  043101  042524
1179  002302  020122     000
1180  002305     015  045412  054505  MSG16:  .ASCIZ  <15><12>/KEY-8   BUSY-7/
1181  002312  034055  020040  041040
1182  002320  051525  026531  000067
1183  002326  051440  020122  047516  MSG17:  .ASCII  / SR NO/
1184  002334  000072                  MSG18:  .ASCIZ  /:/
1185  002336  005015  042040  047522  MSG19:  .ASCIZ  <15><12>/ DROPPED DRIVE # /
1186  002344  050120  042105  042040
1187  002352  044522  042526  021440
1188  002360  000040
1189  002362  005015  051104  053111  MSG20:  .ASCIZ  <15><12>/DRIVE/
1190  002370  000105
1191  002372  020054     000          MSG24:  .ASCIZ  / /
1192  002375     106     000          MSG25:  .ASCIZ  /F/
1193  002377     015  042012  044522  MSG26:  .ASCIZ  <15><12>/DRIVE  WRDS WRITN  WRDS READ    CSE    WCE DATERR     HE  /
1194  002404  042526  020040  051127
1195  002412  051504  053440  044522
1196  002420  047124  020040  051127
1197  002426  051504  051040  040505
1198  002434  020104  020040  041440
1199  002442  042523  020040  020040
1200  002450  041527  020105  040504
1201  002456  042524  051122  020040
1202  002464  020040  044040  020105
1203  002472  000040
1204  002474  005015  051104  053111  MSG26A: .ASCIZ  <15><12>/DRIVE     SKE  ABORT    SIN/
1205  002502  020105  020040  020040
1206  002510  045523  020105  040440
1207  002516  047502  052122  020040
1208  002524  020040  044523  000116
1209  002532  005015  047125  041101  MSG27:  .ASCIZ  <15><12>/UNABLE TO CLEAR ERROR AFTER THREE TRIES/
1210  002540  042514  052040  020117
1211  002546  046103  040505  020122
1212  002554  051105  047522  020122
1213  002562  043101  042524  020122
1214  002570  044124  042522  020105
1215  002576  051124  042511  000123
1216  002604  005015  051105  047522  MSG28:  .ASCIZ  <15><12>/ERROR CONDITION CLEARED ON RETRY # /
1217  002612  020122  047503  042116
1218  002620  052111  047511  020116
1219  002626  046103  040505  042522
1220  002634  020104  047117  051040
1221  002642  052105  054522  021440
1222  002650  000040
1223  002652  005015  044524  042515  MSG29:  .ASCIZ  <15><12>/TIME /
1224  002660  000040
1225
```

```
1226  002662      040                 BLNKS3: .ASCII  / /
1227  002663      040                 BLNKS2: .ASCII  / /
1228  002664   000040                 BLNKS1: .ASCIZ  / /
1229                                          .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  027632                          EM1     ;ERROR ON WRITE
002670  031730                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
002672  032422                          DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
002674  000000                          0

                                ;ITEM   2

002676  027650                          EM2     ;ATTEMPT TO INITIATE FUNCTION ON 'BUSY' DRIVE
002700  031776                          DH2     ;PC     DRIVE
002702  032436                          DT2     ;$ERRPC $REG0
002704  000000                          0

                                ;ITEM   3

002706  027723                          EM3     ;CONTROL READY NOT SET
002710  031730                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
002712  032422                          DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
002714  000000                          0

                                ;ITEM   4

002716  027746                          EM4     ;R/W/S READY NOT SET
002720  031730                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
002722  032422                          DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
002724  000000                          0

                                ;ITEM   5

002726  027771                          EM5     ;CONTROL READY NOT SET AFTER FIRST INTERRUPT ON ISSUING SEEK
002730  031730                          DH1     ;PC     RKCS    RKER    RKDS    RKDA
002732  032422                          DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
```

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

```
1342  003032  032422                  DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
1343  003034  000000                  0
1344
1345                          ;ITEM   16
1346
1347  003036  030603                  EM16    ;'SIN' ERROR
1348  003040  031730                  DH1     ;PC     RKCS    RKER    RKDS    RKDA
1349  003042  032422                  DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
1350  003044  000000                  0
1351
1352                          ;ITEM   17
1353
1354  003046  030614                  EM17    ;'ERR'OR ON DOING SEEK
1355  003050  031730                  DH1     ;PC     RKCS    RKER    RKDS    RKDA
1356  003052  032422                  DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
1357  003054  000000                  0
1358
1359                          ;ITEM   20
1360
1361  003056  030642                  EM20    ;SCP DID NOT SET AFTER SEEK WAS DONE
1362  003060  031730                  DH1     ;PC     RKCS    RKER    RKDS    RKDA
1363  003062  032422                  DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
1364  003064  000000                  0
1365
1366                          ;ITEM   21
1367
1368  003066  030706                  EM21    ;SOFT ERROR
1369  003070  032013                  DH21    ;$ERRPC RKCS    RKER    RKDS    RKDA:   DRV#
1370                                          ;CYL    SUR     SEC
1371  003072  032444                  DT21    ;$ERRPC $REG0   $REG1   $REG2           $REG3
1372                                          ;$REG4  $REG5   $REG6
1373  003074  000000                  0
1374
1375                          ;ITEM   22
1376
1377  003076  000000                  0
1378  003100  032013                  DH21    ;$ERRPC RKCS    RKER    RKDS    RKDA:   DRV#
1379                                          ;CYL    SUR     SEC
1380  003102  032444                  DT21    ;$ERRPC $REG0   $REG1   $REG2           $REG3
1381                                          ;$REG4  $REG5   $REG6
1382  003104  000000                  0
1383
1384                          ;ITEM   23
1385
1386  003106  030720                  EM23    ;DATA (COMPARISON) ERROR
1387  003110  032110                  DH23    ;PC     RKBA    EXPCT   RECVD   RKDA
1388  003112  032422                  DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
1389  003114  000000                  0
1390
1391                          ;ITEM   24
1392
1393  003116  030747                  EM24    ;CONTROL READY CLEAR ON  INTERRUPT AFTER RK FUNCTION
1394  003120  031730                  DH1     ;PC     RKCS    RKER    RKDS    RKDA
1395  003122  032422                  DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
1396  003124  000000                  0
1397
```

```
1398                                    ;ITEM   25
1399
1400  003126  000000                            0
1401  003130  032155                    DH25    ;PC     RKCS    RKER    RKDS    RKDA    DRIVE #
1402  003132  032466                    DT25    ;$ERRPC $REG0   $REG1   $REG2   $REG3   $REG4
1403  003134  000000                            0
1404
1405                                    ;ITEM   26
1406
1407  003136  031023                    EM26    ;STUCK IN LOOP, 8 Q-COMMANDS SHOULD BE DONE BY NOW
1408  003140  031730                    DH1     ;PC     RKCS    RKER    RKDS    RKDA
1409  003142  032422                    DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
1410  003144  000000                            0
1411
1412                                    ;ITEM   27
1413
1414  003146  031076                    EM27    ;ATTEMPT TO DO WRITE CHECK BEFORE WRITE
1415  003150  032232                    DH27    ;PC     KEY     FUNCTION CODE
1416  003152  032504                    DT103   ;PC     $REG0   $REG1
1417  003154  000000                            0
1418
1419                                    ;ITEM   30
1420
1421  003156  031137                    EM30    ;ATTEMPT TO REEXECUTE A COMMAND IN PROGRESS OR ALREADY FINISHED
1422  003160  032313                    DH30    ;PC     KEY
1423  003162  032436                    DT2     ;$ERRPC $REG0
1424  003164  000000                            0
1425
1426                                    ;ITEM   31
1427
1428  003166  031232                    EM31    ;'FUNCTION IN PROGRESS' FLAG FOR INTERRUPTING DRIVE IS NOT SET
1429  003170  031776                    DH2     ;PC     DRIVE
1430  003172  032436                    DT2     ;$ERRPC $REG0
1431  003174  000000                            0
1432
1433                                    ;ITEM   32
1434
1435  003176  031323                    EM32    ;UNEXPECTED DRIVE INTERRUPTED
1436  003200  032265                    DH103   ;PC     EXPCT   RECVD
1437  003202  032504                    DT103   ;$ERRPC $REG0   $REG1
1438  003204  000000                            0
1439
1440                                    ;ITEM   33
1441
1442  003206  031355                    EM33    ;WRONG FUNCTION CODE IN RKCS AFTER INTERRUPT
1443  003210  032265                    DH103   ;PC     EXPCT   RECVD
1444  003212  032504                    DT103   ;$ERRPC $REG0   $REG1
1445  003214  000000                            0
1446
1447                                    ;ITEM   34
1448
1449  003216  031432                    EM34    ;DRIVE READY CLEAR
1450  003220  031730                    DH1     ;PC     RKCS    RKER    RKDS    RKDA
1451  003222  032422                    DT1     ;$ERRPC $REG0   $REG1   $REG2   $REG3
1452  003224  000000                            0
1453
```

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

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

```
1541                                            ;IF POWER FAILED, ON RETURN OF POWER ENTER HERE.
1542
1543  003366  004737  022630          PFSTRT: JSR     PC,@#TIME       ;WAIT SOME TIME
1544  003372  105237  001253                  INCB    FRSTRT          ;INDICATE THAT THE STATISTICS HAVE
1545                                          ;TO BE SAVED, ON RETRN FROM PWR FAIL.
1546
1547
1548
1549
1550  003376  000005                  START:  RESET                   ;CLEAR THE BUS
1551                                  ;;GIVE DRIVES TIME TO LOAD IF PROGRAM WAS STARTED BY APT
1552  003400  023737  000042  000046          CMP     @#42,@#46
1553  003406  001016                          BNE     STARTA
1554  003410  005077  175614                  CLR     @RKDA           ;SELECT UNIT 0
1555  003414  012700  000250                  MOV     #250,R0         ;WAIT FOR..
1556  003420  032777  000200  175570  20$:    BIT     #200,@RKDS      ;DRIVE READY..
1557  003426  001006                          BNE     STARTA          ;IN CASE..
1558  003430  005001                          CLR     R1              ;OF APT..
1559  003432  005301                          DEC     R1              ;START, BUT..
1560  003434  001376                          BNE     .-2             ;DON'T WAIT..
1561  003436  005300                          DEC     R0              ;FOREVER.
1562  003440  001367                          BNE     20$
1563  003442  000000                          HALT                    ;RKDS BIT 7 (DRIVE READY) NEVER SET.
1564  003444                          STARTA:
1565                                  .SBTTL  INITIALIZE THE COMMON TAGS
1566                                  ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1567  003444  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1568  003450  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
1569  003452  022706  001140                  CMP     #SWR,R6 ;;DONE?
1570  003456  001374                          BNE     .-6             ;;LOOP BACK IF NO
1571  003460  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
1572                                  ;;INITIALIZE A FEW VECTORS
1573  003464  012737  027204  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1574  003472  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
1575  003500  012737  027350  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1576  003506  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
1577  003514  012737  027450  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1578  003522  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
1579  003530  012737  003530  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1580  003536  012737  003536  001110          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
1581                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1582                                  ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1533  003544  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
1584  003550  012737  003604  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
1585  003556  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
1586  003564  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
1587  003572  022777  177777  175340          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
1588  003600  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1589                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
1590  003602  000403                          BR      65$             ;;BRANCH IF  ) TIMEOUT
1591  003604  012716  003612          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
1592  003610  000002                          RTI
1593  003612  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1594  003620  012737  000174  001142          MOV     #DISPREG,DISPLAY
1595  003626  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1596
```

```
1597  003632  023737  000042  000046          CMP     @#42,@#46       ;ARE WE IN ACT11 AUTOMATIC MODE?
1598  003640  001416                          BEQ     69$             ;YES, SKIP TITLE PRINTOUT
1599                                  .SBTTL  TYPE PROGRAM NAME
1600                                  ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1601  003642  005227  177777                  INC     #-1             ;;FIRST TIME?
1602  003646  001052                          BNE     67$             ;;BRANCH IF NO
1603  003650  104401  003706                  TYPE    ,68$            ;;TYPE ASCIZ STRING
1604                                  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1605  003654  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
1606  003660  001006                          BNE     69$             ;;BRANCH IF YES
1607  003662  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
1608  003670  001005                          BNE     70$             ;;BRANCH IF NO
1609  003672  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
1610  003674  000403                          BR      70$
1611  003676  112737  000001  001134  69$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
1612  003704                          70$:
1613  003704  000433                          BR      67$             ;;GET OVER THE ASCIZ
1614                                  ;;68$:  .ASCIZ  <CRLF>*RK11/RKOS PERFORMANCE EXERCISER*<15><12>*MAINDEC-11-DZRKH-G*<CRLF
1615  003774                          67$:
1616  003774  012737  026206  000030          MOV     #$ERROR,@#EMTVEC        ;EMT VECTOR FOR ERROR ROUTINE
1617  004002  013737  001244  000032          MOV     PPRLVL,@#EMTVEC+2       ;LEVEL 5
1618  004010  012737  022552  000100          MOV     #KWSRVE,@#KWLVEC        ;KW11L CLOCK SERVICE
1619  004016  013737  001246  000102          MOV     KWPLVL,@#KWLVEC+2       ;LEVEL 7
1620
1621                                  ;THE FOLLOWING CODE FINDS OUT THE PROGRAM CONTROL MODE:
1622                                  ;PAPER TAPE (MANUAL), ACT11, RKDP CHAIN OR DUMP
1623
1624
1625  004024  005037  002060                  CLR     XXDPMD          ;CLEAR 'XXDP' LOAD DEVICE STORAGE
1626  004030  122737  000002  000041          CMPB    #2,41           ;LOADED FROM AN RKOS ?
1627  004036  001160                          BNE     ST2             ;BR IF NOT
1628  004040  013737  000040  002060          MOV     40,XXDPMD       ;GET DEVICE INDICATOR AND DRIVE ADDRESS OF
1629                                                                  ;LOADING RKOS
1630  004046  122737  000010  002060          CMPB    #10,XXDPMD      ;VALID DRIVE ADDRESS ?
1631  004054  101002                          BHI     2$              ;BR IF YES
1632  004056  105037  002060                  CLRB    XXDPMD          ;CHANGE TO DRIVE ZERO
1633  004062  005737  000042          2$:     TST     42              ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
1634  004066  001424                          BEQ     3$              ;BR IF NEITHER
1635  004070  104401  004076                  TYPE    ,72$            ;;TYPE ASCIZ STRING
1636  004074  000413                          BR      71$             ;;GET OVER THE ASCIZ
1637                                  ;;72$:  .ASCIZ  <15><12>/NOT TESTING DRIVE /
1638  004124                          71$:
1639  004124  005046                          CLR     -(SP)           ;CLEAR WORD ON STACK
1640  004126  113716  002060                  MOVB    XXDPMD,(SP)     ;GET DRIVE ADDRESS
1641  004132  104403                          TYPOS                   ;TYPE THE ADDRESS
1642  004134     001                          .BYTE   1               ;ONLY 1 CHARACTER
1643  004135     000                          .BYTE   0               ;SUPRESS LEADING ZEROS
1644  004136  000520                          BR      ST2             ;GET NUMBER OF DRIVES
1645  004140  005227  177777          3$:     INC     #-1             ;FIRST TIME THROUGH HERE ?
1646  004144  001115                          BNE     ST2             ;BR IF NOT
1647  004146  104401  004154                  TYPE    ,74$            ;;TYPE ASCIZ STRING
1648  004152  000411                          BR      73$             ;;GET OVER THE ASCIZ
1649                                  ;;74$:  .ASCIZ  <15><12>/TO TEST DRIVE /
1650  004176                          73$:
1651  004176  005046                          CLR     -(SP)           ;CLEAR WORD ON THE STACK
1652  004200  113716  002060                  MOVB    XXDPMD,(SP)     ;GET DRIVE ADDRESS
```

```
1653  004204  104403                          TYPOS                   ;TYPE THE DRIVE ADDRESS
1654  004206     001                          .BYTE   1               ;ONLY 1 CHARACTER
1655  004207     000                          .BYTE   0               ;SUPRESS LEADING ZEROS
1656  004210  104401  004216                  TYPE    ,76$            ;;TYPE ASCIZ STRING
1657  004214  000431                          BR      75$             ;;GET OVER THE ASCIZ
1658                                  ;;76$:  .ASCIZ  / HALT PROGRAM, REMOVE RKDP PACK AND REPLACE IT/<15><12>
1659  004300                          75$:
1660  004300  104401  004306                  TYPE    ,78$            ;;TYPE ASCIZ STRING
1661  004304  000435                          BR      77$             ;;GET OVER THE ASCIZ
1662                                  ;;78$:  .ASCIZ  /WITH A WORK PACK, CLEAR LOCATION 40, AND RESTART PROGRAM/
1663  004400                          77$:
1664
1665
1666
1667  004400  104416                  ST2:    CON.RESET
1668  004402  005037  001264                  CLR     DRVPRS          ;FIND WHICH DRIVE #'S ARE PRESENT
1669  004406  005000                          CLR     R0
1670  004410  005002                          CLR     R2
1671  004412  005037  001254                  CLR     PDR             ;CLEAR DRIVES PRESENT TABLE
1672  004416  005037  001256                  CLR     PDR+2
1673  004422  005037  001260                  CLR     PDR+4
1674  004426  005037  001262                  CLR     PDR+6
1675  004432  012701  001254                  MOV     #PDR,R1
1676  004436  012703  001306                  MOV     #KEY,R3
1677  004442  010277  174562          1$:     MOV     R2,@RKDA        ;SELECT A DRIVE
1678  004446  032777  000200  174542          BIT     #200,@RKDS      ;IS IT IN SYSTEM?
1679  004454  001415                          BEQ     3$              ;NO
1680  004456  010237  001502                  MOV     R2,QDRV         ;LOAD DRIVE ADDRESS INTO QDRV
1681  004462  104420                          DRV.RESET               ;RESET THE DRIVE
1682  004464  005737  002060                  TST     XXDPMD          ;PROGRAM LOADED FROM AN RK05 ?
1683  004470  001403                          BEQ     2$              ;BR IF NOT
1684  004472  120037  002060                  CMPB    R0,XXDPMD       ;LOADED FROM THIS RK05 ?
1685  004476  001404                          BEQ     3$              ;BR IF YES
1686  004500  110021                  2$:     MOVB    R0,(R1)+        ;STORE THE DRIVE NUMBER
1687  004502  010223                          MOV     R2,(R3)+        ;STORE ADDRESS IN KEY TABLE
1688  004504  005237  001264                  INC     DRVPRS          ;BUMP THE NUMBER OF DRIVES COUNTER
1689  004510  062702  020000          3$:     ADD     #20000,R2       ;NEXT DRIVE ADDRESS
1690  004514  005200                          INC     R0              ;NEXT DRIVE NUMBER
1691  004516  022700  000010                  CMP     #8.,R0          ;DONE ALL DRIVES???
1692  004522  001347                          BNE     1$              ;LOOP TILL DONE
1693  004524  013703  001264                  MOV     DRVPRS,R3       ;FIND WHICH DRIVES ARE TYPE F
1694  004530  001510                          BEQ     ST4             ;BR IF NOT DRIVES PRESENT
1695  004532  012701  001254                  MOV     #PDR,R1
1696  004536  005000                          CLR     R0
1697  004540  005002                          CLR     R2
1698  004542  104401  002362                  TYPE    ,MSG20
1699  004546  111102                  4$:     MOVB    (R1),R2         ;GET DRIVE NUMBER
1700  004550  010200                          MOV     R2,R0
1701  004552  002472                          BLT     9$
1702
1703  004554  104401  002372                  TYPE    ,MSG24
1704
1705  004560  010246                          MOV     R2,-(SP)        ;TYPE THE DRIVE NUMBER
1706  004562  104403                          TYPOS
1707  004564     001                          .BYTE   1
1708  004565     000                          .BYTE   0
```

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

```
1757                                    ;'MAXBA' IS THE HIGHEST BUS ADDRESS (MEMORY) TO WHICH DATA TRANSFERS CAN
1758                                    ;BE DONE BY THE PROGRAM.
1759                                    ;'MAXBA' IS FIGURED USING THE FOLLOWING ALGORITHM:
1760
1761                                    ;1.  IF KT11 IS NOT PRESENT,
1762                                    ;A.  AND THE PROGRAM IS RUN UNDER XXDP, THEN THE TOP 1.5 K IS RESERVED
1763                                    ;AND THE  'MAXBA' IS COMPUTED ($LSTAD-6000).
1764                                    ;B.  AND THE PROGRAM IS NOT RUNNING UNDER XXDP, THEN THE TOP 320 WORDS
1765                                    ;ARE RESERVED FOR 'MOM',LOADER,ETC. AND THE 'MAXBA' IS COMPUTED ($LSTAD-500).
1766
1767                                    ;2. IF KT11 IS PRESENT,
1768                                    ;A.  AND MORE THAN 28K MEMORY IS PRESENT, THEN THE MAXIMUM BUS ADDRESS
1769                                    ;IS 147776 (OCTAL).
1770                                    ;B.  AND LESS THAN 28K IS PRESENT, THEN THE TOP 2K IS RESERVED FOR RKDP
1771                                    ;MONITOR AND 'MAXBA' IS COMPUTED.
1772                                    ;FIGURE OUT THE AVAILABLE MEMORY AND 'MAXBA'
1773
1774  004752  004737  017466        ST4:    JSR     PC,$SIZE        ;GO SIZE THE MEMORY
1775  004756  012702  002052                MOV     #BASEBA,R2      ;INITIALIZE POINTERS
1776  004762  012703  002054                MOV     #MAXBA,R3
1777  004766  005737  017524                TST     $KT11           ;KT11 AVAILABLE?
1778  004772  100022                        BPL     4$              ;NO
1779  004774  013700  017772                MOV     $LSTBK,R0       ;GET THE LAST BANK OF MEMORY
1780  005000  020027  001540                CMP     R0,#1540        ;28K OR MORE?
1781  005004  002012                        BGE     3$              ;YES
1782
1783  005006  162700  000040                SUB     #40,R0          ;BACK UP 2 K'S (RKDP MONITOR, ETC.)
1784  005012  012701  177772                MOV     #-6,R1          ;AND FORM THE MAXIMUM BUS ADDRESS
1785                                                                  ;FOR DATA TRANSFER
1786  005016  006300                1$:     ASL     R0
1787  005020  005201                        INC     R1
1788  005022  001375                        BNE     1$
1789  005024  162700  000002                SUB     #2,R0
1790  005030  000415                2$:     BR      6$
1791
1792  005032  012713  147776        3$:     MOV     #147776,(R3)    ;FOR 28K OR MORE, THIS IS THE 'MAXBA'
1793  005036  000413                        BR      7$
1794
1795  005040  013700  017770        4$:     MOV     $LSTAD,R0       ;KT11 NOT PRESENT, GET THE LAST
1796                                                                  ;AVAILABLE ADDRESS
1797
1798  005044  005737  000040        5$:     TST     @#40            ;'XXDP' LOADED PROGRAM ?
1799  005050  001003                        BNE     8$              ;YES
1800  005052  162700  000500                SUB     #500,R0         ;NO, SAVE THE LAST 320 WORDS
1801  005056  000402                        BR      6$
1802  005060  162700  006000        8$:     SUB     #6000,R0        ;SAVE THE LAST 1.5K OF MEMORY (RKDP
1803                                                                  ;MONITOR, ETC.)
1804  005064  010013                6$:     MOV     R0,(R3)         ;SAVE THE MAXIMUM BUS ADDRESS(MAXBA) TO
1805                                                                  ;WHICH DATA TRANSFER CAN BE DONE SAFELY
1806  005066  012712  032514        7$:     MOV     #PGEND,(R2)     ;'BASEBA'
1807  005072  032777  000100  174040        BIT     #SW06,@SWR
1808  005100  001510                        BEQ     ST3
1809  005102  104401  005110                TYPE    ,65$            ;;TYPE ASCIZ STRING
1810  005106  000432                        BR      64$             ;;GET OVER THE ASCIZ
1811                                ;;65$:  .ASCIZ  <15><12>/TYPE OCTAL BUS ADDRESSES FOR DATA XFER, BETWEEN
1812  005174                        64$:
```

```
1813  005174  011246                          MOV     (R2),-(SP)      ;'BASEBA'
1814  005176  104402                          TYPOC
1815  005200  104401  005206                  TYPE    ,67$            ;;TYPE ASCIZ STRING
1816  005204  000402                          BR      66$             ;;GET OVER THE ASCIZ
1817                                  ;;67$:  .ASCIZ  / & /
1818  005212                          66$:
1819  005212  011346                          MOV     (R3),-(SP)      ;'MAXBA'
1820  005214  104402                          TYPOC
1821  005216                          9$:
1822  005216  104401  005224                  TYPE    ,69$            ;;TYPE ASCIZ STRING
1823  005222  000407                          BR      68$             ;;GET OVER THE ASCIZ
1824                                  ;;69$:  .ASCIZ  <15><12>/LO LIMIT? /
1825  005242                          68$:
1826  005242  104412                          RDOCT
1827  005244  012600                          MOV     (SP)+,R0
1828  005246  020012                          CMP     R0,(R2)         ;CORRECT LO LIMIT?
1829  005250  103762                          BLO     9$
1830  005252  020013                          CMP     R0,(R3)         ;CORRECT LO LIMIT?
1831  005254  103360                          BHIS    9$
1832  005256  010012                          MOV     R0,(R2)         ;'BASEBA'
1833  005260                          10$:
1834  005260  104401  005266                  TYPE    ,71$            ;;TYPE ASCIZ STRING
1835  005264  000407                          BR      70$             ;;GET OVER THE ASCIZ
1836                                  ;;71$:  .ASCIZ  <15><12>/HI LIMIT? /
1837  005304                          70$:
1838  005304  104412                          RDOCT
1839  005306  012600                          MOV     (SP)+,R0
1840  005310  020013                          CMP     R0,(R3)         ;CORRECT HI LIMIT?
1841  005312  101362                          BHI     10$
1842  005314  020012                          CMP     R0,(R2)         ;CORRECT LO LIMIT?
1843  005316  101760                          BLOS    10$
1844  005320  010013                          MOV     R0,(R3)         ;'MAXBA'
1845
1846  005322  023727  002054  037476  ST3:   CMP     MAXBA,#37476    ;8K MEMORY - CLOBBER XXDPT
1847  005330  002003                          BGE     1$
1848  005332  012737  037476  002054          MOV     #37476,MAXBA    ;BUT SAVE LOADER
1849  005340  105737  001253          1$:     TSTB    FRSTRT          ;PROGRAM RESTARTED AT 210?
1850  005344  001402                          BEQ     BCTST           ;NO
1851  005346  000137  007772                  JMP     EXRCSR          ;YES, SKIP TEST 1 TO 7
```

```
1852                                        ;THIS IS THE BEGINING OF THE CONSTRAINED TESTS AIMED AT CHECKING THE
1853                                        ;DIFFERENT BOUNDARY CONDITIONS OF RK11/RK05
1854
1855                                        ;FIND OUT THE DRIVE NUMBER TO BE TESTED.
1856  005352  012737  001306  001476  BOTST:  MOV     #KEY,DRVPTR     ;INITIALIZE PTR TO DRV#
1857  005360  013737  001264  001500          MOV     DRVPRS,DRVCNT   ;NUMBER OF DRIVES PRESENT
1858  005366  017737  174104  001502  NXTDRV: MOV     @DRVPTR,GDRV    ;SAVE DRIVE #(BITS 15-13)
1859  005374  062737  000002  001476          ADD     #2,DRVPTR       ;INCRMENT PTR TO NXT DRV#
1860  005402  005337  001500                  DEC     DRVCNT          ;DONE ALL DRIVES?
1861  005406  100002                          BPL     TST1            ;NO, GO TEST THIS DRIVE
1862  005410  000137  007772                  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.
                                ;;*****************************************************************
005414  000004                  TST1:   SCOPE

005416  013701  001502                  MOV     @DRV,R1         ;GET RKDA
005422  010102                          MOV     R1,R2           ;SAVE RKDA
005424  062702  000002                  ADD     #2,R2           ;EXPCTD RKDA AFTER WRITE IS DONE

005430  012737  005436  001110          MOV     #1$,$LPERR      ;RETURN ADDRESS FOR LUPING

005436  104416                  1$:     CON.RESET
005440  104420                          DRV.RESET
005442  104416                          CON.RESET               ;CLEAR MASK BITS IN POLLING LOGIC
005444  012737  111111  032514  2$:     MOV     #111111,DBUF    ;PATTERN TO BE WRITTEN
005452  012703  000401                  MOV     #401,R3         ;WORD COUNT FOR WRITE
005456  004737  006304                  JSR     PC,DOWRITE      ;GO DO WRITE

005462  104101                          ERROR   101             ;INTERRUPT DID NOT OCCUR AFTER WRITE
005464  004737  020112                  JSR     PC,CHKCS        ;CHECK ERROR BIT IN RKCS
005470  104102                          ERROR   102             ;ERROR BIT IN RKCS SET ON DOING WRITE
005472  004737  020126                  JSR     PC,CHKDA        ;CHECK IF RKDA INCREMENTED RIGHT
005476  104103                          ERROR   103             ;RKDA DID NOT INCREMENT RIGHT AFTER
                                                                ;A WRITE OF 401 WORDS.
005500  004737  020230                  JSR     PC,CHKWC        ;CHECK IF RKWC OVERFLOWED TO 0
005504  104105                          ERROR   105             ;RKWC DID NOT OVERFLOW TO 0 AFTER
                                                                ;A WRITE OF 401 WORDS.
005506  032701  000010                  BIT     #10,R1          ;SECTORS 10,11 WRITTEN?
005512  001005                          BNE     TST2            ;YES
005514  062701  000012                  ADD     #12,R1          ;RKDA TO BE USED NEXT (SEC 10)
005520  062702  000016                  ADD     #16,R2          ;EXPCTD RKDA AFTER WRITE IS DONE
005524  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.
                        ;;*****************************************************************
005526  000004         TST2:  SCOPE

005530  013701 001502          MOV    @DRV,R1          ;GET DRIVE #
005534  012737 005542 001110   MOV    #1$,$LPERR       ;ADDRESS FOR LUPING ON EROR
005542  104416         1$:    CON.RESET
005544  104420                DRV.RESET
005546  104416                CON.RESET

                                                        ;CLEAN UP THE DATA BUFFER
005550  004737 006434          JSR    PC,CLEANBUF      ;INTO WHICH READ WILL
                                                        ;BE DONE
005554  012703 001000          MOV    #1000,R3         ;WORD COUNT

005560  004737 006424          JSR    PC,DOREAD        ;GO DO A READ OF 2 SECTORS
                                                        ;FROM DISK ADDRESS GIVEN IN R1
005564  104101                ERROR  101              ;INTERRUPT DID NOT OCCUR AFTER
                                                        ;READ OF 401 WORDS WAS DONE.
005566  004737 020112          JSR    PC,CHKCS         ;CHECK IF EROR BIT IN RKCS SET?
005572  104102                ERROR  102              ;EROR BIT IN RKCS SET ON DOING A
                                                        ;READ OF 401 WORDS.
005574  012704 032514          MOV    #DBUF,R4         ;STARTING BUS ADDRESS, INTO WHICH READ
005600  010402                MOV    R4,R2            ;WAS DONE
005602  004737 020150          JSR    PC,CHKBA         ;CHECK IF RKBA INCREMENTED RIGHT
005606  104104                ERROR  104              ;RKBA DID NOT INCREMENT RIGHT AFTER READ
                                                        ;OF 401 WORDS.
005610  012705 177764          MOV    #-14,R5          ;ALLOW 12 ERRORS, AT THE MOST
005614  022712 111111  2$:     CMP    #111111,(R2)     ;CORRECT DATA READ?
005620  001410                BEQ    3$               ;YES
005622  012737 111111 001164   MOV    #111111,$REG1    ;GET EXPCTD DATA WORD
005630  004737 005720          JSR    PC,ERINF1        ;GET ERROR INFORMATION
005634  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

005636  005205                INC    R5               ;REPORT 12 ERORS AT MOST
005640  001421                BEQ    6$
005642  005722          3$:   TST    (R2)+            ;INCREMENT POINTER
005644  020227 033516          CMP    R2,#DBUF+1002    ;CHECKED ALL 401 WORDS?
005650  001361                BNE    2$

005652  005712          4$:   TST    (R2)             ;CHECK THAT REST OF 377 WORDS
005654  001407                BEQ    5$               ;ARE ALL 0'S
005656  005037 001164          CLR    $REG1            ;GET EXPCTD DATA WORD (0)
005662  004737 005720          JSR    PC,ERINF1        ;GET ERROR INFO
005666  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)

005670  005205                        INC     R5              ;REPORT 12 ERORS AT MOST
005672  001404                        BEQ     6$
005674  005722                5$:     TST     (R2)+           ;ALL WORDS CHECKED?
005676  020227  034514                CMP     R2,#DBUF+2000
005702  001363                        BNE     4$              ;IF NOT GO BAK

005704  032701  000010        6$:     BIT     #10,R1          ;WERE SECTORS 10,11 READ
005710  001030                        BNE     TST3            ;YES
005712  062701  000012                ADD     #12,R1          ;FROM NEW RKDA, SEC 10
005716  000711                        BR      1$              ;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.

005720  010237  001162        ERINF1: MOV     R2,$REG0        ;GET BUS ADDRESS OF DATA ERROR
005724  011237  001166                MOV     (R2),$REG2      ;GET BAD DATA WORD (READ)
005730  010146                        MOV     R1,-(SP)
005732  020227  033512                CMP     R2,#DBUF+776    ;FIGURE OUT THE DISK ADDRESS
005736  003001                        BGT     1$              ;WHERE DATA ERROR OCCURRED
005740  005316                        DEC     (SP)
005742  005216                1$:     INC     (SP)
005744  032716  000010                BIT     #10,(SP)
005750  001405                        BEQ     2$
005752  032716  000004                BIT     #4,(SP)
005756  001402                        BEQ     2$
005760  062716  000004                ADD     #4,(SP)
005764  012637  001170        2$:     MOV     (SP)+,$REG3
005770  000207                        RTS     PC
```

```
1997                                    ;;*****************************************************************
1998                                    ;*TEST 3        PERFORM WRITE OF 12 SECTORS + 1 WORD
1999                                            ;THIS TEST CHECKS FOR ANOTHER BOUNDARY CONDITION.  IT
2000                                            ;PERFORMS A WRITE OF 12 SECTORS + 1 WORD.  RKDA,RKBA,
2001                                            ;RKWC ARE CHECKED TO SEE IF THEY ARE INCREMENTED CORRECTLY
2002                                            ;VALIDITY OF THE DATA WRITTEN IS CHECKED IN THE NEXT
2003                                            ;TEST.  DATA IS WRITTEN ON SECTORS 0-11, SURFACE 0
2004                                            ;CYLINDER 0 (6001TH WORD ON SECTOR 0, SURFACE 1, CYL 0).
2005                                            ;ALSO ON SECTORS 0-11, SURFACE 1 (6001TH WORD ON SECTOR
2006                                            ;0, CYL 1)
2007                                    ;;*****************************************************************
2008  005772  000004                    TST3:   SCOPE
2009  005774  013701  001502                    MOV     @DRV,R1         ;GET DRIVE #
2010  006000  012737  006020  001110            MOV     #1$,$LPERR      ;LUP ON EROR TO '1$'
2011  006006  010102                            MOV     R1,R2
2012  006010  062702  000021                    ADD     #21,R2
2013  006014  012703  006001                    MOV     #6001,R3
2014  006020  104416                    1$:     CON.RESET
2015  006022  104420                            DRV.RESET
2016  006024  104416                            CON.RESET
2017
2018
2019  006026  012737  044444  032514            MOV     #44444,DBUF     ;PATTERN TO BE WRITTEN
2020
2021  006034  004737  006304                    JSR     PC,DOWRITE      ;GO DO WRITE
2022
2023  006040  104101                            ERROR   101             ;INTERUPT DID NOT OCCUR ON
2024                                                                    ;COMPLETION OF WRITE
2025  006042  004737  020112                    JSR     PC,CHKCS        ;CHECK IF EROR BIT IN RKCS SET
2026  006046  104102                            ERROR   102             ;EROR BIT IN RKCS SET ON DOING WRITE
2027  006050  004737  020126                    JSR     PC,CHKDA        ;CHECK IF RKDA INCREMENTED RIGHT
2028  006054  104103                            ERROR   103             ;RKDA DID NOT INCREMENT CORRECTLY
2029                                                                    ;AFTER A WRITE OF 6001 (OCTAL) WORDS.
2030                                                                    ;(12 SECTORS + 1)
2031  006056  004737  020230                    JSR     PC,CHKWC        ;CHECK IF RKWC OVERFLOWED TO 0
2032  006062  104105                            ERROR   105             ;RKWC DID NOT OVERFLOW TO 0
2033  006064  032701  000020                    BIT     #20,R1          ;WRITTEN ON SURFACE 1?
2034  006070  001006                            BNE     TST4            ;YES
2035  006072  010201                            MOV     R2,R1
2036  006074  062702  000020                    ADD     #20,R2          ;SURFACE 1
2037  006100  012703  005401                    MOV     #5401,R3        ;WORD COUNT
2038  006104  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.
                                ;;*****************************************************************
006106  000004                  TST4:   SCOPE
006110  013701  001502                  MOV     @DRV,R1         ;GET DRIVE #
006114  062701  000013                  ADD     #13,R1          ;DISK ADDRESS FROM WHERE READ IS DONE
006120  012737  006126  001110          MOV     #1$,$LPERR
006126  104416                  1$:     CON.RESET
006130  104420                          DRV.RESET
006132  104416                          CON.RESET

006134  004737  006434                  JSR     PC,CLEANBUF     ;CLEAN UP THE BUFFER INTO WHICH
                                                                ;READ WILL BE DONE
                                                                ;SET UP RKDA
                                                                ;SECTOR 11, SURFACE 0
006140  012703  001000                  MOV     #1000,R3        ;WORD COUNT

006144  004737  006424                  JSR     PC,DOREAD       ;GO READ 1000 WORDS (2 SECS)
006150  104101                          ERROR   101             ;INTERRUPT DID NOT OCCUR AFTER
                                                                ;COMPLETION OF READ
006152  004737  020112                  JSR     PC,CHKCS        ;CHECK IF EROR BIT IN RKCS SE~
006156  104102                          ERROR   102             ;EROR (RKCS) SET ON DOING READ
006160  012704  032514                  MOV     #DBUF,R4        ;STARTING BA OF DATA BUFER
006164  010402                          MOV     R4,R2
006166  004737  020150                  JSR     PC,CHKBA        ;RKBA INCREMENTED CORRECTLY?
006172  104104                          ERROR   104             ;RKBA DID NOT INCREMENT CORRECTLY
006174  012705  177764                  MOV     #-14,R5
006200  022712  044444          2$:     CMP     #44444,(R2)     ;DATA WORD OK?
006204  001410                          BEQ     3$
006206  012737  044444  001164          MOV     #44444,$REG1    ;NO, GET EXPCTD DATA WORD
006214  004737  005720                  JSR     PC,ERINF1       ;GET, OTHER ERROR INFO
006220  104100                          ERROR   100             ;DATA ERROR.  A WRITE OF 6001
                                                                ;WORDS (12 SECS + 1 WORD) WAS DONE
                                                                ;IN A PREVIOUS TEST.  THE LAST TWO
                                                                ;SECTORS (LAST 401 WORDS) WERE READ
                                                                ;BACK.  THIS ERROR INDICATES THAT
                                                                ;SEC #11 (LAST BUT ONE SECTOR) GAVE
                                                                ;BAD DATA WORDS
006222  005205                          INC     R5              ;REPORT 12 ERORS AT MOST
006224  001421                          BEQ     6$
006226  005722                  3$:     TST     (R2)+           ;INCREMENT POINTER TO BA
006230  020227  033516                  CMP     R2,#DBUF+1002   ;CHECKED 401 WORDS?
006234  001361                          BNE     2$
006236  005712                  4$:     TST     (R2)            ;CHECK THAT THE REMAINING 377
006240  001407                          BEQ     5$              ;WORDS OF THE LAST SECTOR (SEC #0)
006242  005037  001164                  CLR     $REG1           ;WERE READ BACK AS 0'S
006246  004737  005720                  JSR     PC,ERINF1
006252  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
```

```
2095                                                                    ;AS NON-ZERO, THE REST 377 SHOULD BE
2096                                                                    ;ALL 0'S.  THIS ERROR INDICATES THAT
2097                                                                    ;THE SOME OF 377 WORDS
2098                                                                    ;WERE NOT CORRECT
2099  006254  005205                       INC     R5                  ;REPORT 12 ERORS AT MOST
2100  006256  001404                       BEQ     6$
2101  006260  005722                 5$:   TST     (R2)+               ;INCREMENT POINTER
2102  006262  020227  034514              CMP     R2,#DBUF+2000       ;CHECKED ALL WORDS?
2103  006266  001363                       BNE     4$                  ;NO
2104
2105  006270  032701  000020         6$:   BIT     #20,R1              ;DONE CHECKING FOR SURFACE 1?
2106  006274  001070                       BNE     TST5                ;YES
2107  006276  062701  000020               ADD     #20,R1              ;SO SET UP FOR SURFACE 1
2108  006302  000711                       BR      1$                  ;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.

006304  012777  004002  172710  DOWRITE: MOV     #4002,@RKCS     ;WRITE, IBA
006312  010177  172712          DOXFER: MOV     R1,@RKDA        ;ADDRESS THE DRIVE
006316  010377  172702                  MOV     R3,@RKWC        ;XFER THIS # OF WORDS
006322  005477  172676                  NEG     @RKWC
006326  012777  032514  172672          MOV     #DBUF,@RKBA     ;USE THIS BUS ADDRESS
006334  012777  006414  172676          MOV     #3$,@RKVEC      ;SET UP INTERRUPT VECTOR
006342  005046                          CLR     -(SP)           ;NEW PSW
006344  012746  006352                  MOV     #1$,-(SP)       ;SET NEW PC TO STACK **********
006350  000002                          RTI
006352  052777  000101  172642  1$:     BIS     #101,@RKCS      ;SET IDE, GO (WRITE,IBA/ READ
006360  005037  001466                  CLR     CICNT
006364  012737  177760  001470          MOV     #-20,CICNT1
006372  005237  001466          2$:     INC     CICNT           ;WAIT  FOR INTERRUPT
006376  001375                          BNE     .-4

006400  005237  001470                  INC     CICNT1
006404  001372                          BNE     2$

006406  004737  022032                  JSR     PC,GT4RG        ;TIMED OUT, INTERRUPT DID NOT OCCUR
006412  000207                          RTS     PC              ;RETURN TO THE EROR MEASGE

006414  022626          3$:     CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
006416  062716  000002          ADD     #2,(SP)         ;ADJUST RETURN ADDRESS TO SKIP OVER
006422  000207                  RTS     PC              ;EROR MESAGE ON RETURN
```

```
2143                                    ;THIS ROUTINE PERFORMS A READ ON THE DISK AT THE TIME OF ENTRY R1 CONTAINS
2144                                    ;THE DISK ADDRESS FROM WHERE THE READ IS TO BE DONE. R3 CONTAINS THE WORD
2145                                    ;COUNT (RKWC). READ WILL BE DONE INTO DATA BUFFER AT 'DBUF'.
2146
2147                                    ;READ IS DONE IN INTERRUPT MODE. IF THE INTERRUPT DOES NOT OCCUR WITHIN
2148                                    ;A CERTAIN TIME, RETURN IS MADE TO THE ERROR MESSAGE FOLLOWING THE 'JSR'
2149                                    ;CALL. IF THE INTERRUPT OCCURS AS EXPECTED, RETURN ADDRESS IS ADJUSTED
2150                                    ;TO SKIP OVER THE ERROR MESSAGE.
2151
2152  006424  012777  000004  172570  DOREAD: MOV     #4,@RKCS        ;READ
2153  006432  000727                          BR      DOXFER
2154
2155
2156                                    ;CLEANBUF
2157                                    ;CLEANS OUT THE DATA BUFFER (ALL WORDS WRITTEN TO 177777) INTO WHICH THE
2158                                    ;READ FROM THE DISK WILL BE DONE.  DATA BUFFER STARTS AT 'DBUF'AND IS
2159                                    ;1000 (OCTAL) WORDS LONG.
2160
2161  006434  012702  177000           CLEANBUF:MOV    #-1000,R2       ;SET COUNT
2162  006440  012705  032514                   MOV     #DBUF,R5        ;INITIALIZE BA
2163  006444  012725  022222           1$:     MOV     #22222,(R5)+
2164  006450  005202                          INC     R2              ;DONE ALL WORDS?
2165                                                                    ;BUFFER STARTING AT (PHYSICAL, BUS
2166                                                                    ;ADDRESS 177000 (177000-200776.
2167  006452  001374                          BNE     1$
2168  006454  000207                          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 0,1, CYL 0, SURFACE
                                ;*0 ARE WRITTEN.  PHYSICAL BUS ADDRESSES FOR THE DATA BUFFER:
                                ;* 177000 TO 200776 I.E. (32K-256) TO (32K+255)

                                ;*CHECKING IS DONE TO SEE IF MEX BITS, RKBA,RKDA,RKWC INCREMENTED
                                ;*CORRECTLY. THEN DATA BUFFER IS CLEARED OUT AND A READ IS DONE
                                ;*INTO IT.  A CHECK IS MADE TO SEE IF THE CORRECT DATA WAS RECIEVED.
                                ;*ONLY 12 DATA ERRORS ARE REPORTED.
                                ;;*****************************************************************
006456  000004                T5TS:   SCOPE
006460  023727  017772  002000        CMP     $LSTBK,#2000    ;33K OR MORE OF MEMORY?
006466  103002                        BHIS    1$              ;YES
006470  000137  007040                JMP     T5T6            ;IF NOT, DONT DO THIS TEST
006474  012737  001600  172352  1$:   MOV     #1600,@#KIPAR5  ;MAP 28-32K THRU PAR 5
006502  012737  002000  172354        MOV     #2000,@#KIPAR6  ;MAP 32-36K THRU PAR 6
006510  012737  000001  177572        MOV     #1,@#SR0        ;TURN ON MEMORY MANAGEMENT

                                ;SET UP DATA BUFFER (1000 OCTAL WORDS LONG) FOR WRITING TWO SECTORS
                                ;ON THE DISK.  THE TRANSFER IS DONE AROUND THE 32K BOUNDARY FROM
                                ;BUS ADDRESS (PHYSICAL) 177000 TO 200776, (32K-256) TO (32+256)

                                ;DATA IN THE BUFFER IS A COUNT PATTERN STARTING FROM 1 FOR THE FIRST
                                ;WORD TO (000 FOR THE LAST WORD.
                                ;PHYSICAL BUFFER ADDRESS:  177000 TO  200776 (32K-256 TO 32K+256)
006516  012737  006524  001110        MOV     #2$,$LPERR      ;LUP TO 2$ ON EROR (SW 9)
006524  104416                2$:     CON.RESET
006526  104420                        DRV.RESET

006530  012700  000001                MOV     #1,R0           ;INITIALIZE DATA PATTERN TO BE
                                                              ;WRITTEN
006534  012701  137000                MOV     #137000,R1      ;BA TO START PHYSICAL ADDRESS=177000
006540  010021                3$:     MOV     R0,(R1)+        ;WRITE COUNT PATTERN (1-1000)
006542  005200                        INC     R0              ;INTO DATA BUFFER (PHYS ADDRES
006544  020027  001001                CMP     R0,#1001        ;177000 TO 200776)
006550  001373                        BNE     3$
006552  013777  001502  172450        MOV     @DRV,@RKDA      ;SUR 0, SEC 0, CYL 0
006560  012777  177000  172436        MOV     #-1000,@RKWC    ;WORD COUNT =2 SECS
006566  012777  177000  172432        MOV     #177000,@RKBA   ;BUS ADDRESS.
006574  012777  000003  172420        MOV     #3,@RKCS        ;WRITE,GO
006602  104417                        CON.RDY                 ;WAIT FOR CNTROL RDY

006604  004737  020112                JSR     PC,CHKCS        ;ANY EROR IN RKCS?
006610  104102                        ERROR   102             ;'ERR' SET IN RKCS, ON DOING A
                                                              ;WRITE OF SECTORS (0,1) FROM
                                                              ;(PHYSICAL) BUS ADDRESSS 177000
                                                              ;(177000 TO 200776)

006612  004737  020204                JSR     PC,CHKMEX
                                                              ;CHECK THAT RKBA OVERFLOWED INTO
                                                              ;EXTENDED MEM. BIT (0) OF RKCS (BIT 4)
                                                              ;EX MEM BIT 0 SET?
                                                              ;GET RKCS,ER,DS,DA
006616  104106                        ERROR   106             ;MEX BITS INCORRECT. BIT 4 OF RKCS
                                                              ;(MEX BIT 0) SHOULD HAVE SET
```

```
2225                                                    ;AFTER RKBA OVERFLOWED ON DOING
2226                                                    ;A TRANSFER OF 2 SECTORS FROM BUS
2227                                                    ;ADDRESS (PHYSICAL) STARTING AT 177000
2228  006620  012703  001000          MOV     #1000,R3  ;WORD COUNT
2229  006624  012704  177000          MOV     #177000,R4  ;STARTING BUS ADDRESS
2230  006630  004737  020150          JSR     PC,CHKBA  ;CHECK IF RKBA INCREMENTED CORRECTLY
2231  006634  104104                  ERROR   104       ;;RKBA DID NOT INCREMENT CORRECTLY
2232                                                    ;AFTER WRITE IF 2 SECTORS FROM A DATA
2233                                                    ;BUFFER STARTING AT (PHYSICAL) BUS
2234                                                    ;ADDRESS (177000-200776)
2235  006636  013702  001502          MOV     @DRV,R2   ;GET EXPECTED
2236  006642  062702  000002          ADD     #2,R2     ;DISK ADDRESS
2237  006646  004737  020126          JSR     PC,CHKDA  ;CHECK IF RKDA INCREMENTED CORRECTLY
2238  006652  104103                  ERROR   103       ;RKDA INCREMENTED WRONGLY AFTER
2239                                                    ;A WROTE OF 2 SECTOR (0,1) FROM
2240                                                    ;DATA BUFFER STARTING AT BUS
2241                                                    ;ADDRESS (PHYSICAL) 177000.
2242
2243  006654  004737  020230          JSR     PC,CHKWC  ;CHECK IF RKWC OVERFLOWED CORRECTLY
2244  006660  104105                  ERROR   105       ;RKWC DID NOT OVERFLOW TO 0 ON
2245                                                    ;DOING A WRITE OF 2 SECTORS FROM
2246                                                    ;BA = 177000
2247
2248                                                    ;NOW, READ IS DONE OF THE DATA
2249                                                    ;THAT WAS WRITTEN TO SEE IF IT
2250                                                    ;CAN BE READ CORRECTLY
2251
2252  006662  012737  006670  001110          MOV     #4$,$LPERR  ;LUP TO 4$ ON EROR (SW 9)
2253  006670  104416                  4$:     CON.RESET
2254  006672  104420                  DRV.RESET
2255
2256  006674  012701  137000          MOV     #137000,R1  ;CLEAR THE 1000-WORD DATA
2257  006700  005021                  5$:     CLR     (R1)+     ;BUFFER (PA 177000 TO 200776)
2258  006702  020127  141000          CMP     R1,#141000  ;ALL DONE?
2259  006706  001374                  BNE     5$
2260
2261                                                    ;NOW, READ BACK INTO THE
2262                                                    ;SAME BUFFER 2 SECTORS
2263                                                    ;WRITTEN PREVIOUSLY
2264
2265  006710  013777  001502  172312  MOV     @DRV,@RKDA  ;ADDRESS THE DRIVE CYL 0, SEC 0, 0
2266  006716  012777  177000  172300  MOV     #-1000,@RKWC  ;READ 2 SECTORS
2267  006724  012777  177000  172274  MOV     #177000,@RKBA  ;INTO THIS BUS ADDRESS
2268  006732  012777  000005  172262  MOV     #5,@RKCS  ;READ, GO
2269
2270  006740  104417                  CON.RDY           ;WAIT FOR CNTROL RDY
2271
2272  006742  004737  020112          JSR     PC,CHKCS  ;ANY ERROR IN RKCS?
2273  006746  104102                  ERROR   102       ;ERROR BIT SET IN RKCS ON DOING
2274                                                    ;A READ OF 2 SECTORS (0,1), CYL 0,
2275                                                    ;SUR 0, INTO DATA BUFFER STARTING
2276                                                    ;AT BUS ADDRESS 1770000, NOTE AFTER
2277                                                    ;1777776, RKBA WILL OVERFLOW (0)
2278                                                    ;INTO MEX BITS (BIT 4) OF RKCS.
2279                                                    ;IF THE ENTIRE TRANSFER (1000 WORD)
2280                                                    ;WAS DONE RKBA WILL CONTAIN 1000
```

```
2281                                                                    ;AND BIT 4 OF RKCS (MEX) WILL BE SET.
2282
2283  006750  012705  177764    6$:     MOV     #-14,R5                ;REPORT ONLY 12 ERRORS.
2284  006754  012702  137000            MOV     #137000,R2             ;STARTING ADDRESS OF BUFFER
2285                                                                    ;(PA=177000)
2286  006760  012701  000001            MOV     #1,R1                  ;INITIALIZE DATA PATTERN
2287
2288  006764  020112            7$:     CMP     R1,(R2)                ;CORRECT DATA WORD RECVD?
2289  006766  001414                    BEQ     8$
2290  006770  005705                    TST     R5                     ;REPORT THIS ERROR?
2291  006772  001420                    BEQ     9$
2292  006774  005205                    INC     R5                     ;COUNT ERORS
2293
2294  006776  010137  001162            MOV     R1,$REG0               ;GET EXPCTED DATA WORD
2295
2296  007002  011237  001164            MOV     (R2),$REG1             ;GET DATA WORD RECVD
2297  007006  104107                    ERROR   107                    ;DATA COMPARISON ERROR ON DOING A
2298                                                                    ;READ OF 2 SECTORS (0,1, CYL 0, SURFACE 0
2299                                                                    ;INTO DATA BUFFER (PHYSICAL ADDRESS
2300                                                                    ;177000 TO 200776)
2301
2302  007010  104421  002214            TYPMSG  ,MSG13                 ;TYPE 'PHYSICAL BUS ADDRESS'
2303  007014  004737  017774            JSR     PC,TYPDBO              ;TYPE THE 6 DIGIT PHYSICAL BUS ADDRESS
2304                                                                    ;WHERE THE DATA ERROR OCCURRED.
2305
2306  007020  062702  000002    8$:     ADD     #2,R2                  ;INCREMENT POINTER TO BA
2307  007024  005201                    INC     R1
2308  007026  022701  001001            CMP     #1001,R1               ;CHECKED THE ENTIRE BUFFER?
2309  007032  001354                    BNE     7$
2310
2311                                                                    ;**OFF
2312  007034  005037  177572    9$:     CLR     @#SR0                  ;TURN OFF MEMORY MANAGEMENT
```

```
2313                                    ;;*****************************************************************
2314                                    ;*TEST 6        CHECK DATA TRANSFER FROM 28K TO 32K
2315                                    ;*THIS TEST DOES A WRITE OF 4K WORDS FROM A BUFFER LOCATED AT 28K
2316                                    ;*THEN THE DATA IS READ BACK INTO THE SAME BUFFER(28K-32K) AND IS
2317                                    ;*CHECKED TO SEE IF IT IS CORRECT.  NOTE THAT THE BUFFER IS FILLED
2318                                    ;*WITH ALL 1'S BEFORE DOING THE READ.
2319
2320                                    ;*THE WRITE IS DONE STARTING AT CYLINDER 0, SECTOR 0, SURFACE 0.
2321                                    ;;*****************************************************************
2322  007040  000004                   TST6:   SCOPE
2323
2324  007042  023727  017772  001740            CMP     $LSTBK,#1740    ;32K OR MORE OF MEMORY?
2325  007050  103002                           BHIS    1$              ;YES
2326  007052  000137  007374                   JMP     TST7            ;IF NOT, DONT DO THIS TEST
2327  007056  012737  001600  172352  1$:      MOV     #1600,@#KIPAR5  ;MAP 28-32K THRU PAR 5
2328  007064  012737  000001  177572          MOV     #1,@#SR0        ;TURN ON MEM MANAGEMENT
2329
2330                                    ;WRITE A COUNT PATTERN (1-10000) INTO DATA BUFFER (PHYSICAL ADDRESS
2331                                    ;160000-177776).  THIS DATA BUFFER WILL BE USED FOR WRITING ON THE DISK.
2332
2333  007072  012737  007100  001110          MOV     #2$,$LPERR      ;LUP TO 2$ ON EROR
2334  007100  104416            2$:     CON.RESET
2335  007102  104420                    DRV.RESET
2336  007104  012700  000001                  MOV     #1,R0           ;INITIALIZE DATA PATTERN TO BE WRITTEN
2337  007110  012701  120000                  MOV     #120000,R1      ;INITIALIZE BA (PA=160000) 28K
2338  007114  010021            3$:     MOV     R0,(R1)+        ;WRITE COUNT PATTERNS (1-10000)
2339  007116  005200                    INC     R0              ;INTO DATA BUFFER (PA 160000 TO
2340  007120  020027  010001                  CMP     R0,#10001       ;177776, 28K-32K)
2341  007124  001373                           BNE     3$
2342
2343  007126  013777  001502  172074          MOV     @DRV,@RKDA      ;WRITE ON SEC 0, CYL 0, SUR1
2344  007134  012777  170000  172062          MOV     #-10000,@RKWC   ;4K WORDS
2345  007142  012777  160000  172056          MOV     #160000,@RKBA   ;FROM THIS BUS ADDRESS
2346  007150  012777  000003  172044          MOV     #3,@RKCS        ;WRITE, GO
2347
2348  007156  104417                    CON.RDY                         ;WAIT FOR CONTROL READY
2349
2350  007160  004737  020112                  JSR     PC,CHKCS        ;ANY ERROR IN RKCS?
2351  007164  104102                    ERROR   102                     ;'ERR' BIT SET IN RKCS ON DOING
2352                                                                    ;A WRITE OF 4K WORDS FROM 160000
2353                                                                    ;(28K-32K).  DISK WRITE BEGAN ON
2354                                                                    ;SEC 0, CYL 0, SUR 0.  IF ALL 4K
2355                                                                    ;WORDS WERE WRITTEN RKBA SHOULD OVERFLOW
2356                                                                    ;AND CONTAIN 0.  BIT 4 OF RKCS
2357                                                                    ;(MEX BIT) SHOULD BE 1
2358
2359  007166  004737  020204                  JSR     PC,CHKMEX       ;CHECK IF RKBA OVERFLOWED AND MEX
2360                                                                    ;BITS (4) IN RKCS WAS SET.
2361  007172  104106                    ERROR   106                     ;;MEX BIT 4 NOT SET AFTER OVERLFOW OF
2362                                                                    ;RKBA.  WRITE OF 4K WORDS (28K-32K) WAS DONE.
2363
2364                                                                    ;RETURN HERE IF NO ERROR
2365  007174  012703  010000                  MOV     #10000,R3       ;WORD COUNT
2366  007200  012704  160000                  MOV     #160000,R4      ;STARTING BUS ADDRESS
2367  007204  004737  020150                  JSR     PC,CHKBA        ;CHECK IF RKBA INCREMENTED CORRECTLY
2368  007210  104104                    ERROR   104                     ;RKBA DID NOT OVERFLOW TO AFTER A WRITE IF 4K
```

```
2369                                                                    ;WORDS (160000 TO 177776)
2370
2371  007212  004737  020230                    JSR     PC,CHKWC        ;CHECK RKWC OVERFLOWED CORRECTLY
2372  007216  104105                             ERROR   105             ;RKWC DID NOT OVEFLOW TO 0
2373                                                                    ;AFTER A WRITE OF 4K WORD (160000
2374                                                                    ;TO 177776)
2375
2376
2377  007220  012737  007226  001110             MOV     #4$,$LPERR      ;RETURN ADDRESS FOR LUPING
2378  007226  104416                    4$:      CON.RESET
2379  007230  104420                             DRV.RESET
2380
2381  007232  012701  120000                    MOV     #120000,R1      ;CLEAR THE DATA BUFFER BEFORE
2382  007236  005021                    5$:      CLR     (R1)+           ;DOING A READ INTO IT
2383  007240  022701  140000                    CMP     #140000,R1
2384  007244  001374                             BNE     5$
2385
2386  007246  013777  001502  171754             MOV     @DRV,@RKDA      ;READ FROM SEC 0, SUR 0, CYL 0
2387  007254  012777  170000  171742             MOV     #-10000,@RKWC   ;4K WORDS
2388  007262  012777  160000  171736             MOV     #160000,@RKBA   ;INTO THIS BUS ADDRESS
2389
2390  007270  012777  000005  171724             MOV     #5,@RKCS        ;READ, GO
2391  007276  104417                             CON.RDY                 ;WAIT FOR CNTROL RDY
2392
2393  007300  004737  020112                    JSR     PC,CHKCS        ;ANY ERROR IN RKCS?
2394  007304  104102                             ERROR   102             ;ERROR BIT SET IN RKCS ON DOING
2395                                                                    ;A READ OF 4K WORDS INTO MEMORY
2396                                                                    ;(160000-177776)
2397  007306  012705  177764                    MOV     #-14,R5         ;REPORT 12 ERRORS AT MOST
2398  007312  012702  120000                    MOV     #120000,R2      ;STARTING ADDRESS OF BUFFER
2399                                                                    ;(PA=160000)
2400  007316  012701  000001                    MOV     #1,R1           ;INITIALIZE DATA PATTERN
2401
2402  007322  020112                    6$:      CMP     R1,(R2)         ;CORRECT DATA WORD?
2403  007324  001413                             BEQ     7$
2404  007326  010137  001162                    MOV     R1,$REG0        ;GET EXPCTD DATA WORD
2405  007332  011237  001164                    MOV     (R2),$REG1      ;GET DATA WORD RECVD
2406  007336  104107                             ERROR   107             ;DATA ERROR ON DOING A READ OF
2407                                                                    ;4K WORDS STARTING FROM SEC 0,
2408                                                                    ;CYL 0, SUR 0 INTO MEMORY (160000-
2409                                                                    ;177776)
2410  007340  104421  002214                    TYPMSG  ,MSG13          ;TYPE 'PHYSICAL BUS ADDRESS'
2411  007344  004737  017774                    JSR     PC,TYPOBO       ;TYPE OUT THE PHYSICAL BUS ADDRESS
2412                                                                    ;WHERE DATA
2413  007350  005205                             INC     R5              ;ERROR OCCURRED.  R2 CONTAINS
2414  007352  001406                             BEQ     8$              ;THE VIRTUAL ADDRESS
2415
2416  007354  062702  000002            7$:     ADD     #2,R2           ;POINTER TO NEXT BA
2417  007360  005201                             INC     R1              ;NEXT PATTERN
2418  007362  022701  001000                    CMP     #1000,R1        ;ALL DONE?
2419  007366  001355                             BNE     6$              ;NO
2420
2421  007370  005037  177572            8$:     CLR     @#SR0           ;TURN OFF MEM MANAGEMENT
```

```
2422                                    ;;*****************************************************************
2423                                    ;*TEST 7        PERFORM THE LARGEST POSSIBLE DATA TRANSFER
2424                                    ;THIS TEST PERFORMS THE LARGEST DATA TRANSFER POSSIBLE
2425                                    ;WITHIN AVAILABLE MEMORY.
2426
2427                                    ;1) IF KT11 IS PRESENT A WRITE OF 16K WORDS IS DONE ON
2428                                    ;THE DISK (PROVIDED 29K OR MORE IS PRESENT).
2429
2430                                    ;2) IF KT11 IS NOT PRESENT THEN ALL BUT TOP 1.5K OF MEMORY WILL BE USED
2431                                    ;FOR WRITING (RKDP MONITOR).
2432
2433                                    ;ALL DATA TRANSFERS ARE DONE STARTING AT BA 'PGEND'.
2434                                    ;ALL WRITES ARE DONE BEGINNING AT SECTOR 0, CYLINDER 0.
2435                                    ;SURFACE 0.
2436
2437                                    ;AFTER DOING THE 'WRITE' A 'WRITE CHECK' IS DONE
2438                                    ;TO SEE IF THE WRITE WAS DONE CORRECTLY.  ANY WRITE
2439                                    ;CHECK ERROR IS REPORTED.
2440                                    ;;*****************************************************************
2441  007374  000004                T7$:    SCOPE
2442  007376  005737  017524                  TST     @#KT11          ;MEMORY MANAGEMENT PRESENT?
2443  007402  100004                          BPL     1$              ;NO
2444                                                                  ;YES
2445  007404  023727  017772  001540          CMP     @#LSTBK,#1540   ;28K OR MORE?
2446  007412  002034                          BGE     XFR16K          ;YES
2447
2448  007414  013703  002054          1$:     MOV     MAXBA,R3        ;FIGURE OUT THE MAXIMUM
2449  007420  162703  032514                  SUB     #PGEND,R3       ;XFER THAT CAN BE DONE
2450  007424  000241                          CLC
2451  007426  006003                          ROR     R3              ;FROM 'PGEND' TO THE
2452                                                                  ;'MAXBA'
2453  007430  005001                          CLR     R1
2454  007432  010346                          MOV     R3,-(SP)        ;PUT DIVIDEND ON STACK (LO)
2455  007434  005046                          CLR     -(SP)           ;(HIGH PART)
2456  007436  012746  000400                  MOV     #400,-(SP)      ;DIVISOR
2457  007442  004737  025212                  JSR     PC,@#$DIV       ;GO TO DIVIDE ROUTINE
2458  007446  005726                          TST     (SP)+           ;POP OFF REMAINDER FROM STACK
2459  007450  001401                          BEQ     2$
2460  007452  005201                          INC     R1              ;GET # OF SECTORS REQUIRED TO
2461                                                                  ;DO WRITE OF # OF WORDS
2462  007454  062601                  2$:     ADD     (SP)+,R1        ;(CONTAINED IN R0). R1 CONTAINS
2463                                                                  ;# OF SECTORS
2464  007456  005002                          CLR     R2              ;FORM THE EXPECTED DISK
2465  007460  162701  000014          3$:     SUB     #14,R1          ;ADDRESS - AFTER THE WRITE
2466  007464  100403                          BMI     4$              ;IS DONE.
2467  007466  062702  000020                  ADD     #20,R2
2468  007472  000772                          BR      3$
2469  007474  062701  000014          4$:     ADD     #14,R1          ;R2 CONTAINS EXPCTD RKDA
2470  007500  050102                          BIS     R1,R2           ;AFTER WRITE IS DONE
2471  007502  000404                          BR      XFR
2472
2473  007504  012703  040000          XFR16K: MOV     #40000,R3       ;# OF WORDS = 16K
2474  007510  012702  000124                  MOV     #124,R2         ;RKDA SHOULD INCREMENT TO
2475                                                                  ;THIS AFTER A TRANSFER OF 16K
2476                                                                  ;WORDS STARTING AT DISK
2477                                                                  ;ADDRESS = 0 (CYL 2,SUR 1,SEC 7
```

```
2478
2479  007514  053702  001502          XFR:    BIS     @DRV,R2         ;ADD THE DRIVE # TO
2480                                                                  ;EXPCTD RKDA AFTER XFER
2481  007520  012737  007526  001110          MOV     #1$,$LPERR      ;LUP BAK TO '1$' ON ERROR
2482                                                                  ;(SW 9)
2483  007526  104416                  1$:     CON.RESET
2484  007530  104420                          DRV.RESET
2485  007532  013777  001502  171470          MOV     @DRV,@RKDA      ;ADDRESS THE DRIVE, CYL 0,SUR 0,
2486                                                                  ;SEC 0
2487  007540  010377  171460                  MOV     R3,@RKWC
2488  007544  005477  171454                  NEG     @RKWC           ;WORD COUNT (IF MORE THAN
2489                                                                  ;28K IS AVAILABLE A TRANSFER
2490                                                                  ;OF 20K WILL BE DONE, IF
2491                                                                  ;LESS THAN 28K, THE
2492                                                                  ;LARGEST TRANSFER WITHIN THE
2493                                                                  ;AVAILABLE MEMORY WILL
2494                                                                  ;BE DONE. IN BOTH
2495                                                                  ;CASES, DATA TRANSFER
2496                                                                  ;WILL BE DONE STARTING
2497                                                                  ;AT 'PGEND'
2498  007550  012777  032514  171450          MOV     #PGEND,@RKBA    ;START WRITE FROM HERE
2499  007556  012777  000003  171436          MOV     #3,@RKCS        ;WRITE, GO
2500
2501  007564  104417                          CON.RDY                 ;WAIT FOR CONTROL READY
2502
2503  007566  004737  020112                  JSR     PC,CHKCS        ;CHCK RKCS FOR ERROR?
2504  007572  104102                          ERROR   102             ;ERROR BIT SET IN RKCS ON
2505                                                                  ;DOING A LARGE 'WRITE'
2506
2507  007574  004737  020126                  JSR     PC,CHKDA        ;CHECK IF RKDA INCREMENTED CORRECTLY?
2508  007600  104103                          ERROR   103             ;RKDA DID NOT INCREMENT
2509                                                                  ;CORRECTLY
2510
2511  007602  012704  032514                  MOV     #PGEND,R4
2512  007606  004737  020150                  JSR     PC,CHKBA        ;CHECK THAT RKBA INCREMENTED
2513                                                                  ;CORRECTLY?
2514  007612  104104                          ERROR   104             ;RKBA DID NOT INCREMENT
2515                                                                  ;CORRECTLY
2516
2517  007614  004737  020230                  JSR     PC,CHKWC        ;CHECK THAT RKWC OVERFLOWED
2518  007620  104105                          ERROR   105             ;RKWC DID NOT OVERFLOW TO
2519                                                                  ;ZERO AFTER A WRITE
2520
2521  007622  012737  007630  001110          MOV     #2$,$LPERR      ;LUP BAK TO '2$' ON EROR (SW 9)
2522  007630  104416                  2$:     CON.RESET
2523  007632  104420                          DRV.RESET
2524  007634  013777  001502  171366  3$:     MOV     @DRV,@RKDA      ;ADDRESS THE DRIVE, CYL 0,SEC 0,SUR 0
2525  007642  010377  171356                  MOV     R3,@RKWC
2526  007646  005477  171352                  NEG     @RKWC
2527  007652  012777  032514  171346          MOV     #PGEND,@RKBA    ;STARTING BA
2528
2529  007660  012777  000407  171334  4$:     MOV     #407,@RKCS      ;WRITE CHECK, SSE, GO
2530
2531  007666  104417                          CON.RDY                 ;WAIT FOR CONTROL RDY
2532
2533  007670  004737  020112                  JSR     PC,CHKCS        ;ANY ERROR IN RKCS
```

```
2534  007674  104102                          ERROR   102
2535
2536  007676  032777  040000  171316          BIT     #BIT14,@RKCS    ;SKIP CHECKING FOR WCE IF HE SET
2537  007704  001030                          BNE     7$
2538  007706  032777  000001  171304  5$:     BIT     #WCE,@RKER      ;WRITE CHECK ERROR?
2539  007714  001406                          BEQ     6$              ;NO
2540
2541  007716  004737  022032                  JSR     PC,GT4RG        ;GET RKCS,ER,DS,DA
2542  007722  017737  171300  001166          MOV     @RKBA,$REG2
2543  007730  104110                          ERROR   110             ;WRITE CHECK ERROR. RKDA GIVES THE DISK ADDRESS
2544                                                                  ;WHERE WCE OCCURRED. (NOTE THE SECTOR IN
2545                                                                  ;ERROR IS OBTAINED BY BACKING OFF 1 SECTOR).
2546                                                                  ;NOTE THAT THE DATA BUFFER WHICH WAS WRITE
2547                                                                  ;CHECKED IS NOT ON A SECTOR BOUNDARY
2548                                                                  ;(LIKE 256, 512 WORD ETC.), BUT IS SOME
2549                                                                  ;SECTORS & A FRACTION (LIKE 300 WORDS ETC.)
2550
2551  007732  005777  171266          6$:     TST     @RKWC           ;WAS THE ENTIRE DATA BUFFER
2552  007736  001413                          BEQ     7$              ;WRITE CHECKED?
2553
2554  007740  017705  171260                  MOV     @RKWC,R5
2555  007744  042705  177760                  BIC     #177760,R5      ;FORM THE RKBA AND RKWC
2556  007750  006305                          ASL     R5
2557  007752  160577  171250                  SUB     R5,@RKBA        ;TO BE USED FOR DOING
2558                                                                  ;WRITE-CHECK IF THE REST
2559  007756  042777  000017  171240          BIC     #17,@RKWC       ;OF THE DATA BUFFER
2560
2561  007764  000735                          BR      4$              ;GO DO WRT-CHK FOR
2562                                                                  ;REST OF THE BUFFER
2563
2564                                  ;GO CHECK THE REST OF THE DRIVES.
2565
2566  007766  000137  005366          7$:     JMP     NXTDRV
```

```
                                        .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.


007772  104416                  EXRCSR: CON.RESET
007774  012700  001306                  MOV     #KEY,R0         ;CLEAR UP THE LOCATIONS FROM
010000  005020                  1$:     CLR     (R0)+           ;'KEY' TO 'KWHR' (EXCLUDED)
010002  020027  001552                  CMP     R0,#KWHR
010006  001374                          BNE     1$
010010  105737  001253                  TSTB    FRSTRT          ;RESTARTED AT 210?
010014  001045                          BNE     3$              ;YES, SAVE THE STATISTICS COLLECTED
                                                                ;UPTILL NOW, DONT CLEAR THEM

010016  005020                  2$:     CLR     (R0)+           ;CLEAR LOCATIONS FROM 'HECN'
010020  020027  002032                  CMP     R0,#PCMND       ;TO 'PCMND' (EXCLUDED)
010024  001374                          BNE     2$

010026  012700  001554                  MOV     #KWMIN,R0       ;INITIALIZE COUNTS FOR MINS,
010032  012701  177704                  MOV     #-60.,R1        ;SECS, KW11
010036  010120                          MOV     R1,(R0)+
010040  010120                          MOV     R1,(R0)+
010042  010120                          MOV     R1,(R0)+

010044  012700  025730                  MOV     #RSDRVL,R0      ;INITIALIZE PTR TO RANDOM NO. SEEDS
010050  012720  123456                  MOV     #123456,(R0)+   ;USED BY THE RANDOM NUMBER GENERATOR
010054  012720  176543                  MOV     #176543,(R0)+   ;SET UP RANDOM NUMBER SEEDS TO BE
010060  012720  001201                  MOV     #1201,(R0)+
010064  012720  062465                  MOV     #62465,(R0)+
010070  012720  176105                  MOV     #176105,(R0)+
010074  012720  174532                  MOV     #174532,(R0)+
010100  012720  157650                  MOV     #157650,(R0)+
010104  012720  030753                  MOV     #30753,(R0)+
010110  012720  131547                  MOV     #131547,(R0)+
010114  012720  032070                  MOV     #32070,(R0)+
010120  012720  123456                  MOV     #123456,(R0)+
010124  012720  176543                  MOV     #176543,(R0)+


010130  005227  177777          3$:     INC     #-1             ;FIRST PASS?
010134  001004                          BNE     4$              ;BRANCH IF NOT
010136  012737  000062  002056          MOV     #50.,REPCNT     ;SET UP SHORT REPITITION COUNT
                                                                ;FOR FIRST PASS.
010144  000403                          BR      5$
010146  013737  001236  002056  4$:     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.


010154  004737  022656          5$:     JSR     PC,CHDPRS       ;CHECK IF ANY DRIVES ARE PRESENT
                                                                ;IF NONE, GO TO $EOP, END OF PASS
```

```
2623
2624  010160  012746  177777                MOV   #177777,-(SP)   ;PUT LOW DIVIDEND ON STACK
2625  010164  005046                        CLR   -(SP)            ;CLEAR HIGH DIVIDEND AND PUSH
2626                                                               ;IT ON STACK
2627  010166  013746  001264                MOV   DRVPRS,-(SP)     ;PUSH DIVISOR ON STACK
2628  010172  004737  025212                JSR   PC,@#$DIV        ;GO TO THE 'DIVIDE' SUBROUTINE
2629  010176  005726                        TST   (SP)+            ;DISCARD THE REMAINDER. QUOTIENT IS
2630                                                               ;NOW ON TOP OF THE STACK
2631  010200  012637  001520                MOV   (SP)+,DRMAP      ;GET MAPPING FACTOR FOR DRIVES
2632  010204  012737  000504  001522        MOV   #504,CYLMAP      ;GET MAPPING FACTOR FOR CYLINDERS
2633  010212  012737  012527  001524        MOV   #5463.,SECMAP    ;GET MAPPING FACTOR FOR SECTORS
2634  010220  012737  031463  001526        MOV   #13107.,FNMAP    ;GET MAPPING FACTOR FOR FUNCTION
2635
2636  010226  013700  002054                MOV   MAXBA,R0         ;COMPUTE THE MAXIMUM ALLOWABLE
2637  010232  163700  002052                SUB   BASEBA,R0        ;WORD COUNT FOR DATA TRANSFERS
2638  010236  000241                        CLC
2639  010240  006000                        ROR   R0               ;CONVERT TO WORDS
2640
2641  010242  012746  177777                MOV   #177777,-(SP)    ;PUT LOW DIVIDEND ON STACK
2642  010246  005046                        CLR   -(SP)            ;CLEAR HIGH DIVIDEND AND PUSH
2643                                                               ;IT ON STACK
2644  010250  010046                        MOV   R0,-(SP)         ;PUSH DIVISOR ON STACK
2645  010252  004737  025212                JSR   PC,@#$DIV        ;GO TO THE 'DIVIDE' SUBROUTINE
2646  010256  005726                        TST   (SP)+            ;DISCARD THE REMAINDER. QUOTIENT IS
2647                                                               ;NOW ON TOP OF THE STACK
2648  010260  005216                        INC   (SP)
2649  010262  012637  001530                MOV   (SP)+,BAMAP      ;SAVE THE MAPPING FACTOR FOR BUS ADDRESS
```

```
                                        ;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.

010266  105737  001253          WRDSK:  TSTB    FRSTRT          ;RESTARTED AT 210?
010272  001407                          BEQ     1$              ;NO
010274  105037  001253                  CLRB    FRSTRT          ;YES, CLEAR THE RESTART FLAG
010300  032777  000020  170632          BIT     #SW4,@SWR       ;SW 4 SET?
010306  001401                          BEQ     1$              ;NO
010310  000540                          BR      BEGEX1          ;YES, DONT REWRITE ALL DISKS WITH
                                                                ;RANDOM PATTERNS
010312  012737  010330  001110  1$:     MOV     #2$,$LPERR      ;LUP TO '2$' ON EROR, SW 9 SET!
010320  012702  001254                  MOV     #PDR,R2         ;POINTER TO DRIVE #'S TABLE
010324  013703  001264                  MOV     DRVPRS,R3       ;# OF DRIVES PRESENT
010330  012700  011410          2$:     MOV     #4872.,R0       ;NUMBER OF SECTORS PER DISK
010334  112201                          MOVB    (R2)+,R1        ;GET THE FIRST AVAILABLE DRIVE
010336  042701  177770                  BIC     #177770,R1      ;POSITION THE BITS (15,14,13)
010342  010137  001502                  MOV     R1,@DRV
010346  000241                          CLC
010350  006001                          ROR     R1
010352  006001                          ROR     R1
010354  006001                          ROR     R1
010356  006001                          ROR     R1
010360  010177  170644                  MOV     R1,@RKDA        ;BASE DISK ADDRESS

010364  013704  002054                  MOV     MAXBA,R4        ;CALCULATE MAXIMUM BUFFER
010370  163704  002052                  SUB     BASEBA,R4
010374  006204                          ASR     R4              ;CONVERT TO WORDS
010376  042704  000377                  BIC     #377,R4         ;KEEP ONLY WHOLE BLOCKS
010402  000304                          SWAB    R4
010404  020427  000014                  CMP     R4,#12.         ;MAX OF 12 SECTORS
010410  003402                          BLE     3$
010412  012704  000014                  MOV     #12.,R4

010416  010401          3$:     MOV     R4,R1
010420  020400                  CMP     R4,R0
010422  003401                  BLE     4$
010424  010001                  MOV     R0,R1
010426  000301          4$:     SWAB    R1
010430  005401                  NEG     R1
010432  010137  010446                  MOV     R1,5$
010436  010177  170562                  MOV     R1,@RKWC
010442  004537  016704                  JSR     R5,GENBUF       ;GENERATE RANDOM DATA BUFER
010446  000000          5$:     .WORD   0
010450  032514                  .WORD   DBUF            ;STARTING ADDRESS OF DATA BUFER

010452  012777  032514  170546          MOV     #DBUF,@RKBA     ;FROM THIS BUS ADDRESS
010460  012777  000003  170534          MOV     #3,@RKCS        ;WRITE, GO

010466  104417                  CON.RDY                 ;WAIT FOR CONTROL RDY
010470  004737  020112                  JSR     PC,CHKCS        ;ANY ERROR?
010474  104001                          ERROR   1               ;AN ERROR OCCURED WHILE DOING WRITE
```

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

010476  032777  000400  170434          BIT     #BIT8,@SWR      ;TYPE CURRENT STATUS
010504  001435                          BEQ     6$
010506  032700  000377                  BIT     #377,R0
010512  001032                          BNE     6$
010514  104401  010522                  TYPE    ,65$            ;;TYPE ASCIZ STRING
010520  000421                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/WRITTING RANDOM PATTERN, DRIVE /
010564                          64$:
010564  013746  001502                  MOV     @DRV,-(SP)
010570  104403                          TYPOS
010572     001                          .BYTE   1
010573     000                          .BYTE   0

010574  104401  001213                  TYPE    ,$CRLF
010600  104407                  6$:     CKSWR                   ;CHECK FOR SOFTWARE SWR CHANGE
010602  160400                          SUB     R4,R0           ;DECREMENT SECTOR COUNT
010604  003304                          BGT     3$

010606  005303                          DEC     R3              ;MORE DRIVES TO DO?
010610  001247                          BNE     2$

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

010612  032777  000010  170320  BEGEX1: BIT     #SW3,@SWR       ;KW11L CLOCK PRESENT?
010620  001403                          BEQ     BEGNEX          ;IF NOT, SKIP. NOTE:IF KW11L IS
                                                                ;NOT PRESENT SW3 SHOULD NOT BE SET
                                                                ;OTHERWISE, A TIMEOUT WILL OCCUR.
010622  052777  000100  170404          BIS     #BIT6,@KWLS     ;ENABLE KW11L CLOCK
                                                                ;SERVICED AT PRIORITY 7 (KWSRVE)
```

```
2738                                    ;THE PROGRAM IS GOING TO LOOP BACK TO THIS POINT AT THE END OF A PASS.
2739
2740  010630  104416                    BEGNEX: CON.RESET               ;CLEAR EVERYTHING IN DRIVES
2741  010632  012706  001100                    MOV     #STACK,SP       ;RESET STACK
2742  010636  005737  001264                    TST     DRVPRS          ;ANY DRIVES LEFT IN SYSTEM?
2743  010642  001402                            BEQ     QMNGER
2744  010644  004737  014500                    JSR     PC,GENBRQ       ;GO GENERATE 8 COMMANDS FOR THE Q
2745
2746
2747
2748                                    ;CHECK IF THERE IS ANY UNFINISHED COMMAND IN THE Q, WAITING TO BE EXECUTED.
2749                                    ;IF THERE IS NONE, GO TO THE 'GENBRQ' AND GENERATE 8 REQUESTS (COMMANDS),
2750                                    ;AND PUT THEM IN THE Q. IF THERE IS AN UNFINISHED COMMAND, FIND OUT IF
2751                                    ;THERE IS A PRIORITY COMMAND TO BE EXECUTED IMMEDIATELY.
2752
2753  010650  013746  001244            QMNGER: MOV     PPRLVL,-(SP)    ;NEW PSW   RAISE PRIORITY
2754  010654  012746  010662                    MOV     #1$,-(SP)       ;RETURN PC **************
2755  010660  000002                            RTI
2756
2757  010662  005737  001264            1$:     TST     DRVPRS          ;ANY DRIVES IN SYSTEM?
2758  010666  001040                            BNE     2$
2759  010670  104401  010676                    TYPE    ,65$            ;;TYPE ASCIZ STRING
2760  010674  000432                            BR      64$             ;;GET OVER THE ASCIZ
2761                                    ;;65$:  .ASCIZ  <15><12>/HALTING - PRESS CONTINUE TO RESTART AT '200'/<15><12><12><12>
2762  010762                            64$:
2763  010762  000000                            HALT
2764  010764  000137  003376                    JMP     START
2765
2766  010770  012700  001306            2$:     MOV     #KEY,R0
2767  010774  005720                    3$:     TST     (R0)+           ;ANY UNFINISHED COMMAND
2768  010776  100006                            BPL     4$              ;IN Q
2769  011000  020027  001326                    CMP     R0,#KEY+20
2770  011004  001373                            BNE     3$
2771  011006  004737  014500                    JSR     PC,GENBRQ       ;IF NOT, GO GENERATE 8 MORE COMMANDS
2772  011012  000460                            BR      CHFAFN
2773
2774
2775  011014  012700  001306            4$:     MOV     #KEY,R0
2776  011020  032710  010000            5$:     BIT     #BIT12,(R0)     ;ANY HIGH PRIORITY COMMAND IN Q
2777  011024  001404                            BEQ     6$
2778  011026  005710                            TST     (R0)            ;FINISHED?
2779  011030  100402                            BMI     6$              ;YES
2780  011032  105710                            TSTB    (R0)            ;IN EXECUTION?
2781  011034  100165                            BPL     PRICMND         ;NO, GO PROCESS HIGH PRIORITY
2782
2783  011036  005720                    6$:     TST     (R0)+           ;CHECK THE ENTIRE Q
2784  011040  020027  001326                    CMP     R0,#KEY+20
2785  011044  001365                            BNE     5$
2786
2787
2788                                    ;FIND OUT IF THERE IS A WRITE CHECK FUNCTION TO BE DONE IMMEDIATELY AFTER
2789                                    ;A WRITE, THAT WAS DONE PREVIOUSLY.
2790
2791  011046  005737  001456                    TST     WCFLG           ;IS WRITE CHECK TO FOLLOW
2792  011052  100040                            BPL     CHFAFN          ;WRITE? IF NOT, BRANCH
2793                                                                    ;YES
```

```
2794  011054  013700  001456                  MOV     WCFLG,R0        ;GET WC FLAG
2795  011060  042700  177400                  BIC     #177400,R0      ;LOWER BYTE CONTAINS KEY#X2 (OFFSET FROM
2796                                                                    ;KEY), OF THE WRITE FUNCTION
2797  011064  016001  002032                  MOV     PCMND(R0),R1    ;GET POINTER
2798  011070  062700  001306                  ADD     #KEY,R0         ;FROM ADDRESS OF THE KEY
2799                                                                    ;WHICH WAS USED FOR PREVIOUS
2800                                                                    ;WRITE
2801  011074  022761  000002  000002          CMP     #2,2(R1)        ;CHECK THAT THE FUNCTION
2802  011102  001413                          BEQ     7$              ;WAS A WRITE
2803  011104  011037  001162                  MOV     (R0),$REG0      ;GET KEY CONTENTS
2804  011110  016137  000002  001164          MOV     2(R1),$REG1     ;GET THE FUNCTION INDICATED BY THIS KEY
2805  011116  104027                          ERROR   27              ;IF NOT, ERROR
2806                                                                    ;BEFORE DOING A WRITE CHECK A WRITE IS
2807                                                                    ;ALWAYS DONE. OCCURANCE OF THIS ERROR
2808                                                                    ;INDICATES THAT SOMEHOW AN ATTEMPT IS BEING
2809                                                                    ;MADE TO DO WRITE CHECK BEFORE A WRITE IS
2810                                                                    ;PERFORMED. THE KEY IN ERROR MESSAGE WAS
2811                                                                    ;THE ONE WHICH GAVE RISE TO THIS ATTEMPT.
2812                                                                    ;THE FUNCTION CODE IS THE FUNCTION ASSOSCIATED
2813                                                                    ;WITH THAT KEY
2814  011120  005037  001456                  CLR     WCFLG           ;ABORT THIS WRITE CHECK
2815  011124  052710  100000                  BIS     #BIT15,(R0)
2816  011130  000647                          BR      QMNGER
2817  011132  012761  000006  000002  7$:     MOV     #6,2(R1)        ;SET UP BITS FOR WRT CHK
2818                                                                    ;NOW
2819  011140  042710  174340                  BIC     #174340,(R0)    ;CLEAR OUT THE UNNECESSARY
2820                                                                    ;FLAGS FROM THE KEY
2821  011144  052710  000100                  BIS     #BIT6,(R0)      ;SET FLAG FOR WRITE CHECK, FOR
2822                                                                    ;THIS KEY
2823  011150  000137  011612                  JMP     EXCUT
2824
2825                                                                    ;NO HIGH PRIORITY COMMAND IN Q
```

```
2826                                    ;NO HIGH PRIORITY COMMAND WAS FOUND IN THE Q. FIND OUT THE FIRST AVAILABLE
2827                                    ;COMMAND IN THE Q, FOR EXECUTION.
2828  011154  005000            CHFAFN: CLR     R0              ;WAIT FOR ANY IMMEDIATE INTERUPT
2829  011156  005046                    CLR     -(SP)           ;NEW PSW
2830  011160  012746  011166            MOV     #1$,-(SP)       ;RETURN FOR RTI
2831  011164  000002                    RTI
2832
2833
2834  011166  105200            1$:     INCB    R0
2835  011170  001376                    BNE     .-2
2836  011172  013746  001244            MOV     PPRLVL,-(SP)    ;NEW PSW, LOCK OUT CPU
2837  011176  012746  011204            MOV     #2$,-(SP)       ;RETURN FOR RTI *************
2838  011202  000002                    RTI
2839
2840
2841  011204  012700  001306    2$:     MOV     #KEY,R0
2842  011210  005710            CH1:    TST     (R0)            ;UNFINISHED COMMAND?
2843  011212  100436                    BMI     CH2
2844  011214  105710                    TSTB    (R0)            ;IN PROGRESS?
2845  011216  100434                    BMI     CH2
2846  011220  011001                    MOV     (R0),R1
2847  011222  042701  177770            BIC     #177770,R1
2848  011226  105761  001426            TSTB    BUSY(R1)        ;IS THIS DRIVE BUSY?
2849  011232  100426                    BMI     CH2
2850  011234  105761  001436            TSTB    POS(R1)         ;HAS THIS DRIVE BEEN POSITIONED
2851                                                            ;FOR ANY OTHER COMMAND.  IF YES,
2852                                                            ;SKIP.  IF NOT, PROCEED
2853  011240  001023                    BNE     CH2
2854
2855                                                            ;CHECK IF THERE IS AY OTHER COMMAND
2856  011242  010002                    MOV     R0,R2           ;ON A DRIVE THAT IS NOT THE SAME
2857                                                            ;AS THE PREVIOUS DRIVE.  THIS COMMAND
2858  011244  000414                    BR      2$              ;SHOULD NOT BE IN PROGRESS
2859                                                            ;AND SHOULD NOT HAVE BEEN COMPLETED
2860  011246  005712            1$:     TST     (R2)            ;UNFINISHED COMMAND?
2861  011250  100412                    BMI     2$
2862  011252  105712                    TSTB    (R2)            ;IN PROGRESS?
2863  011254  100410                    BMI     2$
2864  011256  011203                    MOV     (R2),R3
2865  011260  042703  177770            BIC     #177770,R3
2866  011264  105763  001426            TSTB    BUSY(R3)        ;IS THE DRIVE BUSY?
2867  011270  100402                    BMI     2$
2868  011272  020301                    CMP     R3,R1           ;IS THIS COMMAND ON THE SAME DRIVE?
2869  011274  001447                    BEQ     POSTION         ;IF YES, GO AND POSITION THE COMMAND
2870                                                            ;POINTED TO BY R0, (BECAUSE THERE IS
2871                                                            ;ONE MORE COMMAND THAT CAN BE PERFORMED
2872                                                            ;ON THE SAME DRIVE)
2873  011276  005722            2$:     TST     (R2)+           ;CHECK REST OF THE Q
2874  011300  020227  001326            CMP     R2,#KEY+20
2875  011304  001360                    BNE     1$
2876                                                            ;IF THERE WAS NO EXECUTABLE COMMAND
2877  011306  000470                    BR      EXNSK           ;ON ANY OTHER DRIVE, THEN EXECUTE
2878                                                            ;COMMAND POINTED TO BY R0
2879
2880  011310  005720            CH2:    TST     (R0)+           ;CHECK ENTIRE Q
2881  011312  020027  001326            CMP     R0,#KEY+20
```

```
011316  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.

011320  105777  167676                  TSTB    @RKCS           ;IF THE CONTROLLER IS BUSY GO TO
011324  100402                          BMI     1$              ;STATUS AND WAIT FOR INTERRUPTS
011326  000137  021436                  JMP     STATUS          ;IF THE CONTROLLER IS NOT BUSY FIND

011332  012700  001306          1$:     MOV     #KEY,R0         ;ANY POSITIONED COMMAND AND EXECUTE
011336  032710  000020          2$:     BIT     #BIT4,(R0)
011342  001414                          BEQ     3$              ;IT
011344  005710                          TST     (R0)            ;MAKE SURE COMMAND IS POSITIONED
011346  100412                          BMI     3$
011350  105710                          TSTB    (R0)            ;COMMAND IS UNFINISHED,
011352  100410                          BMI     3$              ;IT IS NOT IN PROGRESS,
011354  011001                          MOV     (R0),R1         ;THE DRIVE IS NOT IN BUSY
011356  042701  177770                  BIC     #177770,R1
011362  105761  001426                  TSTB    BUSY(R1)
011366  100402                          BMI     3$
011370  000137  011612                  JMP     EXCUT           ;GO EXECUTE THE COMMAND IF THE

011374  005720          3$:     TST     (R0)+           ;ABOVE CONDITIONS ARE SATISFIED
011376  020027  001326                  CMP     R0,#KEY+20              ;CHECK ALL COMMANDS IN Q
011402  001355                          BNE     2$
011404  000137  021436                  JMP     STATUS


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

011410  000137  011612          PRICMN: JMP     EXCUT           ;GO EXECUTE NON-SEEK COMMAND.

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

011414  032710  000020          POSTION: BIT    #BIT4,(R0)      ;ALREADY POSITIONED?
011420  001333                          BNE     CH2             ;YES, GO BACK AND CHECK IF REST OF THE
                                                                ;COMMANDS IN THE Q ARE TO BE POSITIONED

011422  004737  012164                  JSR     PC,POSK         ;GO CHECK, & SET UP FLAGS
011426  004737  020350                  JSR     PC,POSCMND      ;SAVE INFO ABOUT PRESENT & PAST COMAND
011432  012777  000111  167562          MOV     #111,@RKCS      ;POSITION THE COMMAND
011440  005046                          CLR     -(SP)           ;NEW PSW, DROP PRIORITY
011442  012746  011450                  MOV     #1$,-(SP)       ;RETURN FOR RTI
011446  000002                          RTI
011450                          1$:
011450  105737  001535                  TSTB    INT1FL
011454  001775                          BEQ     .-4             ;WAIT FOR INTERRUPT
011456  012777  013256  167554          MOV     #INTHND,@RKVEC  ;RE-ESTABLISH RK INTERRUPT VECTOR
011464  000137  011154                  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

011470  012701  001306          EXNSK:  MOV     #KEY,R1
011474  032711  000020          1$:     BIT     #BIT4,(R1)      ;HEADS POSITIONED?
011500  001412                          BEQ     2$
011502  005711                          TST     (R1)            ;FUNCTION COMPLETED?
011504  100410                          BMI     2$              ;YES, BRANCH
011506  105711                          TSTB    (R1)            ;FUNCTION IN PROGRESS?
011510  100406                          BMI     2$
011512  011102                          MOV     (R1),R2
011514  042702  177770                  BIC     #177770,R2
011520  105762  001426                  TSTB    BUSY(R2)        ;DRIVE BUSY?
011524  100005                          BPL     3$

011526  005721                  2$:     TST     (R1)+           ;CHECK ALL COMMANDS IN Q
011530  020127  001326                  CMP     R1,#KEY+20
011534  001357                          BNE     1$

011536  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.
011540  010137  001536          3$:     MOV     R1,SAVKEY       ;SAVE POINTER TO THE COMMAND(KEY)
                                                                ;WHICH IS ALREADY POSITIONED
011544  004737  012164                  JSR     PC,POSK         ;GO AND POSITION THE COMMAND(KEY)
                                                                ;POINTED TO BY R0
011550  004737  020350                  JSR     PC,POSCMND      ;SAVE INFO ABOUT PAST & PRESENT COMAND
011554  012777  000111  167440          MOV     #111,@RKCS      ;SEEK, IDE, GO
011562  005046                          CLR     -(SP)           ;ALLOW FIRST INTERRUPT
011564  012746  011572                  MOV     #4$,-(SP)       ;RETURN FOR RTI **********
011570  000002                          RTI


011572  105737  001535          4$:     TSTB    INT1FL          ;DID INTERRUPT OCCUR?
011576  001775                          BEQ     4$              ;BR IF NOT
011600  012777  013256  167432          MOV     #INTHND,@RKVEC  ;REESTABLISH INTERRUPT ADDRESS
011606  013700  001536                  MOV     SAVKEY,R0       ;RESTORE THE SAVED POINTER TO KEY
```

```
2982                                                    ;EXECUTE THE COMMAND POINTED TO BY R0
2983                                                    ;R0 CONTAINS THE POINTER TO THE
2984                                                    ;COMMAND KEY
2985  011612  011001            EXCUT:  MOV     (R0),R1 ;GET DRIVE NO
2986  011614  042701  177770            BIC     #177770,R1
2987  011620  005710                    TST     (R0)    ;THIS COMMAND UNFINISHED?
2988  011622  100004                    BPL     1$
2989  011624  011037  001162            MOV     (R0),$REG0      ;GET KEY
2990  011630  104030                    ERROR   30      ;IF NOT, ERROR
2991                                                    ;AN ATTEMPT WAS MADE TO REEXECUTE A COMMAND
2992                                                    ;THAT HAS ALREADY BEEN EXECUTED BEFORE.
2993                                                    ;(ON DRIVE NO. GIVEN IN ERROR MESSAGE)
2994  011632  000512                    BR      ABRT1
2995  011634  105710            1$:     TSTB    (R0)    ;IS IT IN PROGRESS?
2996  011636  100004                    BPL     3$
2997  011640  011037  001162            MOV     (R0),$REG0      ;GET KEY
2998  011644  104030                    ERROR   30      ;IF YES, ERROR
2999
3000  011646  000504                    BR      ABRT1   ;AN ATTEMPT WAS MADE TO REEXECUTE A COMMAND
3001                                                    ;THAT IS IN PROGRESS (ON DRIVE NO. GIVEN
3002                                                    ;IN EROR MESAGE)
3003
3004  011650  105761  001426    3$:     TSTB    BUSY(R1)        ;IS THE DRIVE BUSY?
3005  011654  100004                    BPL     4$
3006  011656  010137  001162            MOV     R1,$REG0        ;GET DRIVE #
3007  011662  104002                    ERROR   2       ;'BUSY' FLAG FOR THE DRIVE (CONTAINED
3008  011664  000470                    BR      ABRT    ;IN R1) IS SET, INDICATING THAT THE DRIVE
3009                                                    ;IS 'BUSY' AND ENGAGED IN AN ACTIVITY,
3010                                                    ;STILL AN ATTEMPT WAS MADE TO INITIATE
3011                                                    ;A FUNCTION ON THIS DRIVE.  THIS IS AN
3012                                                    ;UNEXCEPTED ERROR CONDITION.
3013
3014  011666  105777  167330    4$:     TSTB    @RKCS   ;CONTROL READY SET?
3015  011672  100404                    BMI     5$      ;YES
3016  011674  004737  022032            JSR     PC,GT4RG        ;GET RKCS, ER, DS, DA
3017  011700  104003                    ERROR   3       ;CONTROL READY IS NOT SET.  THIS IS AN
3018  011702  000461                    BR      ABRT    ;UNEXPECTED ERROR CONDITION AT THIS
3019                                                    ;POINT OF EXECUTION, CONTROL SHOULD
3020                                                    ;BE NORMALLY READY.
3021  011704  011002            5$:     MOV     (R0),R2
3022  011706  000302                    SWAB    R2
3023  011710  042702  177770            BIC     #177770,R2      ;FORM THE ADDRESS OF THE
3024  011714  006302                    ASL     R2      ;PARAMETER LIST FOR THIS
3025  011716  010204                    MOV     R2,R4   ;COMMAND
3026  011720  016205  002032            MOV     PCMND(R2),R5
3027
3028  011724  011577  167300            MOV     (R5),@RKDA      ;GET THE FIRST ITEM FROM LIST
3029                                                    ;DISK ADDRES
3030  011730  032777  000100  167260    BIT     #RWS,@RKDS      ;R/W/S RDY SET?
3031  011736  001006                    BNE     6$      ;YES
3032  011740  010137  001250            MOV     R1,$RDRV        ;GET DRIVE #, FOR TYPING SERIAL #
3033  011744  004737  022032            JSR     PC,GT4RG        ;GET RKCS, ER, DS, DA
3034  011750  104004                    ERROR   4       ;R/W/S RDY IS NOT SET FOR THE DRIVE.
3035  011752  000435                    BR      ABRT    ;A (NON-SEEK) FUNCTION WAS SUPPOSED
3036                                                    ;TO BE EXECUTED ON THIS DRIVE.  THIS IS
3037                                                    ;AN UNEXPECTED ERROR CONDITIONS AT THIS
```

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

```
3067                                                                    ;THE FUNCTION TO BE EXECUTED IS A
3068                                                                    ;WRITE FUNCTION
3069  012064  016537  000004  012104  WRFNC:  MOV     4(R5),1$        ;GET # OF WORDS TO BE WRITTEN
3070  012072  016537  000006  012106          MOV     6(R5),2$        ;GET STARTING BA OF BUFFER
3071  012100  004537  016704                  JSR     R5,GENBUF       ;GO GENERATE BUFFER
3072  012104  000000                  1$:     .WORD   0               ;SAVE # OF WORDS
3073  012106  000000                  2$:     .WORD   0               ;SAVE STARTING BUS ADDRESS OF BUFFER
3074  012110  052703  000101                  BIS     #101,R3         ;SET IDE AND GO BIT
3075  012114  013777  012104  167102          MOV     1$,@RKWC        ;SET WORD COUNT
3076  012122  013777  012106  167076          MOV     2$,@RKBA        ;SET BUS ADDRESS
3077  012130  004737  020372                  JSR     PC,FNCMND       ;SAVE INFO ABOUT PAST & PRESENT COMAND
3078
3079  012134  010377  167062                  MOV     R3,@RKCS        ;ISSUE WRITE,IDE,GO
3080  012140  052710  000200                  BIS     #BIT7,(R0)      ;SET FLAG INDICATING FUNCTION IN
3081  012144  052704  000200                  BIS     #BIT7,R4        ;PROGRESS
3082
3083  012150  110461  001426                  MOVB    R4,BUSY(R1)     ;SET FLAG INDICATING DRIVE BUSY
3084  012154  110437  001534                  MOVB    R4,INTFLG       ;SET FLAG FOR INTERUPT USE
3085                                                                    ;R4 CONTAINS OFFSET TO THE COMMAND KEY
3086                                                                    ;(FROM 'KEY') BITS 0-3, BIT 7 IS SET
3087  012160  000137  021436                  JMP     STATUS
```

```
                                        POSK:   MOV     (R0),R1         ;INITIAL CHECKING PRIOR TO DOING
                                                BIC     #177770,R1      ;POSITIONING COMMAND POINTED TO BY R0
                                                TSTB    BUSY(R1)        ;DRIVE BUSY?
                                                BPL     1$
                                                MOV     R1,$REG0        ;GET DRIVE # FOR WHICH 'BUSY' IS SET
                                                ERROR   2               ;'BUSY' FLAG FOR THE DRIVE (CONTAINED
                                                BR      7$              ;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.
                                        1$:     TSTB    @RKCS           ;CONTROL READY SET?
                                                BMI     2$              ;YES
                                                JSR     PC,GT4RG        ;GET RKCS, ER, DS, DA
                                                ERROR   3               ;CONTROL READY IS NOT SET.  THIS IS AN
                                                BR      6$              ;UNEXPECTED ERROR CONDITION AT THIS
                                                                        ;POINT OF EXECUTION, CONTROL SHOULD
                                                                        ;BE NORMALLY READY.
                                        2$:     MOV     (R0),R2
                                                SWAB    R2
                                                BIC     #177770,R2
                                                ASL     R2
                                                MOV     PCMND(R2),R3    ;STARTING WORD OF COMMAND TABLE
                                                MOV     (R3),@RKDA
                                                BIT     #RWS,@RKDS      ;R/W/S RDY SET?
                                                BNE     3$              ;YES
                                                MOV     R1,SRDRV        ;GET DRIVE #, FOR TYPING SERIAL #
                                                JSR     PC,GT4RG        ;GET RKCS, ER, DS, DA
                                                ERROR   4               ;R/W/S RDY IS NOT SET FOR THE DRIVE.  A
                                                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.
                                        3$:     MOVB    R2,BUSY(R1)     ;SET FLAG INDICATING DRIVE BUSY
                                                BISB    #BIT7,BUSY(R1)
                                                BIT     #BIT3,(R0)      ;IS THIS A SEEK COMMAND
                                                BNE     4$
                                                MOVB    #377,POS(R1)    ;SET FLAG INDICATING, THIS DRIVE POSITIONS
                                                BIS     #BIT4,(R0)      ;SET FLAG INDICATING THIS COMMAND
                                                BR      5$
                                        4$:     BIS     #BIT7,(R0)      ;POSITIONED.  SET FLAG INDICATING
                                                                        ;FUNCTION IN PROGRESS
                                        5$:     MOV     #INT1SK,@RKVEC  ;SET UP RK11 VECTOR
                                                MOV     PPRLVL,@RKSTAT  ;PSW ON INTERRUPT
                                                CLRB    INT1FL          ;CLEAR FLAG INDICATING - INTERRUPT
                                                RTS     PC              ;OCCURRED

                                        6$:     JSR     PC,DRVABT       ;ABORT THE FUNCTION
                                                JSR     PC,CHKDRV       ;GO CHECK. DRIVE ERRORS
                                                NOP
                                        7$:     TST     (SP)+           ;POP THE RETURN ADDRESS
                                                JMP     @MNGER
```

```
3141                                    ;AFTER ISSUING A SEEK FOR POSITIONING, AN INTERRUPT IS ALLLOWED TO TAKE
3142                                    ;*PLACE. THIS HANDLER SERVICES THE FIRST INTERRUPT, WHICH OCCURS AS A RESULT
3143                                    ;OF THE SEEK BEING ACCEPTED.
3144
3145  012376  105237  001535            INT1SK: INCB    INT1FL          ;INDICATE THAT INTERRUPT TOOK PLACE
3146  012402  053766  001244  000002            BIS     PPRLVL,2(SP)    ;CPU PRIORITY TO 5 ON RETURN FROM
3147                                                                    ;INTERRUPT
3148
3149  012410  004737  020272                    JSR     PC,CHKCRDY      ;CONTROL READY SET?
3150  012414  104005                            ERROR   5               ;CONTROL READY NOT SET AFTER THE FIRST
3151                                                                    ;INTERRUPT ON INITIATING SEEK
3152
3153  012416  032777  020000  166576            BIT     #SCP,@RKCS      ;SCP BIT SET?
3154  012424  001403                            BEQ     1$
3155  012426  004737  022222                    JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
3156                                                                    ;TYPING SERIAL DRIVE #
3157  012432  104012                            ERROR   12              ;SCP SET AFTER FIRST INTERRUPT ON
3158                                                                    ;ACCEPTING SEEK.  A SEEK WAS INITIATED,
3159                                                                    ;AS A RESULT OF WHICH (THIS) FIRST
3160                                                                    ;INTERRUPT OCCURRED, BUT SCP SHOULD
3161                                                                    ;NOT BE SET AFTER THE FIRST INTERRUPT.
3162                                                                    ;(IT GETS SET ONLY AFTER THE SEEK
3163                                                                    ;IS DONE).
3164                                                                    ;THIS IS THE FIRST INTERRUPT
3165                                                                    ;AFTER ISSUING SEEK
3166  012434  017700  166562            1$:     MOV     @RKCS,R0
3167  012440  042700  177760                    BIC     #177760,R0      ;CHECK THERE IS A SEEK FUNCTION
3168  012444  022700  000010                    CMP     #10,R0          ;IN RKCS
3169  012450  001403                            BEQ     2$
3170  012452  004737  022032                    JSR     PC,GT4RG        ;GET RKCS,ER,DS,DA
3171  012456  104006                            ERROR   6               ;RKCS DOES NOT CONTAIN 'SEEK' FUNCTION.
3172                                                                    ;A SEEK WAS INITIATED AS A RESULT OF WHICH
3173                                                                    ;(THIS) FIRST INTERRUPT OCCURRED.  RKCS
3174                                                                    ;SHOULD CONTAIN THE SEEK FUNCTION BITS.
3175                                                                    ;BUT IT DID NOT
3176
3177  012460  017700  166544            2$:     MOV     @RKDA,R0        ;GET THE DRIVE # FROM RKDA
3178  012464  042700  017777                    BIC     #17777,R0
3179  012470  000241                            CLC
3180  012472  006100                            ROL     R0
3181  012474  006100                            ROL     R0
3182  012476  006100                            ROL     R0
3183  012500  006100                            ROL     R0
3184  012502  010001                            MOV     R0,R1
3185  012504  105761  001426                    TSTB    BUSY(R1)        ;CHECK THIS DRIVE NO. WAS BUSY
3186  012510  100403                            BMI     3$              ;OK, IF IT WAS
3187  012512  010137  001162                    MOV     R1,$REG0        ;GET DRIVE #(FROM RKDA)
3188  012516  104007                            ERROR   7               ;'BUSY' FLAG FOR THE DRIVE WAS NOT SET.
3189                                                                    ;THE DRIVE # (IN RKDA) WHICH CAUSED (?)
3190                                                                    ;THIS (FIRST) INTERRUPT SHOULD HAVE BEEN
3191                                                                    ;'BUSY' (SOFTWARE FLAG).  NOTE WHENEVER
3192                                                                    ;ANY OPERATION IS INITIATED ON ANY DRIVE
3193                                                                    ;'BUSY' FLAG FOR THAT DRIVE IS SET.
3194  012520  116100  001426            3$:     MOVB    BUSY(R1),R0
3195  012524  042700  177600                    BIC     #177600,R0      ;FORM THE ADDRESS OF
3196  012530  062700  001306                    ADD     #KEY,R0         ;THIS KEY
```

```
3197
3198  012534  032710  000020            BIT     #BIT4,(R0)      ;IS THE KEY ACTIVE FOR 'POSITIONING'?
3199  012540  001003                    BNE     4$
3200  012542  010137  001162            MOV     R1,$REG0        ;GET DRIVE NUMBER
3201  012546  104010                    ERROR   10              ;POSITIONING FLAG (IN THE KEY) IS NOT
3202                                                            ;SET FOR THE INTERRUPTING DRIVE (GIVEN
3203                                                            ;IN EROR MESAGE)
3204
3205  012550  105761  001436      4$:   TSTB    POS(R1)         ;IS THE 'POSITIONING' FLAG SET?
3206  012554  001003                    BNE     5$
3207  012556  010137  001162            MOV     R1,$REG0        ;GET DRIVE # FROM RKDA
3208  012562  104010                    ERROR   10              ;THIS INTERRUPT IS SUPPOSED TO BE AS
3209                                                            ;A RESULT OF INITIATING A (POSITIONING)
3210                                                            ;SEEK.  WHENEVER A SEEK IS INITIATED
3211                                                            ;TO POSITION THE HEADS THE POSITIONING
3212                                                            ;FLAG (POS#) IS SET. OCCURANCE OF THIS
3213                                                            ;ERROR INDICATED THAT THE DRIVE #
3214                                                            ;(FROM RKDA)GIVING THIS INTERRUPT DID
3215                                                            ;NOT HAVE ITS POSITIONING FLAG SET.
3216
3217  012564  005777  166432      5$:   TST     @RKCS           ;ANY ERROR?
3218  012570  100044                    BPL     8$              ;NO - RETURN
3219
3220  012572  032737  000040  001474          BIT     #BIT5,ERCODE
3221  012600  001012                    BNE     6$              ;SAME ERROR
3222  012602  042737  000040  001474          BIC     #BIT5,ERCODE
3223  012610  001026                    BNE     7$
3224  012612  052737  000040  001474          BIS     #BIT5,ERCODE
3225
3226  012620  004737  022222            JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
3227                                                            ;TYPING SERIAL DRIVE #
3228  012624  104011                    ERROR   11              ;BIT 15, 'ERR' SET IN RKKCS AFTER FIRST
3229                                                            ;INTERRUPT - WHICH OCCURRED AFTER A
3230                                                            ;POSITIONING SEEK WAS INITIATED.  RKER
3231                                                            ;WILL CONTAIN ERROR BIT/S
3232  012626  042710  000020      6$:   BIC     #BIT4,(R0)      ;CLEAR 'POSITIONING' BIT
3233  012632  105061  001426            CLRB    BUSY(R1)        ;CLEAR 'BUSY' FLAG
3234  012636  105061  001436            CLRB    POS(R1)         ;CLEAR 'POSITIONING' FLAG
3235
3236  012642  004737  016006            JSR     PC,CLRERR       ;CLEAR THE ERROR
3237  012646  052710  010000            BIS     #BIT12,(R0)     ;INDICATE HIGH PRIORITY ON
3238  012652  105261  001446            INCB    RETRY(R1)       ;INCREMENT RETRY COUNT
3239  012656  126127  001446  000003    CMPB    RETRY(R1),#3    ;DONE RETRIES?
3240  012664  003406                    BLE     8$              ;NO
3241
3242  012666  004737  014426      7$:   JSR     PC,DRVABT       ;ABORT THE FUNCTION
3243  012672  005061  001446            CLR     RETRY(R1)
3244  012676  005037  001474            CLR     ERCODE
3245
3246  012702  000002      8$:   RTI
```

```
3247                                                    ;THIS ROUTINE IS ENTERED UPON COMPLETION
3248                                                    ;OF A SEEK OPERATION UNDER INTERRUPT MODE
3249
3250  012704  004737  020272      SKCMP:  JSR     PC,CHKCRDY      ;CONTROL READY SET?
3251  012710  104013                      ERROR   13              ;CONTROL RDY NOT SET, AFTER A SEEK
3252                                                            ;DONE INTERRUPT!
3253
3254  012712  017746  166300      1$.     MOV     @RKDS,-(SP)     ;GET DRIVE # THAT INTERRUPTED
3255  012716  004737  022170              JSR     PC,CROTLF       ;ROTATE BITS 15,14,13 TO POSITION
3256                                                            ;0,1,2
3257  012722  012601                      MOV     (SP)+,R1        ;GET THE ROTATED BITS (DRIVE ID BITS)
3258  012724  105761  001426              TSTB    BUSY(R1)        ;CHECK THAT DRIVE WAS BUSY
3259  012730  100403                      BMI     2$
3260  012732  004737  022032              JSR     PC,GT4RG
3261  012736  104014                      ERROR   14              ;'BUSY' FLAG FOR THE INTERRUPTING DRIVE
3262                                                            ;(RKDS) WAS NOT SET.  DRIVE # (15,14,13 - RKDS
3263                                                            ;INTERRUPTED AFTER SEEK WAS COMPLETE, BUT
3264                                                            ;'BUSY' FLAG CORRSPONDING TO THAT DRIVE
3265                                                            ;WAS CLEAR.  IF THE CORRECT DRIVE WAS
3266                                                            ;INTERRUPTING 'BUSY' FLAG SHOULD HAVE
3267                                                            ;BEEN SET.
3268  012740  116100  001426      2$:     MOVB    BUSY(R1),R0     ;FORM THE ADDRESS OF THE
3269  012744  042700  177600              BIC     #177600,R0      ;KEY
3270  012750  062700  001306              ADD     #KEY,R0
3271  012754  032710  000020              BIT     #BIT4,(R0)      ;CHECK THAT POSITION BIT WAS SET
3272  012760  001003                      BNE     3$
3273  012762  010137  001162              MOV     R1,$REGO        ;GET DRIVE NUMBER
3274  012766  104010                      ERROR   10              ;POSITIONING FLAG (IN KEY) IS NOT SET
3275                                                            ;SET FOR INTERRUPTING DRIVE (GIVEN IN EROR
3276                                                            ;MESAGE)
3277  012770  105761  001436      3$:     TSTB    POS(R1)         ;CHECK POSITIONING FLAG WAS SET
3278  012774  001003                      BNE     4$
3279  012776  010137  001162              MOV     R1,$REGO
3280  013002  104010                      ERROR   10              ;'POSITIONING' FLAG WAS CLEAR.  WHEN
3281                                                            ;DOING A (POSITIONING) SEEK, THE FLAG
3282                                                            ;'POS#' IS SET TO INDICATE THAT DRIVE
3283                                                            ;IS POSITIONING HEADS (SEEKING).  THIS
3284                                                            ;ERROR INDICATFS THAT FOR THE INTERRUPTING
3285                                                            ;DRIVE (RKDS) POSITIONING FLAG WAS
3286                                                            ;NOT SET.
3287  013004  004737  016540      4$:     JSR     PC,CHKDRV       ;GO, CHECK FOR DRV ERORS (DPL,DRU,DRY,WPS,
3288  013010  000501                      BR      PFTERR          ;IF THERE IS A DRV ERROR, RETURN HERE
3289                                                            ;THE DRIVE HAS BEEN DESELECTED, ALL
3290                                                            ;FURTHER COMANDS ON THAT DRIVE ABORTED
3291                                                            ;RETURRN HERE IF NO DRIVE ERRORS
3292  013012  017777  166200  166210      MOV     @RKDS,@RKDA     ;GET THE DRIVE # FROM DRIVE ID BITS
3293                                                            ;IN RKDS AND ADDRESS THE DRIVE
3294  013020  032777  000100  166170      BIT     #RWS,@RKDS      ;R/W/S RDY SET?
3295  013026  001005                      BNE     5$              ;YES
3296  013030  004737  022032              JSR     PC,GT4RG        ;GET RKCS,ER,DS,DA
3297  013034  010137  001250              MOV     R1,SRDRV        ;GET DRIVE #, FOR TYPING SERIAL #
3298  013040  104015                      ERROR   15              ;R/W/S RDY NOT SET FOR INTERRUPTING
3299                                                            ;DRIVE (RKDS), AFTER SEEK WAS COMPLETED.
3300  013042  032777  001000  166146  5$: BIT     #SIN,@RKDS      ;'SIN' ERROR?
3301  013050  001007                      BNE     PSINER          ;YES, BRANCH
3302
```

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

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

```
3353                                                                    ;ENTER THIS ROUTINE WHEN AN INTERRUPT
3354                                                                    ;OCCURS. NOTE THERE IS ONE OTHER
3355                                                                    ;INTERRUPT ROUTINE- 'INTISK'
3356  013256  105037  001534          INTHND: CLRB    INTFLG            ;CLEAR FLAG TO INDICATE THAT
3357                                                                    ;AN INTERRUPT OCCURED.
3358  013262  013766  001244  000002          MOV     PPRLVL,2(SP)      ;CPU PRIORITY TO 5 ON RTI
3359  013270  004737  020272                  JSR     PC,CHKCRDY        ;CONTROL READY SET?
3360  013274  104024                          ERROR   24                ;CONTROL RDY CLEAR AFTER INTRUPT ON COMPLETION
3361                                                                    ;OF FUNCTION, IT SHOULD BE SET
3362  013276  032777  020000  165716  1$:     BIT     #SCP,@RKCS        ;SCP SET? SEARCH COMPLETE?
3363  013304  001403                          BEQ     2$
3364  013306  004737  012704                  JSR     PC,SKCMP          ;GO TO SEEK COMPLETE ROUTINE
3365  013312  000002                          RTI
3366  013314  017746  165710          2$:     MOV     @RKDA,-(SP)       ;GET THE DRIVE # TO WHICH
3367                                                                    ;THE COMMAND WAS ISSUED UNDER
3368                                                                    ;INTERRUPT MODE
3369  013320  004737  022170                  JSR     PC,CROTLF         ;ROTATE BITS 15,14,13 TO POSITION
3370                                                                    ;0,1,2
3371  013324  012605                          MOV     (SP)+,R5          ;POP OFF THE DRIVE # FROM THE STACK
3372  013326  105765  001426                  TSTB    BUSY(R5)          ;CHECK THAT DRIVE WAS BUSY
3373  013332  100403                          BMI     3$
3374  013334  010537  001162                  MOV     R5,$REGO
3375  013340  104007                          ERROR   7                 ;'BUSY' FLAG FOR THE INTERUPTING DRIVE
3376                                                                    ;WAS NOT SET.  WHENEVER ANY ACTIVITY
3377                                                                    ;IS INITIATED ON A DRIVE, THE (SOFTWARE)
3378                                                                    ;'BUSY' FLAG IS SET TO INDICATE THAT
3379                                                                    ;DRIVE IS BUSY DOING SOMETHING.
3380                                                                    ;OCCURANCE OF THIS ERROR INDICATES
3381                                                                    ;THAT THE 'BUSY' FLAG FOR THE INTERRUPTING
3382                                                                    ;DRIVE (RKDA) IS CLEAR!
3383  013342  105065  001436          3$:     CLRB    POS(R5)           ;CLEAR THE POSITION FLAG
3384  013346  116500  001426                  MOVB    BUSY(R5),R0       ;GET THE ADDRESS OF THE
3385  013352  042700  177760                  BIC     #177760,R0        ;COMMAND KEY
3386  013356  062700  001306                  ADD     #KEY,R0
3387  013362  032710  000010                  BIT     #BIT3,(R0)        ;CHECK IT'S NOT A SEEK FUNCTION
3388  013366  001401                          BEQ     4$
3389  013370  104020                          ERROR   20                ;(SOFTWARE) FLAG FOR THE INTERRUPTING
3390                                                                    ;DRIVE (RKDA) INDICATES THAT A SEEK
3391                                                                    ;FUNCTION WAS BEING EXECUTED ON THE
3392                                                                    ;DRIVE.  IF THAT'S THE CASE, SCP BIT SHOULD
3393                                                                    ;HAVE SET ON SEEK DONE INTRUPT, BUT IT
3394                                                                    ;WAS NOT.  NOTE, HERE THERE IS A CHANGE
3395                                                                    ;OF MULTIPLE ERRORS OR ERROR
3396                                                                    ;MISINTERPRETATION
3397  013372  105710                  4$:     TSTB    (R0)              ;CHECK THAT FUNCTION IN
3398  013374  100403                          BMI     5$                ;PROGRESS BIT WAS SET
3399  013376  010537  001162                  MOV     R5,$REGO          ;GET DRIVE NUMBER
3400  013402  104031                          ERROR   31                ;IF NOT, ERROR
3401                                                                    ;'FUNCTION IN PROGRESS' FLAG (IN THE KEY)
3402                                                                    ;WAS NOT SET FOR THE INTERRUPTING DRIVE.
3403                                                                    ;IF THIS DRIVE WAS BUSY DOING THE
3404                                                                    ;FUNCTION, THE ABOVE FLAG SHOULD BE SET.
3405  013404  011002                  5$:     MOV     (R0),R2           ;CHECK THAT THE INTERRUPTING
3406  013406  042702  177770                  BIC     #177770,R2        ;DRIVE (IN RKDA) IS THE SAME AS
3407  013412  020502                          CMP     R5,R2             ;THE DRIVE IN THE KEY
3408  013414  001405                          BEQ     6$
```

```
3409  013416  010537  001162                  MOV     R5,$REGO        ;GET EXPCTD DRIVE NO.
3410  013422  010237  001164                  MOV     R2,$REGI        ;GET DRIVE NO. RECVD
3411  013426  104032                          ERROR   32              ;UNEXPECTED DRIVE NO. INTERRUPTED
3412  013430  006302                  6$:     ASL     R2
3413  013432  011003                          MOV     (R0),R3
3414  013434  010037  001536                  MOV     R0,SAVKEY
3415  013440  000303                          SWAB    R3
3416  013442  042703  177770                  BIC     #177770,R3      ;GET POSITION OF THE PARAMETER
3417  013446  006303                          ASL     R3              ;LIST FROM THE KEY
3418  013450  017704  165546                  MOV     @RKCS,R4
3419  013454  042704  177761                  BIC     #177761,R4      ;CLEAR ALL BUT FUNCTION CODE
3420  013460  016305  002032                  MOV     PCMND(R3),R5    ;CHECK THAT THE FUNCTION IN
3421                                                                  ;RKCS AND THE KEY ARE SAME.
3422  013464  126504  000002                  CMPB    2(R5),R4
3423
3424  013470  001406                          BEQ     7$              ;IF NOT, ERROR
3425  013472  016537  000002  001162          MOV     2(R5),$REGO     ;GET EXPCTD FUNCTION CODE IN RKCS
3426  013500  010437  001164                  MOV     R4,$REGI        ;GET CODE RECVD
3427  013504  104033                          ERROR   33              ;FUNCTION CODE THAT IS IN RKCS AFTER THIS
3428                                                                  ;INTERRUPT OCCURED IS NOT THE EXPECTED ONE
3429  013506  042710  000200          7$:     BIC     #BIT7,(R0)      ;CLEAR 'FUNCTION IN PROGRESS' BIT
3430  013512  142761  000200  001426          BICB    #BIT7,BUSY(R1)  ;CLEAR BUSY FLAG FOR THIS DRIVE
3431  013520  004737  016540                  JSR     PC,CHKDRV       ;CHECK FOR DRIVE ERRORS
3432  013524  000002                          RTI                     ;IF THERE WAS ANY DRIVE EROR
3433                                                                  ;DESELECT THE DRIVE AND EXIT
3434                                                                  ;IF NO ERROR, RETURN HERE
3435  013526  032777  001000  165462          BIT     #SIN,@RKDS      ;SIN ERROR?
3436  013534  001426                          BEQ     10$
3437  013536  004737  022222                  JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
3438                                                                  ;TYPING SERIAL DRIVE #
3439  013542  104016                          ERROR   16              ;SIN ERROR
3440  013544  004737  016152                  JSR     PC,CLRSIN
3441  013550  004737  016264                  JSR     PC,SINCNT       ;HELP COUNT OF SIN'S. IF MORE
3442                                                                  ;THAN MAXM ALLOWABLE, DESELECT THE DRIVE
3443  013554  000002                          RTI                     ;RETURN HERE IF COUNT=MAX
3444                                                                  ;RETURN HERE IF LESS THAN MAX
3445  013556  105261  001446                  INCB    RETRY(R1)
3446  013562  122761  000003  001446          CMPB    #3,RETRY(R1)
3447  013570  001405                          BEQ     8$
3448  013572  052710  010000                  BIS     #BIT12,(R0)
3449  013576  042710  000020                  BIC     #BIT4,(R0)
3450  013602  000002                          RTI
3451
3452  013604  004737  014426          8$:     JSR     PC,DRVABT       ;ABORT THIS FUNCTION
3453  013610  000002                  9$:     RTI
3454
3455  013612  032777  040000  165402  10$:    BIT     #HE,@RKCS       ;HARD ERROR?
3456  013620  001076                          BNE     HRDERR
3457  013622  032777  100000  165372          BIT     #ERR,@RKCS      ;SOFT ERROR?
3458  013630  001002                          BNE     SFTERR          ;YES
3459  013632  000137  014300                  JMP     NOEROR          ;NO
```

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

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

014016 032777 010000 165174 HRDERR: BIT     #SKE,@RKER          ;SKE?
014024 001421                       BEQ     1$
                                                                
014026 032737 000004 001474         BIT     #BIT2,ERCODE        ;ALREADY WORKING ON A SK ERROR?
014034 001051                       BNE     CMNERR              ;YES - IGNORE THIS ONE
014036 042737 000004 001474         BIC     #BIT2,ERCODE        ;ANY OTHER ERROR?
014044 001064                       BNE     CMNABT              ;YES - ABORT THIS FUNCTION
014046 052737 000004 001474         BIS     #BIT2,ERCODE        ;SET THE SK ERROR BIT
                                                                
014054 005262 001602                INC     SKECN(R2)           ;INCREMENT COUNT FOR SKE
014060 005262 001562                INC     HECN(R2)            ;ON THIS DRIVE.
                                                                
014064 104421 002062                TYPMSG  ,MSG1
                                                                
014070 032777 167740 165122 1$:     BIT     #167740,@RKER       ;ANY HE BESIDES SKE?
014076 001417                       BEQ     2$
                                                                
014100 032737 000010 001474         BIT     #BIT3,ERCODE        ;ALREADY WORKING ON A HE ERROR?
014106 001024                       BNE     CMNERR              ;YES - IGNORE THIS ONE
014110 042737 000010 001474         BIC     #BIT3,ERCODE        ;ANY OTHER ERROR?
014116 001037                       BNE     CMNABT              ;YES - ABORT THIS FUNCTION
014120 052737 000010 001474         BIS     #BIT3,ERCODE        ;SET THE HE ERROR BIT
                                                                
014126 005262 001562                INC     HECN(R2)            ;INCREMENT HE COUNT FOR
014132 104421 002104                TYPMSG  ,MSG4               ;THIS DRIVE
                                                                
014136 104421 002120        2$:     TYPMSG  ,MSG5               ;PRINT 'ON DOING'
014142 004737 021736                JSR     PC,TYPFN            ;GO PRINT OUT THE FUNCTION
                                                                ;WHICH GAVE THIS ERROR
                                                                
014146 004737 022064                JSR     PC,GETINF
014152 004737 022226                JSR     PC,GTSDRV           ;SAVE DRIVE #, FOR TYPING SERIAL #
014156 104022                       ERROR   22                  ;HARD ERROR ON DOING DATA XFER
                                                                
014160 105261 001446        CMNERR: INCB    RETRY(R1)           ;INCREMENT RETRY COUNT
014164 122761 000003 001446         CMPB    #3,RETRY(R1)        ;3 TRIES IN ALL?
014172 001411                       BEQ     CMNABT              ;YES, ABORT THIS FUNCTION
                                                                
014174 052777 010000 165334         BIS     #BIT12,@SAVKEY      ;INDICATE HIGH PRIORITY ON RETRY
014202 042777 000020 165326         BIC     #BIT4,@SAVKEY       ;CLEAR 'POSITIONING HEADS'. IF SET
014210 004737 016006                JSR     PC,CLRERR           ;CLEAR THIS ERROR
014214 000002                       RTI
                                                                
014216 005037 001474        CMNABT: CLR     ERCODE              ;CLEAR ERROR CODE
014222 104421 002532                TYPMSG  ,MSG27
014226 004737 014426                JSR     PC,DRVABT           ;ABORT THE FUNCTION
014232 032777 010000 164760         BIT     #SKE,@RKER          ;SEEK ERROR?
014240 001416                       BEQ     3$                  ;NO
014242 104416                       CON.RESET                   ;RESET THE CONTROLLER
014244 010137 001502                MOV     R1,@DRV             ;SET DRIVE NUMBER
014250 000241                       CLC                         ;CLEAR THE 'C' BIT
014252 006037 001502                ROR     @DRV                ;LEFT JUSTIFY DRIVE NUMBER
```

```
3556  014256  006037  001502                  ROR     QDRV            ;LEFT JUSTIFY DRIVE NUMBER
3557  014262  006037  001502                  ROR     QDRV            ;LEFT JUSTIFY DRIVE NUMBER
3558  014266  006037  001502                  ROR     QDRV            ;LEFT JUSTIFY DRIVE NUMBER
3559  014272  104420                          DRV.RESET               ;RESET THE DRIVE
3560  014274  104416                          CON.RESET               ;RESET THE CONTROLLER
3561  014276  000002              3$:         RTI                     ;THIS DATA TRANSFER
```

```
3562                                                                    ;IF THERE WAS NO HARD OR
3563                                                                    ;SOFT ERROR ON DATA TRANSFER
3564                                                                    ;ENTER HERE
3565  014300  105761  001446          NOEROR: TSTB    RETRY(R1)
3566  014304  001406                          BEQ     1$
3567  014306  104421  002604                  TYPMSG  ,MSG28
3568  014312  116146  001446                  MOVB    RETRY(R1),-(SP)
3569  014316  104403                          TYPOS
3570  014320     002                          .BYTE   2
3571  014321     000                          .BYTE   0
3572
3573  014322  005037  001474          1$:     CLR     ERCODE
3574  014326  105061  001446                  CLRB    RETRY(R1)
3575  014332  042777  010000  165176          BIC     #BIT12,@SAVKEY  ;CLEAR PRIORITY BIT IF SET
3576  014340  004737  021006                  JSR     PC,STATSTC      ;GO, COLLECT STATISTICS ON THIS
3577                                                                  ;DATA TRANSFER
3578
3579  014344  022704  000004                  CMP     #4,R4           ;WAS IT A 'READ' FUNCTION?
3580  014350  001002                          BNE     2$
3581  014352  004737  017110                  JSR     PC,DATCHK       ;IF IT WAS 'READ', CHECK THE DATA
3582
3583
3584
3585  014356  022704  000002          2$:     CMP     #2,R4           ;WAS IT A 'WRITE' FUNCTION?
3586  014362  001011                          BNE     3$
3587  014364  032777  000040  165144          BIT     #BIT5,@SAVKEY   ;IS 'WRT CHK' TO FOLLOW THIS
3588  014372  001412                          BEQ     4$              ;'WRT' FUNCTION?
3589  014374  052703  100000                  BIS     #BIT15,R3       ;SET FLAG BIT TO INDICATE
3590                                                                  ;THAT WRITE CHECK SHOULD
3591                                                                  ;FOLLOW THIS WRITE
3592  014400  010337  001456                  MOV     R3,WCFLG        ;SET UP WC FLAG TO INDICATE
3593                                                                  ;THE ABOVE. THE LOWER BYTE
3594                                                                  ;CONTAINS THE POINTER TO THE
3595                                                                  ;COMMAND LIST, WHICH WILL
3596                                                                  ;BE USED, FOR DOING WRITE CHECK
3597  014404  000407                          BR      5$              ;IF WRITE CHECK IS TO BE DONE, DON'
3598                                                                  ;SET BIT 15 OF THE KEY TILL WRT CHK
3599                                                                  ;IS DONE
3600
3601  014406  022704  000006          3$:     CMP     #6,R4           ;WRT CHK FUNCTION?
3602  014412  001002                          BNE     4$
3603  014414  005037  001456                  CLR     WCFLG           ;CLEAR WR CHK FLAG
3604
3605  014420  052710  100000          4$:     BIS     #BIT15,(R0)     ;SET FLAG TO INDICATE THIS
3606  014424  000002                  5$:     RTI                     ;FUNCTION IS COMPLETED
```

```
3607                                                    ;ABORT THE FUNCTION ON DRIVE POINTED TO BY R1
3608                                                    ;CLEAR ASSOCIATED FLAGS
3609                                                    ;DROP THE DRIVE IF MORE THAN 8. ABORTS
3610
3611  014426  010246                DRVABT: MOV     R2,-(SP)        ;SAVE R2
3612  014430  105061  001446                CLRB    RETRY(R1)
3613  014434  052710  104000                BIS     #104000,(R0)    ;INDICATE THAT FUNCTION IS ABORTED
3614  014440  042710  010000                BIC     #BIT12,(R0)     ;CLEAR HIGH PRIORITY BIT IF SET
3615  014444  010102                        MOV     R1,R2
3616  014446  006302                        ASL     R2
3617  014450  005262  001672                INC     ABORT(R2)
3618  014454  026227  001672  000010        CMP     ABORT(R2),#10
3619  014462  003402                        BLE     1$
3620  014464  000137  016312                JMP     DSELCT          ;DROP THE DRIVE
3621  014470  012602                1$:     MOV     (SP)+,R2        ;RESTORE R2
3622  014472  104401  002165                TYPE    ,MSG10          ;TYPE "ABORTED"
3623  014476  000207                        RTS     PC              ;RETURN
```

```
                                ;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').

014500  104407                  GEN8RQ: CKSWR
014502  005337  002056                  DEC     REPCNT          ;DECREMENT REPETITION COUNT
014506  001025                          BNE     1$              ;CONTINUE IF NOT 0
014510  013737  001236  002056          MOV     PCNTR,REPCNT    ;REESTABLISH REPETITION COUNT FOR EXERCISER
014516  104401  014524                  TYPE    ,65$            ;;TYPE ASCIZ STRING
014522  000407                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:   .ASCIZ  <15><12>/END OF PASS/
014542                          64$:
014542  013746  001100                  MOV     $PASS,-(SP)
014546  005216                          INC     (SP)
014550  104405                          TYPDS
014552  004737  021116                  JSR     PC,REPSTAT
014556  000137  022704                  JMP     $EOP

014562  032777  000400  164350  1$:     BIT     #SW8,@SWR
014570  001422                          BEQ     2$
014572  032777  000001  164340          BIT     #SW00,@SWR      ;SWITCH 0 SET ?
014600  001410                          BEQ     8$              ;BR IF NOT
014602  005727                          TST     (PC)+           ;CHECK INDICATOR
014604  000000                  7$:     .WORD   0               ;TYPE TIME INDICATOR
014606  001013                          BNE     2$              ;BR IF TIME ALREADY TYPED
014610  005237  014604                  INC     7$              ;INCREMENT THE INDICATOR
014614  004737  026660                  JSR     PC,TIMTYP       ;TYPE THE TIME (IF SWR 03 SE
014620  000406                          BR      2$              ;CONTINUE
014622  005037  014604          8$:     CLR     7$              ;CLEAR THE INDICATOR
014626  104401  001213                  TYPE    ,$CRLF
014632  004737  021116                  JSR     PC,REPSTAT
014636  032777  000040  164274  2$:     BIT     #SW5,@SWR       ;HALT?
014644  001401                          BEQ     3$              ;NO
014646  000000                          HALT                    ;YES, PRESSING CONTINUE RESUMES PROG EXECUTION

014650  004737  022656          3$:     JSR     PC,CHDPRS       ;CHECK IF ANY DRIVES PRESENT
014654  012700  001306          4$:     MOV     #KEY,R0
014660  010004                          MOV     R0,R4
014662  005001                          CLR     R1
014664  010120                  5$:     MOV     R1,(R0)+        ;CLEAR THE 8 COMMAND KEYS
014666  062701  000400                  ADD     #400,R1         ;BITS 8,9,10 INDICATE THEIR
014672  022701  004000                  CMP     #4000,R1        ;POSITION 0,1,2,3,4,5,6,7
014676  001372                          BNE     5$
014700  012700  001326                  MOV     #CMND,R0        ;CLEAR THE PARAMETER TABLE
014704  010005                          MOV     R0,R5
014706  012701  177740                  MOV     #-40,R1         ;FOR THE 8 COMMANDS IN Q
014712  005020                  6$:     CLR     (R0)+           ;(8X4) WORDS IN ALL
014714  005201                          INC     R1
014716  001375                          BNE     6$
```

```
3680  014720  004737  015770                    JSR     PC,CLRFLGS      ;CLEAR THE FLAGS PERTAINING TO THE 8
3681                                                                      ;COMMANDS IN THE Q
3682                                                                      ;R4 CONTAINS 'KEY'
3683                                                                      ;R5 CONTAINS 'CMND'
3684  014724  022737  000001  001264  GEN1:   CMP     #1,DRVPRS       ;ONLY 1 DRV PRESENT?
3685  014732  001002                    BNE     22$             ;NO
3686  014734  005000                    CLR     R0              ;YES
3687  014736  000420                    BR      3$
3688  014740  004737  025576          22$:    JSR     PC,$RAND        ;GENERATE A RANDOM NUMBER
3689                                                                      ;GET A RANDOM DRIVE NUMBER
3690                                                                      ;FROM THE AVAILABLE DRIVES
3691  014744  025730                    RSDRVL
3692
3693  014746  013746  025732                    MOV     RSDRVH,-(SP)    ;PUT LOW DIVIDEND ON STACK
3694  014752  005046                    CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
3695                                                                      ;IT ON STACK
3696  014754  013746  001520                    MOV     DRMAP,-(SP)     ;PUSH DIVISOR ON STACK
3697  014760  004737  025212                    JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
3698  014764  005726                    TST     (SP)+           ;DISCARD THE REMAINDER. QUOTIENT IS
3699                                                                      ;NOW ON TOP OF THE STACK
3700  014766  012600                    MOV     (SP)+,R0
3701  014770  020037  001264                    CMP     R0,DRVPRS       ;MAKE SURE CORRECT MAPPING IS DONE
3702  014774  001001                    BNE     3$
3703  014776  005300                    DEC     R0              ;'QDRVE' CONTAINS RANDOM DRIVE NO.
3704  015000  005037  001502          3$:     CLR     QDRV
3705  015004  116037  001254  001502          MOVB    PDR(R0),QDRV
3706  015012  105737  001502                    TSTB    QDRV            ;TEST IF TYPE F DRIVE
3707  015016  100016                    BPL     32$             ;NOT IF POSITIVE
3708
3709  015020  142737  000200  001502          BICB    #200,QDRV       ;CLEAR THE FLAG BIT
3710  015026  132737  000001  001502          BITB    #1,QDRV         ;ODD OR EVEN DRIVE ADDRESS
3711  015034  001404                    BEQ     31$
3712
3713  015036  005737  015724                    TST     ODDEVN          ;MUST HAVE BEEN AN ODD ONE
3714  015042  001730                    BEQ     GEN1            ;INSURE THAT ODDS WHAT WE WANT
3715  015044  000403                    BR      32$
3716
3717  015046  005737  015724          31$:    TST     ODDEVN          ;MUST HAVE BEEN AN EVEN ONE
3718  015052  001332                    BNE     22$             ;NO GOOD - TRY AGAIN
3719
3720  015054  004737  025576          32$:    JSR     PC,$RAND        ;GENERATE A RANDOM NUMBER
3721  015060  025734                    RSFUNL
3722
3723  015062  013746  025736                    MOV     RSFUNH,-(SP)    ;PUT LOW DIVIDEND ON STACK
3724  015066  005046                    CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
3725                                                                      ;IT ON STACK
3726  015070  013746  001526                    MOV     FNMAP,-(SP)     ;PUSH DIVISOR ON STACK
3727  015074  004737  025212                    JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
3728  015100  005726                    TST     (SP)+           ;DISCARD THE REMAINDER. QUOTIENT IS
3729                                                                      ;NOW ON TOP OF THE STACK
3730  015102  021627  000003                    CMP     (SP),#3         ;MAKE SURE CORRECT FUNCTION IS SELCTD
3731  015106  001001                    BNE     20$
3732  015110  005316                    DEC     (SP)
3733  015112  012637  001512          20$:    MOV     (SP)+,QFNC      ;THE FUNCTION THAT CAN BE PERFORMED
3734
3735  015116  004737  025576                    JSR     PC,$RAND        ;GENERATE A RANDOM NUMBER
```

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

```
3792  015306  012703  000312          MOV     #312,R3         ;COMPUTE # OF SECTORS AVAILABLE
3793  015312  163703  001504          SUB     QCYL,R3         ;BEYOND THE CYLINDER SELECTED
3794  015316  012746  000030          MOV     #30,-(SP)               ;;PUT THE MULTIPLER ON THE STACK
3795  015322  010346                  MOV     R3,-(SP)                ;;PUT THE MULTIPLICAND ON THE STACK
3796  015324  004737  025100          JSR     PC,@#$MULT      ;;CALL THE MULTPLY ROUTINE
3797  015330  012616                  MOV     (SP)+,(SP)      ;;DISREGARD THE MSB'S
3798  015332  012603                  MOV     (SP)+,R3                ;;GET THE LSB'S OF THE PRODUCT
3799  015334  060103                  ADD     R1,R3           ;COMPUTE TOTAL # OF SECTORS AVAILABLE
3800  015336  012746  000400          MOV     #400,-(SP)              ;;PUT THE MULTIPLER ON THE STACK
3801  015342  010346                  MOV     R3,-(SP)                ;;PUT THE MULTIPLICAND ON THE STACK
3802  015344  004737  025100          JSR     PC,@#$MULT      ;;CALL THE MULTPLY ROUTINE
3803  015350  012616                  MOV     (SP)+,(SP)      ;;DISREGARD THE MSB'S
3804  015352  012603                  MOV     (SP)+,R3                ;;GET THE LSB'S OF THE PRODUCT
3805  015354  010301                  MOV     R3,R1           ;COMPUTE TOTAL # OF WORDS-SPACE
3806                                                          ;AVAILABLE ON DISK FROM THE SELECTED
3807                                                          ;CYL #
3808                                                          ;COMPUTE MAXM WORD COUNT BASED ON
3809                                                          ;AVAILABLE MEMORY SPACE.
3810  015356  004737  025576  21$:    JSR     PC,$RAND        ;GENERATE RANDOM NUMBER
3811  015362  025744                  RSBAL
3812
3813  015364  013746  025746          MOV     RSBAH,-(SP)     ;PUT LOW DIVIDEND ON STACK
3814  015370  005046                  CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
3815                                                          ;IT ON STACK
3816  015372  013746  001530          MOV     BAMAP,-(SP)     ;PUSH DIVISOR ON STACK
3817  015376  004737  025212          JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
3818  015402  005726                  TST     (SP)+           ;DISCARD THE REMAINDER. QUOTIENT IS
3819                                                          ;NOW ON TOP OF THE STACK
3820  015404  012637  001514          MOV     (SP)+,QBUSAD    ;BE USED
3821  015410  006337  001514          ASL     QBUSAD
3822  015414  063737  002052  001514  ADD     BASEBA,QBUSAD   ;FORM THE RANDOM BUS-ADDRESS
3823                                                          ;BY ADDING RANDOM OFFSET TO
3824                                                          ;THE BASE BUS-ADDRESS
3825  015422  013703  002054          MOV     MAXBA,R3        ;COMPUTE # OF WORDS THAT
3826  015426  163703  001514          SUB     QBUSAD,R3       ;CAN BE TRANSFERRED, USING THE
3827  015432  000241                  CLC
3828  015434  006003                  ROR     R3              ;ABOVE GENERATED BUS-ADDRESS WITHOUT
3829  015436  010302                  MOV     R3,R2           ;CAUSING A NXM
3830
3831  015440  020201          10$:    CMP     R2,R1           ;SELECT SMALLER OF THE TWO
3832  015442  103401                  BLO     11$             ;WORD-COUNTS THAT WILL BE
3833                                                          ;USED FOR GENERATING A RANDOM
3834  015444  010103                  MOV     R1,R3           ;WORD COUNT)
3835
3836                                                          ;R3 CONTAINS THE MAXM WORD COUNT
3837                                                          ;POSSIBLE. COMPUTE THE WORD COUNT
3838                                                          ;MAPPING FACTOR FROM THIS.
3839  015446          11$:
3840
3841  015446  012746  177777          MOV     #177777,-(SP)   ;PUT LOW DIVIDEND ON STACK
3842  015452  005046                  CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
3843                                                          ;IT ON STACK
3844  015454  010346                  MOV     R3,-(SP)        ;PUSH DIVISOR ON STACK
3845  015456  004737  025212          JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
3846  015462  005726                  TST     (SP)+           ;DISCARD THE REMAINDER. QUOTIENT IS
3847                                                          ;NOW ON TOP OF THE STACK
```

```
3848  015464  005216                          INC     (SP)
3849  015466  012637  001532                  MOV     (SP)+,WCMAP     ;WORD COUNT MAPPING FACTOR
3850
3851  015472  004737  025576                  JSR     PC,$RAND        ;GENERATE A RANDOM NUMBER
3852  015476  025750                          RSWCL
3853
3854  015500  013746  025752                  MOV     RSWCH,-(SP)     ;PUT LOW DIVIDEND ON STACK
3855  015504  005046                          CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
3856                                                                  ;IT ON STACK
3857  015506  013746  001532                  MOV     WCMAP,-(SP)     ;PUSH DIVISOR ON STACK
3858  015512  004737  025212                  JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
3859  015516  005726                          TST     (SP)+           ;DISCARD THE REMAINDER. QUOTIENT IS
3860                                                                  ;NOW ON TOP OF THE STACK
3861  015520  012637  001516                  MOV     (SP)+,QWRCNT    ;'QWRCNT' CONTAINS THE RANDOM
3862                                                                  ;WORD COUNT THAT WILL BE USED
3863  015524  005737  001516                  TST     QWRCNT          ;MAKE SURE THE WORD COUNT IS
3864  015530  003004                          BGT     12$             ;NOT 0
3865  015532  005437  001516                  NEG     QWRCNT          ;TAKE CARE OF ZERO AND NEG NMBRS
3866  015536  005237  001516                  INC     QWRCNT
3867  015542  113700  001502          12$:    MOVB    QDRV,R0
3868  015546  000241                          CLC
3869  015550  006000                          ROR     R0
3870  015552  006000                          ROR     R0              ;POSITION THE DRIVE NUMBER IN
3871  015554  006000                          ROR     R0              ;BITS 15,14,13
3872  015556  006000                          ROR     R0
3873  015560  013701  001504                  MOV     QCYL,R1
3874  015564  000301                          SWAB    R1
3875  015566  000241                          CLC
3876  015570  006001                          ROR     R1              ;POSITION THE CYLINDER NUMBER
3877  015572  006001                          ROR     R1              ;IN BITS 12-5
3878  015574  006001                          ROR     R1
3879  015576  050100                          BIS     R1,R0
3880  015600  053700  001510                  BIS     QSEC,R0         ;R0 CONTAINS THE COMPLETE DISK
3881  015604  053700  001506                  BIS     QSUR,R0         ;ADDRESS
3882  015610  010025                          MOV     R0,(R5)+        ;INSERT RKDA IN THE PARAMETER TABLE
3883                                                                  ;(FOR THE 8 COMMANDS)
3884                                                                  ;WHICH FUNCTION?
3885                                                                  ;0-READ CHECK, 1-READ, 2-WRITE
3886  015612  022737  000001  001512          CMP     #1,QFNC
3887  015620  001412                          BEQ     2$              ;READ
3888  015622  003014                          BGT     14$             ;READ CHECK
3889  015624  012725  000002          1$:     MOV     #2,(R5)+        ;WRITE FUNCTION CODE
3890  015630  023727  025752  077777          CMP     RSWCH,#77777    ;SHOULD WRITE CHECK BE DONE
3891  015636  101010                          BHI     15$             ;AFTER THE WRITE?
3892  015640  052714  000040                  BIS     #BIT5,(R4)      ;SET FLAG IN KEY TO INDICATE
3893                                                                  ;THAT WRITE CHECK SHOULD BE
3894                                                                  ;DONE FOLLOWING THE WRITE
3895  015644  000405                          BR      15$
3896
3897  015646  012725  000004          2$:     MOV     #4,(R5)+        ;READ FUNCTION CODE
3898  015652  000402                          BR      15$
3899
3900  015654  012725  000012          14$:    MOV     #12,(R5)+       ;READ CHECK FUNCTION CODE
3901
3902  015660  013715  001516          15$:    MOV     QWRCNT,(R5)     ;INSERT THE WORD COUNT (RKWC)
3903  015664  005425                          NEG     (R5)+           ;(2'S COMPLEMENT)
```

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

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


015726  004737  016006        EXCRLUP: JSR     PC,CLRERR      ;WAIT FOR OTHER DRIVES TO GET DONE
                                                            ;THEN ISSUE A CONTROL RESET
015732  010146                         MOV     R1,-(SP)
015734  012701  001306                 MOV     #KEY,R1        ;CLEAR OUT THE COMMAND-KEYS
015740  042721  114220        1$:      BIC     #114220,(R1)+
015744  020127  001326                 CMP     R1,#KEY+20
015750  001373                         BNE     1$
015752  012601                         MOV     (SP)+,R1
015754  004737  015770                 JSR     PC,CLRFLGS     ;CLEAR OUT THE VARIOUS FLAGS PERTAINING
                                                            ;TO THE 8 COMMANDS IN THE Q
015760  012706  001100                 MOV     #STACK,SP      ;REESTABLISH STACK POINTER
015764  000137  010650                 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.

015770  012700  001426        CLRFLGS: MOV     #BUSY,R0       ;CLEAR THE 8 BUSY FLAGS
015774  005020                1$:      CLR     (R0)+
015776  020027  001462                 CMP     R0,#QSCNT+2    ;ALL DONE?
016002  001374                         BNE     1$             ;NO
016004  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.

016006  010446          CLRERR: MOV     R4,-(SP)        ;SAVE R4, R4 ON THE STACK
016010  010546                  MOV     R5,-(SP)
016012  005005                  CLR     R5
016014  005077  163210          CLR     @RKDA

016020  105765  001426  1$:     TSTB    BUSY(R5)        ;WAS THIS DRIVE BUSY SEEKING?
016024  100035                  BPL     4$              ;NO
016026  005037  001472          CLR     TIMER
016032  032777  000100  163156  2$:     BIT     #RWS,@RKDS      ;R/W/S SET?
016040  001015                  BNE     3$              ;YES
016042  005237  001472          INC     TIMER           ;KEEP TIME
016046  001371                  BNE     2$              ;WAIT FOR R/W/S RDY
016050  004737  022222          JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
                                                        ;TYPING SERIAL DRIVE #
016054  104004                  ERROR   4               ;R/W/S READY DID NOT SET
                                                        ;FOR THIS DRIVE, WAITED LONG ENOUGH.
016056  032777  001000  163132          BIT     #SIN,@RKDS      ;SIN ERROR ON THIS DRIVE?
016064  001403                  BEQ     3$
016066  004737  022222          JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE # FOR
                                                        ;TYPING SERIAL DRIVE #
016072  104016                  ERROR   16              ;SIN OCCURED ON THIS DRIVE

016074  116504  001426  3$:     MOVB    BUSY(R5),R4     ;FORM THE ADDRESS OF THE
016100  042704  177760          BIC     #177760,R4      ;KEY WHICH MADE THIS DRIVE
016104  062704  001306          ADD     #KEY,R4 ;BUSY

016110  042714  010000          BIC     #10000,(R4)     ;CLEAR HIGH PRIORITY BIT, IF SET
016114  105065  001426          CLRB    BUSY(R5)        ;MAKE THIS DRIVE FREE, AVAILABLE

016120  062777  020000  163102  4$:     ADD     #20000,@RKDA    ;ADDRESS THE NEXT POSSIBLE DRIVE
016126  005205                  INC     R5              ;INCREMENT COUNT
016130  022705  000010          CMP     #10,R5          ;ALL DONE
016134  001331                  BNE     1$              ;NO

016136  004737  020340          JSR     PC,CRCMND       ;SAVE INFO ABOUT THE PAST & PRESENT COMAND
016142  104416                  CON.RESET
016144  012605                  MOV     (SP)+,R5        ;RESTORE R4,R5
016146  012604                  MOV     (SP)+,R4
016150  000207                  RTS     PC              ;RETURN
```

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

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

```
4058                                    ;DSELCT
4059                                    ;THIS ROUTINE IS ENTERED WHEN A DRIVE IS TO BE DESELECTED (TAKEN
4060                                    ;OUT OF SELECTION LIST), BECAUSE THE FATAL ERRORS ON THAT DRIVE
4061                                    ;HAS REACHED A MAXIMUM COUNT.  R1 CONTAINS THE DRIVE NUMBER THAT
4062                                    ;THAT IS TO BE DESELECTED. THE DRIVE IS DESELECTED IF 1. TOTAL SIN COUN
4063                                    ;FOR THAT DRIVE REACHES THE MAXIMUM ALLOWABLE  2.IF  A FATAL ERROR
4064                                    ;LIKF DRIVE UNSAFE, DRIVE POWER LOW OCCURS.  3. IF  WPS GETS SET, OR DRY
4065                                    ;IS CLEAR.
4066
4067
4068
4069  016312  012705  001253    DSELCT: MOV     #PDR-1,R5
4070  016316  013702  001264            MOV     DRVPRS,R2       ;NUMBER OF DRIVES BEING TESTED
4071  016322  062702  001253            ADD     #PDR-1,R2       ;FOR END ADDRESS OF TABLE
4072  016326  005205            1$:     INC     R5              ;LOCATE THE DRIVE (TO BE
4073  016330  111503                    MOVB    (R5),R3         ;DESELECTED) IN THE TABLE
4074  016332  042703  177600            BIC     #177600,R3      ;DROP THE F FLAG
4075  016336  020301                    CMP     R3,R1           ;IS THIS THE ONE
4076  016340  001403                    BEQ     2$              ;CONTAINING AVAILABLE DRIVES
4077  016342  020502                    CMP     R5,R2           ;FINISHED ?
4078  016344  103770                    BLO     1$              ;BR IF NOT
4079  016346  000472                    BR      11$             ;DRIVE WAS NOT FOUND IN TABLE, EXIT
4080  016350  111502            2$:     MOVB    (R5),R2         ;GET THE DRIVE NUMBER
4081  016352  020527  001264    5$:     CMP     R5,#PDR+10      ;IS THIS DRIVE # THE LAST ENTRY IN TABLE?
4082  016356  001406                    BEQ     4$              ;YES
4083  016360  010504                    MOV     R5,R4
4084  016362  005205                    INC     R5              ;IF NOT, TAKE OUT THIS DRIVE # FROM
4085  016364  112524            3$:     MOVB    (R5)+,(R4)+     ;THE MIDDLE AND PUSH UP THE
4086  016366  022704  001263            CMP     #PDR+7,R4       ;REST OF THE ENTRIES
4087  016372  001374                    BNE     3$
4088  016374  105065  177777    4$:     CLRB    -1(R5)          ;CLEAR LAST ENTRY IN TABLE
4089
4090                                                            ;THE DRIVE # TYPED OUT WAS DESELECTED
4091                                                            ;BECAUSE ERROR COUNT EXCEEDED THE
4092                                                            ;MAXIMUM ALLOWABLE
4093  016400  104401  002336            TYPE    ,MSG19          ;TYPE "DRVE DROPPED"
4094  016404  010146                    MOV     R1,-(SP)        ;PUSH DRIVE NUMBER ON STACK
4095  016406  104403                    TYPOS                   ;TYPE IT ON THE TERMINAL
4096  016410     001                    .BYTE   1
4097  016411     000                    .BYTE   0
4098  016412  004737  026766            JSR     PC,SNOTYP       ;GO TYPE OUT SERIAL NO OF THE DRIVE,
4099                                                            ;IF SW 1 IS SET.
4100  016416  005337  001264            DEC     DRVPRS          ;DECREMENT THE TOTAL NUMBER OF
4101                                                            ;DRIVES PRESENT
4102  016422  004737  022656            JSR     PC,CHDPRS       ;CHECK IF ANY DRIVES PRESENT
4103                                                            ;IF NONE GOT TO END OF PASS, $EOP
4104
4105  016426  012746  177777            MOV     #177777,-(SP)   ;PUT LOW DIVIDEND ON STACK
4106  016432  005046                    CLR     -(SP)           ;CLEAR HIGH DIVIDEND AND PUSH
4107                                                            ;IT ON STACK
4108  016434  013746  001264            MOV     DRVPRS,-(SP)    ;PUSH DIVISOR ON STACK
4109  016440  004737  025212            JSR     PC,@#$DIV       ;GO TO THE 'DIVIDE' SUBROUTINE
4110  016444  005726                    TST     (SP)+           ;DISCARD THE REMAINDER. QUOTIENT IS
4111                                                            ;NOW ON TOP OF THE STACK
4112  016446  012637  001520            MOV     (SP)+,DRMAP     ;TO BE USED FOR GENERATING RANDOM
4113                                                            ;DRIVE NUMBERS.
```

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

```
4135                                    ;CHKDRV
4136                                    ;THIS ROUTINE CHECKS FOR FATAL ERRORS OF THE DRIVE LIKE DPL, DRV
4137                                    ;WPS.  IF ANY ONE OF THESE ERRORS OCCUR THE DRIVE IS DESELECTED
4138                                    ;AND NO MORE FUNCTIONS WILL BE PERFORMED ON THAT DRIVE.  R1
4139                                    ;CONTAINS THE DRIVE NUMBER TO BE CHECKED.
4140
4141                                    ;*NOTE 1: IN THE ERROR MESSAGE WHERE RKDA IS PRINTED OUT, IT GIVES
4142                                    ;*ONLY THE DRIVE NUMBER (NOT CYLINDER, SUFACE AND SECTOR).
4143
4144                                    ;CALL:  JSR     PC,CHKDRV
4145                                    ;       ----- RETURN HERE IF ANY FATAL ERROR OCCURED
4146                                    ;       ----- RETURN HERE IF THERE WAS NO FATAL ERROR
4147
4148  016540  017746  162464       CHKDRV: MOV     @RKDA,-(SP)     ;SAVE RKDA
4149  016544  010146                       MOV     R1,-(SP)        ;GET DRIVE #
4150  016546  000241                       CLC
4151  016550  006016                       ROR     (SP)
4152  016552  006016                       ROR     (SP)
4153  016554  006016                       ROR     (SP)
4154  016556  006016                       ROR     (SP)
4155  016560  012677  162444               MOV     (SP)+,@RKDA     ;ADDRESS THE DRIVE TO BE CHECKED
4156  016564  032777  010000  162424       BIT     #DPL,@RKDS      ;DRIVE POWER LOW?
4157  016572  001403                       BEQ     1$
4158  016574  004737  022222               JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE #
4159                                                                ;FOR TYPING SERIAL NUMBER
4160  016600  104035                       ERROR   35              ;DRIVE POWER LO, *NOTE 1 ABOVE
4161  016602  032777  002000  162406  1$:  BIT     #DRU,@RKDS      ;DRIVE
4162  016610  001403                       BEQ     2$
4163  016612  004737  022222               JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE #
4164                                                                ;FOR TYPING SERIAL NUMBER
4165  016616  104036                       ERROR   36              ;DRIVE UNSAFE BIT IS SET
4166                                                                ;*NOTE 1 ABOVE
4167  016620  032777  000040  162370  2$:  BIT     #WPS,@RKDS      ;WRITE PROTECT SET?
4168  016626  001403                       BEQ     3$
4169  016630  004737  022222               JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE #
4170                                                                ;FOR TYPING SERIAL NUMBER
4171  016634  104037                       ERROR   37              ;WPS SET, CHECK WRTE PROTECT SWTCH ON DRIVE
4172                                                                ;*NOTE 1 ABOVE
4173  016636  032777  000200  162352  3$:  BIT     #DRY,@RKDS      ;DRIVE READY CLEAR?
4174  016644  001004                       BNE     4$
4175  016646  004737  022222               JSR     PC,RG4SDR       ;GET RKCS, ER, DS, DA AND DRIVE #
4176                                                                ;FOR TYPING SERIAL NUMBER
4177  016652  104034                       ERROR   34              ;DRIVE READY CLEAR, SHOULD BE SET
4178                                                                ;*NOTE 1 ABOVE
4179  016654  000411                       BR      5$
4180
4181  016656  032777  012040  162332  4$:  BIT     #12040,@RKDS    ;ANY ERROR?
4182  016664  001005                       BNE     5$              ;YES
4183  016666  012677  162336               MOV     (SP)+,@RKDA     ;RESTORE RKDA
4184  016672  062716  000002               ADD     #2,(SP)         ;ADJUST RETURN ADDRESS
4185  016676  000207                       RTS     PC
4186
4187  016700  000137  016312          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.

016704  104414          GENBUF: SAVREG                  ;SAVE REGISTERS
016706  016504  000002          MOV     2(R5),R4        ;GET STARTING ADDRESS OF BUFFER
016712  011503                  MOV     (R5),R3         ;GET WORD COUNT (# OF WORDS TO
                                                        ;BE GENERATED)

016714  017702  162310          MOV     @RKDA,R2
016720  010237  025756  1$:     MOV     R2,RSDTH        ;GET THIS RANDOM SEED
016724  010237  025754          MOV     R2,RSDTL
016730  005137  025754          COM     RSDTL           ;GET LO RANDOM SEED

016734  022703  177400          CMP     #-400,R3        ;IF THE BUFFER IS MORE THAN
016740  003003                  BGT     2$              ;ONE SECTOR (400 WORDS) LONG,
016742  010305                  MOV     R3,R5           ;GENERATE THE BUFFER IN SUCH
016744  005003                  CLR     R3              ;A WAY THAT EACH SECTOR
016746  000404                  BR      3$              ;BEGINS WITH RANDOM DATA

016750  062703  000400  2$:     ADD     #400,R3         ;WORDS GENERATED USING THAT
016754  012705  177400          MOV     #-400,R5        ;SECTOR ADDRESS AS THE RANDOM
                                                        ;SEED
016760  010524          3$:     MOV     R5,(R4)+        ;FIRST WORD OF EVERY SECTOR IS
                                                        ;A WORD COUNT (2'S COMP) INDICATING #
                                                        ;OF WORDS ACTUALLY WRITTEN IN THAT SECTOR
016762  005205                  INC     R5              ;ALL DONE?
016764  001427                  BEQ     8$

016766  004737  025576  4$:     JSR     PC,$RAND        ;GENERATE DATA WORDS
016772  025754                  RSDTL
016774  012737  000002  017104  MOV     #2,RCNT
017002  013737  025756  017106  MOV     RSDTH,RNUM
017010  000406                  BR      6$
017012  005337  017104  5$:     DEC     RCNT
017016  001763                  BEQ     4$
017020  013737  025754  017106  MOV     RSDTL,RNUM
017026  005737  017106  6$:     TST     RNUM
017032  001767                  BEQ     5$
017034  013724  017106          MOV     RNUM,(R4)+      ;FILL THE BUFFER. DON'T USE
017040  005205                  INC     R5              ;ALL DONE?
017042  001351                  BNE     4$              ;NO

017044  005703          8$:     TST     R3              ;ANY MORE DATA WORDS (FOR
017046  001412                  BEQ     10$             ;REST OF THE SECTORS)?
                                                        ;YES
017050  010246                  MOV     R2,-(SP)
017052  042716  177760          BIC     #177760,(SP)    ;(ABSOLUTE DISK ADDRESS & I'S
```

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

```
                                        ;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.

017110  104414                  DATCHK: SAVREG                          ;SAVE R0-R5
017112  016303  002032                  MOV     PCMND(R3),R3            ;GET ADDRESS OF THE PARAMETER
                                                                        ;TABLE
017116  016304  000004                  MOV     4(R3),R4                ;GET WORD COUNT (2'S COMP)
017122  016305  000006                  MOV     6(R3),R5                ;GET BUS ADDRESS
017126  011301                          MOV     (R3),R1                 ;GET DISK ADDRESS

017130  010146                          MOV     R1,-(SP)
017132  004737  022170                  JSR     PC,CROTLF               ;ROTATE BITS 15,14,13 TO
                                                                        ;0,1,2
017136  012602                          MOV     (SP)+,R2                ;POP OFF DRIVE # FROM THE STACK
017140  006302                          ASL     R2
017142  062702  001712                  ADD     #DATER,R2               ;FORM THE ADDRESS OF DATA ERROR COUNT
                                                                        ;FOR THIS DRIVE
017146  012737  177764  001540          MOV     #-14,ECOUNT
017154  011337  025756                  MOV     (R3),RSDTH              ;CREATE RANDOM SEEDS TO
017160  013737  025756  025754          MOV     RSDTH,RSDTL             ;BE USED FOR CHECKING DATA
017166  005137  025754                  COM     RSDTL

017172  022704  177400          1$:     CMP     #-400,R4                ;DATA IS CHECKED IN 1 SECTOR
017176  003003                          BGT     2$                      ;BLOCKS. EACH SECTOR IS GENERATED
017200  010403                          MOV     R4,R3                   ;USING THAT SECTOR ADDRESS
017202  005004                          CLR     R4                      ;AS THE RANDOM SEED
017204  000404                          BR      3$

017206  062704  000400          2$:     ADD     #400,R4
017212  012703  177400                  MOV     #-400,R3
017216  012500                  3$:     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
017220  005200                          INC     R0                      ;INCREMENT COUNT OF # OF WORDS  (WRITEN)
                                                                        ;IN THE SECTOR
017222  005203                          INC     R3                      ;INCRMENT COUNT OF DATA WORDS TO BE CHECKED
017224  001465                          BEQ     14$                     ;BRANCH, IF DONE

017226  004737  025576          4$:     JSR     PC,$RAND                ;GENERATE RANDOM DATA WORD
017232  025754                          RSDTL
017234  012737  000002  017104          MOV     #2,RCNT
017242  013737  025756  017106          MOV     RSDTH,RNUM
017250  000406                          BR      10$
017252  005337  017104          9$:     DEC     RCNT
017256  001763                          BEQ     4$
017260  013737  025754  017106          MOV     RSDTL,RNUM
017266  005737  017106          10$:    TST     RNUM
017272  001767                          BEQ     9$

017274  005700                          TST     R0
```

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

```
                                .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
                                ;*$LST3K WILL CONTAIN THE LAST BANK AS A SAF
                                ;*$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

017466  010046                  $SIZE:  MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
017470  010146                          MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
017472  010246                          MOV     R2,-(SP)        ;;SAVE R2 ON THE STACK
017474  010346                          MOV     R3,-(SP)        ;;SAVE R3 ON THE STACK
017476  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
017502  013746  000006                  MOV     @#ERRVEC+2,-(SP)
017506  010600                          MOV     SP,R0           ;;SAVE THE STACK POINTER
                                ;;SET THE ERRVEC PS TO THE PRESENT PS
017510  104400                          TRAP                    ;;PUSH OLD PSW AND PC ON STACK
017512  012637  000006                  MOV     (SP)+,@#ERRVEC+2        ;;SAVE THE PSW IN @#ERRVEC+2
017516  012701  003776                  MOV     #3776,R1        ;;SETUP ADDRESS
017522  105727                          TSTB    (PC)+           ;;USE MEMORY MANAGEMENT?
017524  000200                  $KT11:  .WORD   200             ;;SET TO USE MEMORY MANAGEMENT
017526  100062                          BPL     $CORE           ;;BR IF NO
017530  012737  017666  000004          MOV     #$KTNEX,@#ERRVEC        ;;SET FOR TIMEOUT
017536  005737  177572                  TST     @#SR0           ;;KT11 ARE YOU THERE?
017542  052737  100000  017524          BIS     #100000,$KT11   ;;YES--SET KT11 KEY
017550  005046                          CLR     -(SP)           ;;INITIALIZE FOR "PAR" LOADING
017552  012702  172340                  MOV     #KIPAR0,R2      ;;ADDRESS OF FIRST "PAR"
017556  012703  000010                  MOV     #^D8,R3         ;;LOAD EIGHT "PAR.'S" AND EIGHT "PDR.'S"
017562  012762  077406  177740  1$:     MOV     #77406,-40(R2)  ;;PDR = 4K, UP, READ/WRITE
017570  011622                          MOV     (SP),(R2)+      ;;LOAD "PAR"
017572  062716  000200                  ADD     #200,(SP)       ;;UPDATE FOR NEXT "PAR"
017576  077307                          SOB     R3,1$           ;;LOOP UNTIL ALL EIGHT ARE LOADED
017600  012742  177600                  MOV     #177600,-(R2)   ;;SETUP KIPAR7 FOR I/O
017604  005042                          CLR     -(R2)           ;;SETUP KIPAR6 FOR TESTING
017606  012737  017624  000004          MOV     #2$,@#ERRVEC    ;;CATCH TIMEOUT IF NO SR3
017614  012737  000020  172516          MOV     #20,@#SR3       ;;ENABLE 22 BIT MODE
017622  000401                          BR      3$              ;;THIS PDP-11 HAS A SR3 REGISTER
017624  022626                  2$:     CMP     (SP)+,(SP)+     ;;CLEAN OFF THE STACK--NO SR3
017626  005237  177572          3$:     INC     @#SR0           ;;TURN ON MEMORY MANAGEMENT
017632  012737  017656  000004          MOV     #$KTOUT,@#ERRVEC        ;;SET FOR TIME OUT
017640  005737  143776          4$:     TST     @#143776        ;;TRAP ON NON-EX-MEM
017644  062712  000040                  ADD     #40,(R2)        ;;MAKE A 1K STEP
017650  023712  172356                  CMP     @#KIPAR7 (R2)   ;;LAST ONE?
017654  101371                          BHI     4$              ;;NO--TRY IT
017656  011202                  $KTOUT: MOV     (R2),R2         ;;GET LAST BANK+1
017660  005037  177572                  CLR     @#SR0           ;;TURN OFF MEMORY MANAGEMENT
017664  000421                          BR      $SIZEX
017666  042737  100000  017524  $KTNEX: BIC     #100000,$KT11   ;;KT11 NON-EXISTENT
```

```
017674 012737 017724 000004 $CORE:  MOV    #$CROUT,@#ERRVEC ;;SET FOR TIMEOUT
017702 005002                       CLR    R2               ;;SET UP BANK
017704 062701 004000        1$:     ADD    #4000,R1         ;;INCREMENT BY 1K
017710 062702 000040                ADD    #40,R2           ;;1K STEP
017714 005711                       TST    (R1)             ;;TRAP ON TIME OUT
017716 022701 177776                CMP    #177776,R1       ;;LAST ONE
017722 001370                       BNE    1$               ;;NO--TRY AGAIN
017724 162701 004000        $CROUT: SUB    #4000,R1
017730 162702 000040        $SIZEX: SUB    #40,R2           ;;DROP BACK
017734 010006                       MOV    R0,SP            ;;RESTORE THE STACK
017736 012637 000006                MOV    (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
017742 012637 000004                MOV    (SP)+,@#ERRVEC
017746 010137 017770                MOV    R1,$LSTAD        ;;LAST ADDRESS
017752 010237 017772                MOV    R2,$LSTBK        ;;LAST BANK
017756 012603                       MOV    (SP)+,R3         ;;RESTORE R3
017760 012602                       MOV    (SP)+,R2         ;;RESTORE R2
017762 012601                       MOV    (SP)+,R1         ;;RESTORE R1
017764 012600                       MOV    (SP)+,R0         ;;RESTORE R0
017766 000207                       RTS    PC
017770 000000               $LSTAD: .WORD  0                ;;CONTAINS THE LAST ADDRESS
017772 000000               $LSTBK: .WORD  0                ;;CONTAINS THE LAST BANK
```

```
4445                                            ;TYPDBO
4446                                            ;THIS ROUTINE CONVERTS A VIRTUAL ADDRESS TO PHYSICAL ADDRESS AND TYPES
4447                                            ;OUT THE 6 DIGIT PHYSICAL ADDRESS. R2 CONTAINS VIRTUAL ADDRESS AT THE TIME
4448                                            ;OF ENTRY.
4449                                            ;TYPE OUT IS INHIBITED IF SW 13 IS SET.
4450
4451  017774  032777  020000  161136  TYPDBO: BIT     #SW13,@SWR      ;INHIBIT TYPEOUT?
4452  020002  001042                          BNE     2$              ;YES
4453  020004  010346                          MOV     R3,-(SP)
4454  020006  010446                          MOV     R4,-(SP)
4455  020010  010546                          MOV     R5,-(SP)
4456
4457  020012  010246                          MOV     R2,-(SP)        ;PUSH VA ON STACK
4458  020014  004737  022170                  JSR     PC,CROTLF       ;ROTATE BITS 15,14,13 INTO 2,1,0
4459  020020  012603                          MOV     (SP)+,R3        ;FORM OFFSET TO BE USED
4460  020022  006303                          ASL     R3              ;FOR KIPAR
4461
4462  020024  016304  172340                  MOV     KIPAR0(R3),R4   ;GET THE BASE PAGE ADDRESS FROM
4463                                                                  ;KIPAR
4464  020030  005003                          CLR     R3              ;ROTATE LEFT 6 TIMES (MULTIPLY
4465  020032  012705  177772                  MOV     #-6,R5          ;BY 100 OCTAL) TO GET THE
4466  020036  006104          1$:             ROL     R4              ;BASE BUS ADDRESS (PHYSICAL)
4467  020040  005205                          INC     R5              ;R3 CONTAINS MSB-2 BITS
4468  020042  001375                          BNE     1$
4469
4470
4471  020044  010246                          MOV     R2,-(SP)        ;STRIP OFF TOP 3 BITS FROM  VA &
4472
4473  020046  042716  160000                  BIC     #160000,(SP)    ;GET THE OFFSET INSIDE THE PAGE
4474  020052  062604                          ADD     (SP)+,R4        ;FORM THE ENTIRE PHYSICAL
4475  020054  005503                          ADC     R3              ;ADDRESS. R4 CONTAINS LOWER 16 BITS
4476                                                                  ;R3 CONTAINS TOP 2 BITS
4477  020056  010437  001162                  MOV     R4,$REG0        ;SAVE LOWER 16 BITS OF PA
4478  020062  010337  001164                  MOV     R3,$REG1        ;SAVE TOP 2 BITS OF PA
4479  020066  012746  001162                  MOV     #$REG0,-(SP)    ;PUSH POINTER TO PA ON STACK
4480  020072  004737  024464                  JSR     PC,@#$DB2O      ;CONVERT THE 18 BIT BINARY
4481                                                                  ;ADDRESSS TO OCTAL ASICZ NUMBERS ON RETURN
4482                                                                  ;POINTER TO THE FIRST ASCIZ CHARACTERS
4483                                                                  ;IS ON STACK
4484  020076  004737  025014                  JSR     PC,@#SUPRS      ;TYPE OUT THE OCTAL 6 DIGIT
4485                                                                  ;PHYSICAL ADDRESS.
4486
4487  020102  012605                          MOV     (SP)+,R5
4488  020104  012604                          MOV     (SP)+,R4
4489  020106  012603                          MOV     (SP)+,R3
4490
4491  020110  000207          2$:             RTS     PC
```

```
                                        ;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.

020112  005777  161104        CHKCS:  TST     @RKCS           ;BIT 15 SET?
020116  100073                        BPL     COMRET          ;NO
020120  004737  022032                JSR     PC,GT4RG        ;YES, GET RKCS, ER, DS, DA
020124  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.

020126  020277  161076        CHKDA:  CMP     R2,@RKDA        ;DID RKDA INCREMENT CORRECTLY?
020132  001465                        BEQ     COMRET          ;YES
020134  010237  001162                MOV     R2,$REG0        ;GET EXPCTD RKDA
020140  017737  161064  001164        MOV     @RKDA,$REG1     ;GET RKDA RECVD
020146  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.

020150  000241                CHKBA:  CLC
020152  006103                        ROL     R3              ;FORM THE EXPCTD BUS ADDRESS
020154  060304                        ADD     R3,R4
020156  000241                        CLC
020160  006003                        ROR     R3
020162  020477  161040                CMP     R4,@RKBA        ;DID RKBA INCREMENT CORRECTLY?
020166  001447                        BEQ     COMRET          ;YES
020170  010437  001162                MOV     R4,$REG0        ;GET EXPCTD RKBA
020174  000207                        RTS     PC              ;RETURN TO THE EROR MESAGE

020176  017737  161024  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.

020204  017746  161012        CHKMEX: MOV     @RKCS,-(SP)     ;GET RKCS
020210  042716  177717                BIC     #177717,(SP)    ;GET MEX BITS 4,5
020214  022726  000020                CMP     #BIT4,(SP)+     ;CHECK BIT 4 SET?
020220  001432                        BEQ     COMRET          ;YES, OK
020222  004737  022032                JSR     PC,GT4RG        ;SAVE RKCS,ER,DS,DA
020226  000207                        RTS     PC              ;RETURN
```

```
                                        ;CHKWC
                                        ;THIS ROUTINE CHECKS IF RKWC OVERFLOWED CORRECILY 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.

020230  005777  160770          CHKWC:  TST     @RKWC           ;RKWC OVERFLOWED?
020234  001424                          BEQ     COMRET          ;YES
020236  017737  160766  001162          MOV     @RKDA,$REG0
020244  017737  160754  001164          MOV     @RKWC,$REG1
020252  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.

020254  032777  000100  160734  CHKRWS: BIT     #RWS,@RKDS      ;RWS RDY SET?
020262  001011                          BNE     COMRET          ;YES
020264  004737  022032                  JSR     PC,GT4RG        ;GET RKCS, ER, DS, DA
020270  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.

020272  105777  160724          CHKCRDY: TSTB   @RKCS           ;CONTROL READY SET?
020276  100403                          BMI     COMRET          ;YES
020300  004737  022032                  JSR     PC,GT4RG        ;GET RKCS, ER, DS, DA
020304  000207                          RTS     PC              ;RETURN TO THE EROR MESAGE

020306  062716  000002          COMRET: ADD     #2,(SP)         ;ADJUST RETURN ADDRESS TO SKIP OVER MESAGE
020312  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  HE 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.

020314  012737  100400  001512  DRCMND: MOV     #BIT15+BIT8,QFNC
020322  017746  160702                  MOV     @RKDA,-(SP)             ;SAVE DRIVE #
020326  004737  022170                  JSR     PC,CRDTLF
020332  052637  001512                  BIS     (SP)+,QFNC
020336  000424                          BR      P2

020340  012737  140000  001512  CRCMND: MOV     #BIT15+BIT14,QFNC
020346  000420                          BR      P2

020350  011037  001512          POSCMND:        MOV     (R0),QFNC
020354  042737  177770  001512                  BIC     #177770,QFNC            ;GET DRIVE NO.
020362  052737  100200  001512                  BIS     #BIT15+BIT7,QFNC
020370  000407                          BR      P2

020372  005037  001512          FNCMND: CLR     QFNC
020376  010046                  P1:     MOV     R0,-(SP)
020400  162716  001306                  SUB     #KEY,(SP)
020404  052637  001512                  BIS     (SP)+,QFNC

020410  013737  001462  001464  P2:     MOV     PRSFNC,PSTFNC
020416  013737  001512  001462          MOV     QFNC,PRSFNC
020424  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.

020426  010046                  HISTRY: MOV     R0,-(SP)
020430  010146                          MOV     R1,-(SP)

020432  012700  001462                  MOV     #PRSFNC,R0
020436  104401  020746                  TYPE    ,MH1
020442  104401  020772                  TYPE    ,MH3
020446  104401  020762                  TYPE    ,MH2
020452  005710                  6$:     TST     (R0)
020454  100053                          BPL     3$              ;READ, READ CHECK, WRITE, WRITE CHECK, SEEK
020456  105710                          TSTB    (R0)
020460  100427                          BMI     2$              ;POSITIONING (SEEK)
020462  032710  040000                  BIT     #BIT14,(R0)
020466  001014                          BNE     1$              ;CONTROL RESET

020470  104401  020476                  TYPE    ,65$            ;;TYPE ASCIZ STRING
020474  000410                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  /DRESET ON DRV /
020516                          64$:
020516  000425                          BR      7$

020520                          1$:
020520  104401  020526                  TYPE    ,67$            ;;TYPE ASCIZ STRING
020524  000404                          BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  /CRESET/
020536                          66$:
020536  000463                          BR      4$

020540                          2$:
020540  104401  020546                  TYPE    ,69$            ;;TYPE ASCIZ STRING
020544  000412                          BR      68$             ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  /POSITIONING DRIVE /
020572                          68$:
020572  011046                  7$:     MOV     (R0),-(SP)
020574  042716  177770                  BIC     #177770,(SP)    ;TYPE DRIVE NO.
020600  104402                          TYPOC
020602  000441                          BR      4$

020604  011001                  3$:     MOV     (R0),R1
020606  016101  002032                  MOV     PCMND(R1),R1    ;GO TYPE OUT THE FUNCTION
020612  016104  000002                  MOV     2(R1),R4        ;BEING PERFORMED
020616  004737  021736                  JSR     PC,TYPFN
020622  104401  020630                  TYPE    ,71$            ;;TYPE ASCIZ STRING
020626  000403                          BR      70$             ;;GET OVER THE ASCIZ
                                ;;71$:  .ASCIZ  <15><12>/DA=/
020636                          70$:
020636  011146                          MOV     (R1),-(SP)      ;TYPE OUT DISK ADDRESS
020640  104402                          TYPOC
020642  104401  020650                  TYPE    ,73$            ;;TYPE ASCIZ STRING
020646  000403                          BR      72$             ;;GET OVER THE ASCIZ
```

```
                                        ;;73$:  .ASCIZ  /  BA=/
                                        72$:
020656                                          MOV     6(R1),-(SP)
020656  016146  000006                          TYPOC
020662  104402                                  TYPE    ,75$            ;;TYPE ASCIZ STRING
020664  104401  020672                          BR      74$             ;;GET OVER THE ASCIZ
020670  000403                          ;;75$:  .ASCIZ  /  WC=/
                                        74$:
020700                                          MOV     4(R1),-(SP)
020700  016146  000004                          TYPOC
020704  104402
                                        4$:     CMP     R0,#PSTFNC
020706  020027  001464                          BEQ     5$
020712  001410                                  TST     (R0)+
020714  005720                                  TYPE    ,MH1
020716  104401  020746                          TYPE    ,MH4
020722  104401  020775                          TYPE    ,MH2
020726  104401  020752                          BR      6$
020732  000647
020734  104401  001213                  5$:     TYPE    ,$CRLF
020740  012601                                  MOV     (SP)+,R1
020742  012600                                  MOV     (SP)+,R0
020744  000207                                  RTS     PC
020746  005015  052506  041516          MH1:    .ASCIZ  <15><12>/FUNCTION /
020754  044524  047117  000040
020762  042440  051122  051117          MH2:    .ASCIZ  / ERROR /
020770  000040
020772  052101     000                  MH3:    .ASCIZ  /AT/
020775     120  044522  051117          MH4:    .ASCIZ  /PRIOR TO/
021002  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.

021006  010046                  STATSTC:MOV     R0,-(SP)        ;PUSH R0, R2, R3 ONTO THE
021010  010246                          MOV     R2,-(SP)        ;STACK
021012  010346                          MOV     R3,-(SP)

021014  005002                          CLR     R2
021016  005701                          TST     R1              ;DRIVE 0?
021020  001404                          BEQ     2$              ;FORM THE OFFSET FOR THE
021022  062702  000004          1$:     ADD     #4,R2           ;'WORDS XFERRED COUNTS'-
021026  005301                          DEC     R1              ;NWRTL, NRDL
021030  001374                          BNE     1$
021032  016500  000004          2$:     MOV     4(R5),R0        ;GET WORD COUNT (RKWC) FROM
021036  005400                          NEG     R0              ;THE PARAMETER TABLE

021040  005777  160156                  TST     @RKCS           ;ANY ERROR DURING THE XFER?
021044  100004                          BPL     3$
                                                                ;YES,
021046  017703  160152                  MOV     @RKWC,R3        ;GET THE # OF WORDS THAT
021052  005403                          NEG     R3              ;WERE ACTUALLY X-FERRED
021054  160300                          SUB     R3,R0

021056  022704  000002          3$:     CMP     #2,R4           ;WRITE FUNCTION?
021062  001005                          BNE     5$
                                                                ;YES, ADD THE # OF WORDS
021064  060062  001732          4$:     ADD     R0,NWRTL(R2)    ;XFERRED (WRITE)
021070  005562  001734                  ADC     NWRTH(R2)       ;NOTE IT'S 2-WORD COUNT LO, HI
021074  000404                          BR      6$

021076  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'
021102  005562  001774                  ADC     NRDH(R2)        ;CARRY OVER TO THE HI WORD

021106  012603                  6$:     MOV     (SP)+,R3        ;POP R3,R2,R0 FROM THE STACK
021110  012602                          MOV     (SP)+,R2
021112  012600                          MOV     (SP)+,R0
021114  000207                          RTS     PC
```

```
4757                                    ;REPSTAT
4758                                    ;THIS ROUTINE REPORTS ERROR STATISTICS AND DATA-TRANSFER STATISTICS.
4759
4760  021116  104401  002377  REPSTA: TYPE    ,MSG26
4761  021122  013700  001264          MOV     @DRVPRS,R0
4762  021126  012701  001254          MOV     #PDR,R1
4763
4764  021132  104401  001213  1$:     TYPE    ,$CRLF
4765  021136  112102                  MOVB    (R1)+,R2
4766  021140  042702  177770          BIC     #177770,R2
4767  021144  010246                  MOV     R2,-(SP)
4768  021146  104403                  TYPOS
4769  021150     003                  .BYTE   3
4770  021151     000                  .BYTE   0
4771  021152  104401  002662          TYPE    ,BLNKS3
4772
4773  021156  005004                  CLR     R4
4774  021160  010203                  MOV     R2,R3
4775  021162  001404                  BEQ     3$
4776  021164  062704  000004  2$:     ADD     #4,R4
4777  021170  005303                  DEC     R3
4778  021172  001374                  BNE     2$
4779
4780  021174  104401  002664  3$:     TYPE    ,BLNKS1
4781  021200  010446                  MOV     R4,-(SP)
4782  021202  062716  001732          ADD     #NWRTL,(SP)
4783  021206  004737  024604          JSR     PC,$DB2D
4784  021212  004737  025014          JSR     PC,SUPRS
4785  021216  104401  002664          TYPE    ,BLNKS1
4786  021222  010446                  MOV     R4,-(SP)
4787  021224  062716  001772          ADD     #NRDL,(SP)
4788  021230  004737  024604          JSR     PC,$DB2D
4789  021234  004737  025014          JSR     PC,SUPRS
4790
4791  021240  006302                  ASL     R2
4792
4793  021242  104401  002664          TYPE    ,BLNKS1
4794  021246  016246  001652          MOV     CSECN(R2),-(SP)
4795  021252  104405                  TYPDS
4796
4797  021254  104401  002664          TYPE    ,BLNKS1
4798  021260  016246  001632          MOV     WCECN(R2),-(SP)
4799  021264  104405                  TYPDS
4800
4801  021266  104401  002664          TYPE    ,BLNKS1
4802  021272  016246  001712          MOV     DATER(R2),-(SP)
4803  021276  042716  100000          BIC     #100000,(SP)    ;DONT TYPE A NEGATIVE NO.
4804  021302  104405                  TYPDS
4805
4806  021304  104401  002664          TYPE    ,BLNKS1
4807  021310  016246  001562          MOV     HECN(R2),-(SP)
4808  021314  104405                  TYPDS
4809
4810  021316  005300                  DEC     R0              ;FINISHED WITH THE DRIVES ?
4811  021320  001304                  BNE     1$              ;BR IF NOT
4812  021322  104401  002474          TYPE    ,MSG26A         ;REST OF SUMMARY MESSAGE
```

```
4813  021326  013700  001264          MOV     DRVPRS,R0       ;NUMBER OF DRIVES
4814  021332  012701  001254          MOV     #PDR,R1         ;'DRIVES PRESENT' TABLE ADDRESS
4815  021336  104401  001213  4$:     TYPE    ,$CRLF          ;CR-LF
4816  021342  112102                  MOVB    (R1)+,R2        ;DRIVE ADDRESS
4817  021344  042702  177770          BIC     #177770,R2      ;LEAVE ONLY DRIVE NUMBER
4818  021350  010246                  MOV     R2,-(SP)        ;PUT ON STACK FOR TYPEOUT
4819  021352  104403                  TYPOS                   ;TYPE IT IN OCTAL
4820  021354     003                  .BYTE   3               ;TYPE 3 CHARACTERS
4821  021355     000                  .BYTE   0               ;SUPRESS LEADING ZEROS
4822  021356  104401  002662          TYPE    ,BLNKS3         ;3 BLANKS
4823  021362  006302                  ASL     R2              ;CONVERT TO A WORD TABLE INDEX
4824  021364  104401  002664          TYPE    ,BLNKS1
4825  021370  016246  001602          MOV     SKECN(R2),-(SP)
4826  021374  104405                  TYPDS
4827
4828
4829  021376  104401  002664          TYPE    ,BLNKS1
4830  021402  016246  001672          MOV     ABORT(R2),-(SP)
4831  021406  104405                  TYPDS
4832
4833  021410  006202                  ASR     R2
4834  021412  104401  002664          TYPE    ,BLNKS1
4835  021416  116246  001622          MOVB    SINCN(R2),-(SP)
4836  021422  104405                  TYPDS
4837
4838  021424  005300                  DEC     R0
4839  021426  001343                  BNE     4$
4840  021430  004737  026660          JSR     PC,TIMTYP       ;TYPE THE TIME
4841  021434  000207                  RTS     PC
```

```
 4842                                   ;STATUS
 4843                                   ;THIS ROUTINE IS NORMALLY ENTERED WHEN THE PROGRAM IS WAITING FOR THE
 4844                                   ;CONTROLLER TO FINISH WHAT IT IS DOING. THERE ARE TWO DOUBLE PRECISION
 4845                                   ;COUNTS KEPT IN THIS ROUTINE.
 4846                                   ;CICNT,CICNT1
 4847                                   ;THIS COUNT KEEPS TRACK OF HOW LONG THE CONTROLLER HAS BEEN BUSY AFTER
 4848                                   ;A COMMAND WAS INITIATED. THE CONTROLLER SHOULD FINISH WHATEVER IT IS DOING
 4849                                   ;BEFORE THIS COUNT EXPIRES. IF IT DOES NOT, THERE IS AN ERROR CONDITION ANC
 4850                                   ;IT IS SO REPORTED.
 4851
 4852                                   ;QSCNT
 4853                                   ;THIS COUNT IS INITIALIZED EVERY TIME 8 COMMANDS ARE GENERATED. THE COUNT
 4854                                   ;IS INCREMENTED EVERY TIME THIS ROUTINE IS ENTERED. ALL THE 8 COMMANDS
 4855                                   ;SHOULD BE DONE BEFORE THIS COUNT EXPIRES. IF THEY DO NOT AN ERROR CONDITION
 4856                                   ;IS REPORTED. THIS COUNT HAS BEEN KEPT PRIMARILY TO INSURE THAT THE PROGRAM
 4857                                   ;DOES NOT GET CAUGHT IN AN INDEFINITE LOOP, BECAUSE OF AN ERROR CONDITION.
 4858
 4859  021436  005046          STATUS: CLR     -(SP)           ;DROP PRIORITY AND WAIT FOR INT
 4860  021440  012746  021446          MOV     #1$,-(SP)       ;RETURN FOR RTI
 4861  021444  000002                  RTI
 4862                                   ;NOTE THAT THE INTERRUPTS ARE ALLOWED ONLY
 4863                                   ;AT CERTAIN PLACES IN THE PROGRAM, BECAUSE
 4864                                   ;IT MAKES TROUBLESHOOTING OF FALIURES EASY.
 4865                                   ;OTHER PLACES WHERE INTERRUPTS ARE ALLOWED
 4866                                   ;TO TAKE PLACE:
 4867                                   ;'CHFAFN',  FIRST INTERRUPT AFTER ISSUING
 4868                                   ;A SEEK FUNCTION.
 4869
 4870  021446  005237  001460  1$:     INC     QSCNT
 4871  021452  001456                  BEQ     QERROR
 4872  021454  105777  157542          TSTB    @RKCS
 4873  021460  100514                  BMI     CNOBSY
 4874
 4875  021462  005037  001466          CLR     CICNT
 4876  021466  012737  177761  001470  MOV     #-17,CICNT1
 4877
 4878  021474  105737  001534  CBSY:   TSTB    INTFLG          ;FOR A NON-SEEK COMAND:
 4879                                   ;INTFLG- BIT 7 IS SET, BITS 0-3 CONTAIN
 4880                                   ;OFFSET TO THE COMMAND KEY (FROM KEY),
 4881                                   ;FOR WHICH THIS INTERUPT IS EXPCTD.
 4882                                   ;WHEN THE INTERUPT OCCURS & 'INTHND' IS
 4883                                   ;ENTERED 'INTFLG' IS CLEARED.
 4884  021500  001507                  BEQ     SEXIT
 4885  021502  005237  001466          INC     CICNT
 4886  021506  001372                  BNE     CBSY
 4887  021510  005237  001470          INC     CICNT1
 4888  021514  001367                  BNE     CBSY
 4889
 4890                                   ;TIMED OUT WHILE WAITING FOR THE INTRUPT.
 4891                                   ;ONE OF THE COMMANDS DID NOT INTERRUPT
 4892  021516  113700  001534  NIEROR: MOVB    INTFLG,R0
 4893  021522  042700  177760          BIC     #177760,R0
 4894  021526  010003                  MOV     R0,R3
 4895  021530  062700  001306          ADD     #KEY,R0
 4896  021534  011037  0C1172          MOV     (R0),$REG4
 4897  021540  042737  177770  001172  BIC     #177770,$REG4
```

```
 4898  021546  013737  001172  001250          MOV     $REG4,SRDRV     ;GET DRIVE #, FOR TYPING SERIAL #
 4899
 4900  021554  104421  002245                  TYPMSG  ,MSG15          ;PRINT 'DRVE # DIDN'T INTRUPT AFTER'
 4901  021560  016305  002032                  MOV     PCMND(R3),R5
 4902  021564  016504  000002                  MOV     2(R5),R4
 4903  021570  004737  021736                  JSR     PC,TYPFN
 4904  021574  004737  022032                  JSR     PC,GT4RG
 4905  021600  104025                          ERROR   25              ;COMMAND TYPED OUT IN EROR MESAGE DID
 4906                                                                  ;NOT INTERUPT ON COMPLETION.
 4907  021602  052710  104000                  BIS     #BIT15+BIT11,(R0)       ;INDICATE THAT FUNCTION IS ABORTED
 4908  021606  000444                          BR      SEXIT
 4909
 4910
 4911  021610  005037  001460          QEROR:  CLR     QSCNT           ;REESTABLISH COUNT
 4912  021614  004737  022032                  JSR     PC,GT4RG
 4913  021620  104026                          ERROR   26              ;ALL 8 COMMANDS SHOULD BE DONE BY NOW, TIMED
 4914                                                                  ;OUT. THE PROGRAM IS WAITING FOR ONE OF THE
 4915                                                                  ;COMMANDS IN THE Q TO BE FINISHED AND THIS
 4916                                                                  ;DID NOT HAPPEN OR FOR SOME OTHER REASON THE
 4917                                                                  ;'FINISHED' FLAG (BIT 15) OF ONE OF THE 8
 4918                                                                  ;COMMAND KEYS WAS NOT SET. VARIOUS FLAGS 'POS'(-7
 4919                                                                  ;'BUSY'(-7), 'KEY'(-8) CONTAIN INFORMATION
 4920                                                                  ;ABOUT THE STATUS OF THE SYSTEM.
 4921
 4922  021622  032777  020000  157310          BIT     #SW13,@SWR      ;INHIBIT TYPEOUT?
 4923  021630  001024                          BNE     2$              ;YES
 4924  021632  104401  002305                  TYPE,   MSG16
 4925  021636  012700  001306                  MOV     #KEY,R0
 4926  021642  012701  001426                  MOV     #BUSY,R1
 4927  021646  012702  177770                  MOV     #-10.R2
 4928  021652  104401  001213          1$:     TYPE    ,$CRLF
 4929  021656  012046                          MOV     (R0)+,-(SP)     ;TYPE OUT CONTENTS OF ALL KEYS
 4930  021660  104402                          TYPOC                   ;KEY-KEY8
 4931  021662  104401  002662                  TYPE    ,BLNKS3
 4932  021666  005046                          CLR     -(SP)
 4933  021670  112116                          MOVB    (R1)+,(SP)      ;TYPE OUT CONTENTS OF ALL BUSY FLAGS
 4934  021672  104403                          TYPOS                   ;BUSY-BUSY7
 4935  021674     003                          .BYTE   3
 4936  021675     000                          .BYTE   0
 4937  021676  005202                          INC     R2              ;DONE?
 4938  021700  001364                          BNE     1$              ;NO
 4939
 4940  021702  004737  016006          2$:     JSR     PC,CLRERR       ;MAKE SURE THERE IS NO HEAD MOVEMENT ON
 4941                                                                  ;ANY DRIVE & THEN DO CONTROL RESET
 4942  021706  000137  010630                  JMP     BEGNEX          ;GO, BAK AND CONTINUE
 4943
 4944  021712  005004                  CNOBSY: CLR     R4
 4945  021714  005204                          INC     R4
 4946  021716  001376                          BNE     .-2
 4947  021720  013746  001244          SEXIT:  MOV     PPRLVL,-(SP)
 4948  021724  012746  021732                  MOV     #RTIPC7,-(SP)   ;RETURN FOR RTI **********
 4949  021730  000002                          RTI
 4950
 4951  021732  000137  010650          RTIPC7: JMP     QMNGER
```

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

```
                                        ;GT4RG
                                        ;GET CONTENTS OF RKCS, RKER, RKDS, RKDAA
                                        
022032  017737  157172  001170  GT4RG:  MOV     @RKDA,$REG3
022040  017737  157156  001162  GT3RG:  MOV     @RKCS,$REG0
022046  017737  157146  001164          MOV     @RKER,$REG1
022054  017737  157136  001166          MOV     @RKDS,$REG2
022062  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.
                                        
022064  004737  022040          GETINF: JSR     PC,GT3RG
022070  010046                          MOV     R0,-(SP)
022072  010146                          MOV     R1,-(SP)
022074  010246                          MOV     R2,-(SP)
022076  012700  001200                  MOV     #$REG6+2,R0
022102  017701  157122                  MOV     @RKDA,R1
022106  010102                          MOV     R1,R2
022110  042702  177760                  BIC     #177760,R2
022114  010240                          MOV     R2,-(R0)
022116  006201                          ASR     R1
022120  006201                          ASR     R1
022122  006201                          ASR     R1
022124  006201                          ASR     R1
022126  010102                          MOV     R1,R2
022130  042702  177776                  BIC     #177776,R2
022134  010240                          MOV     R2,-(R0)
022136  006201                          ASR     R1
022140  010102                          MOV     R1,R2
022142  042702  177400                  BIC     #177400,R2
022146  010240                          MOV     R2,-(R0)
022150  000301                          SWAB    R1
022152  042701  177770                  BIC     #177770,R1
022156  010140                          MOV     R1,-(R0)
022160  012602                          MOV     (SP)+,R2
022162  012601                          MOV     (SP)+,R1
022164  012600                          MOV     (SP)+,R0
022166  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.


022170  042766  017777  000002  CROTLF: BIC     #17777,2(SP)
022176  000241                          CLC
022200  006166  000002                  ROL     2(SP)
022204  006166  000002                  ROL     2(SP)
022210  006166  000002                  ROL     2(SP)
022214  006166  000002                  ROL     2(SP)
022220  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'.
022222  004737  022032          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)

022226  017746  156776          GTSDRV: MOV     @RKDA,-(SP)     ;GET BITS 15,14,13 FROM RKDA
022232  004737  022170                  JSR     PC,CROTLF
022236  012637  001250                  MOV     (SP)+,SRDRV     ;SAVE THE DRIVE #
022242  000207                          RTS     PC
```

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

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

022356  012777  000001  156636  CN.RST: MOV     #1,@RKCS
022364  005037  001472          CN.RDY: CLR     TIMER
022370  105777  '56626          1$:     TSTB    @RKCS           ;DID CONTROL RDY SET?
022374  100451                          BMI     2$              ;YES
022376  012746  177750                  MOV     #-30,-(SP)      ;WAIT FOR CNTRL RDY
022402  005216                          INC     (SP)
022404  001376                          BNE     .-2
022406  005726                          TST     (SP)+
022410  005237  001472                  INC     TIMER
022414  001365                          BNE     1$
022416  032777  020000  156514          BIT     #SW13,@SWR      ;INHIBIT TYPEOUT?
022424  001035                          BNE     2$              ;YES
022426  104401  002206                  TYPE    ,MSG12          ;CNTRL RDY DID NOT SET, REPORT ERROR
022432  011646                          MOV     (SP),-(SP)
022434  162716  000002                  SUB     #2,(SP)
022440  104402                          TYPOC
022442  104401  022450                  TYPE    ,65$            ;;TYPE ASCIZ STRING
022446  000421                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/CONTROLLER NOT READY -   RKCS=/
022512                          64$:
022512  017746  156504                  MOV     @RKCS,-(SP)
022516  104402                          TYPOC
022520  000002                  2$:     RTI
```

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

```
                        .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)

022552 005237 001560    KWSRVE: INC     KWCOUNT         ;COUNT 60 CPS
022556 001401                   BEQ     1$              ;OVERFLOWED?
022560 000002                   RTI
022562 012737 177704 001560 1$: MOV     #-60.,KWCOUNT   ;RESET 60 CPS COUNT
022570 005237 001556            INC     KWSEC           ;COUNT SECONDS
022574 001401                   BEQ     2$              ;OVERFLOWED?
022576 000002                   RTI                     ;RETURN
022600 012737 177704 001556 2$: MOV     #-60.,KWSEC     ;RESET "SECONDS" COUNT
022606 005237 001554            INC     KWMIN           ;COUNT MINUTES
022612 001005                   BNE     3$              ;OVERFLOWED?
022614 012737 177704 001554     MOV     #-60.,KWMIN     ;RESET "MINUTES" COUNT
022622 005237 001552            INC     KWHR            ;COUNT HOURS

022626 000002           3$:     RTI                     ;RETURN


                        ;WATIME
                        ;ROUTINE PROVIDES SOME WAITING TIME.

022630 013746 022652    WATIME: MOV     2$,-(SP)        ;COUNTER VALUE
022634 005237 022654    1$:     INC     3$              ;COUNT
022640 001375                   BNE     1$              ;HANG IN THERE UNTIL COUNT WRAPS AROUND
022642 005216                   INC     (SP)            ;COUNT AGAIN
022644 001373                   BNE     1$              ;GO THROUGH MINOR LOOP AGAIN
022646 005726                   TST     (SP)+           ;RESTORE THE STACK POINTER
022650 000207                   RTS     PC              ;RETURN
022652 177730           2$:     .WORD   177730          ;VALUE FOR APPROX 15 SEC DELAY
022654 000000           3$:     .WORD   0               ;'MINOR' LOOP COUNTER
```

```
5156                                    ;CHPDRS
5157                                    ;THIS ROUTINE CHECKS IF THERE ANY DRIVES PRESENT (ON LINE), IF THERE
5158                                    ;ARE, A RETURN IS MADE. IF THERE ARE NONE PRESENT, A MESSAGE IS PRINTED OUT.
5159                                    ;THE STACK POINTER IS RE-INITIATED TO 1100 AND CONTROL IS TRANSFERRED
5160                                    ;TO THE END OF PASS ROUTINE, $EOP. BEFORE PASSING CONTROL TO $EOP, SOME
5161                                    ;TIME IS KILLED (WATIME), THIS IS DONE TO KEEP THE NUMBER OF MESSAGES
5162                                    ;(END OF PASS #X) TO A SMALL AMOUNT.
5163
5164  022656  005737  001264  CHDPRS: TST     DRVPRS          ;ANY DRIVES PRESENT?
5165  022662  001401                  BEQ     1$              ;NO
5166  022664  000207                  RTS     PC              ;YES, EXIT
5167  022666  104401  002225  1$:     TYPE    ,MSG14          ;NO, GIVE A MESSAGE
5168  022672  004737  022630          JSR     PC,WATIME       ;KILL SOME TIME
5169  022676  012706  001100          MOV     #STACK,SP       ;REINITIALIZE STACK
5170  022702  000400                  BR      $EOP            ;GO TO END OF PASS ROUTINE
5171
```

```
                        .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 BEGNEX

022704                  $EOP:
022704  000004                  SCOPE
022706  005037  001102          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
022712  005237  001100          INC     $PASS           ;;INCREMENT THE PASS NUMBER
022716  042737  100000  001100  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
022724  005327                  DEC     (PC)+           ;;LOOP?
022726  000001          $EOPCT: .WORD   1
022730  003013                  BGT     $DOAGN          ;;YES
022732  012737                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
022734  000001          $ENDCT: .WORD   1
022736  022726                  $EOPCT
022740  013700  000042  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
022744  001405                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
022746  000005                  RESET                   ;;CLEAR THE WORLD
022750  004710          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
022752  000240                  NOP                     ;;SAVE ROOM
022754  000240                  NOP                     ;;FOR
022756  000240                  NOP                     ;;ACT11
022760                  $DOAGN:
022760  000137                  JMP     @(PC)+          ;;RETURN
022762  010630          $RTNAD: .WORD   BEGNEX
```

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

```
5257  023176  002420                        BLT     18$             ;;BRANCH IF YES
5258  023200  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
5259  023204  003015                        BGT     18$             ;;BRANCH IF YES
5260  023206  042726  000060                BIC     #60,(SP)+       ;;STRIP-OFF ASCII
5261  023212  005766  000002                TST     2(SP)           ;;IS THIS THE FIRST CHAR
5262  023216  001403                        BEQ     17$             ;;BRANCH IF YES
5263  023220  006316                        ASL     (SP)            ;;NO, SHIFT PRESENT
5264  023222  006316                        ASL     (SP)            ;;   CHAR OVER TO MAKE
5265  023224  006316                        ASL     (SP)            ;;   ROOM FOR NEW ONE.
5266  023226  005266  000002        17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
5267  023232  056616  177776                BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
5268  023236  000707                        BR      7$              ;;GET THE NEXT ONE
5269  023240  1044C1  001212        18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
5270  023244  000720                        BR      20$             ;;SIMULATE CONTROL-U
5271                                .DSABL  LSB
5272
5273
5274                                ;;**********************************************************************
5275                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
5276                                ;*CALL:
5277                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
5278                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
5279                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
5280                                ;
5281
5282  023246  011646                $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
5283  023250  016666  000004  000002        MOV     4(SP),2(SP)     ;;SAVE THE PS
5284  023256  105777  155662        1$:     TSTB    @$TKS           ;;WAIT FOR
5285  023262  100375                        BPL     1$              ;;A CHARACTER
5286  023264  117766  155656  000004        MOVB    @$TKB,4(SP)     ;;READ THE TTY
5287  023272  042766  177600  000004        BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
5288  023300  026627  000004  000023        CMP     4(SP),#23       ;;IS IT A CONTROL-S^
5289  023306  001013                        BNE     3$              ;;BRANCH IF NO
5290  023310  105777  155630        2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
5291  023314  100375                        BPL     2$              ;;LOOP UNTIL ITS THERE
5292  023316  117746  155624                MOVB    @$TKB,-(SP)     ;;GET CHARACTER
5293  023322  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
5294  023326  022627  000021                CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
5295  023332  001366                        BNE     2$              ;;IF NOT DISCARD IT
5296  023334  000750                        BR      1$              ;;YES, RESUME
5297  023336  026627  000004  000140  3$:   CMP     4(SP),#140      ;;IS IT UPPER CASE?
5298  023344  002407                        BLT     4$              ;;BRANCH IF YES
5299  023346  026627  000004  000175        CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
5300  023354  003003                        BGT     4$              ;;BRANCH IF YES
5301  023356  042766  000040  000004        BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
5302  023364  000002                4$:     RTI                     ;;GO BACK TO USER
5303                                ;;**********************************************************************
5304                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
5305                                ;*CALL:
5306                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
5307                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
5308                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
5309
5310  023366  010346                $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
5311  023370  012703  023474        1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
5312  023374  022703  023504        2$:     CMP     #$TTYIN+80.,R3  ;;BUFFER FULL?
```

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

```
                                .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

023540  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
023542  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
023550  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
023552  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
023554  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
023556  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
023560  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
023562  005001                          CLR     R1              ;;CLEAR DATA WORD
023564  005002                          CLR     R2
023566  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
023570  001412                          BEQ     3$              ;;IF ZERO GET OUT
023572  006301                          ASL     R1              ;;*2
023574  006102                          ROL     R2
023576  006301                          ASL     R1              ;;*4
023600  006102                          ROL     R2
023602  006301                          ASL     R1              ;;*8
023604  006102                          ROL     R2
023606  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
023612  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
023614  000764                          BR      2$              ;;LOOP
023616  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
023620  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
023624  010237  023640                  MOV     R2,$HIOCT
023630  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
023632  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
023634  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
023636  000002                          RTI                     ;;RETURN
023640  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
                                ;
023642  011646                  $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
023644  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
023652  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
023654  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
023656  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
023660  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
023662  012600                          MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
023664  010037  024010                  MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
023670  005046                          CLR     -(SP)           ;;CLEAR DATA WORD
023672  005002                          CLR     R2              ;;SIGN SET POSITIVE
023674  122710  000055                  CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
023700  001001                          BNE     2$              ;;BR IF NO MINUS SIGN
023702  112002                          MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
023704  112001                  2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
023706  001424                          BEQ     3$              ;;GET OUT IF ZERO
023710  122701  000060                  CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
023714  003032                          BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
023716  122701  000071                  CMPB    #'9,R1
023722  002427                          BLT     5$
023724  032716  170000                  BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
023730  001024                          BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
023732  006316                          ASL     (SP)            ;;*2
023734  011646                          MOV     (SP),-(SP)      ;;SAVE FOR LATER
023736  006316                          ASL     (SP)            ;;*4
023740  006316                          ASL     (SP)            ;;*8.
023742  062616                          ADD     (SP)+,(SP)      ;;*10.
023744  102416                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
023746  162701  000060                  SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
023752  060116                          ADD     R1,(SP)         ;;ADD IN THIS DIGIT
023754  102412                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
023756  000752                          BR      2$              ;;LOOP
023760  005702                  3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
023762  001401                          BEQ     4$              ;;BR IF NO
023764  005416                          NEG     (SP)            ;;YES--NEGATE THE NUMBER
023766  012666  000012          4$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
023772  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
023774  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
023776  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
024000  000002                          RTI                     ;;RETURN
                                ;
024002  005726                  5$:     TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
024004  105010                          CLRB    (R0)            ;;SET A TERMINATOR
```

```
5434  024006  104401                    TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
5435  024010  000000                    .WORD   0               ;;POINTER GOES HERE
5436  024012  104401  001212    6$:     TYPE    .$QUES          ;;"?" CR & LF
5437  024016  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

024020                          $TYPDS:
024020  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
024022  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
024024  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
024026  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
024030  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
024032  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
024036  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
024042  100004                          BPL     1$              ;;BR IF INPUT IS POS.
024044  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
024046  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
024054  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
024056  012703  024234                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
024062  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
024066  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
024070  016001  024224                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
024074  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
024076  002402                          BLT     4$              ;;BR IF DONE
024100  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
024102  000774                          BR      3$
024104  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
024106  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
024110  001002                          BNE     5$              ;;FALL THROUGH IF 0
024112  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
024114  100407                          BMI     7$              ;;BR IF YES
024116  106316                  5$:     ASLB    (SP)            ;;MSD?
024120  103003                          BCC     6$              ;;BR IF NO
024122  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
024130  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
024134  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
024140  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
024142  005720                          TST     (R0)+           ;;JUST INCREMENTING
024144  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
024150  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
024152  003002                          BGT     8$              ;;GO TO EXIT
024154  010502                          MOV     R5,R2           ;;GET THE LSD
024156  000764                          BR      6$              ;;GO CHANGE TO ASCII
024160  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
024162  100003                          BPL     9$              ;;BR IF NO
024164  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
024172  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
024174  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
024176  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
024200  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
```

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

```
                                        .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
                                        ;*

024244  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
024250  100002                          BPL     1$              ;;BR IF YES
024252  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
024254  000407                          BR      3$              ;;LEAVE
024256  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
024260  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
024264  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
024266  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
024270  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
024272  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
024274  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
024300  000002                          RTI                     ;;RETURN
024302  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
024306  001430                          BEQ     8$
024310  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
024314  001006                          BNE     5$
024316  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
024320  104401                          TYPE                    ;;TYPE A CR AND LF
024322  001213                          $CRLF
024324  105037  024460                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
024330  000755                          BR      2$              ;;GET NEXT CHARACTER
024332  004737  024414          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
024336  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
024342  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
024344  013746  001.54                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
024350  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
024354  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
024356  004737  024414                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
024362  105337  024460                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
024366  000770                          BR      7$              ;;LOOP

                                        ;HORIZONTAL TAB PROCESSOR

024370  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
024374  004737  024414          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
024400  132737  000007  024460          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
024406  001372                          BNE     9$              ;;TAB STOP
024410  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
```

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

```
                                .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,@#$DB2O      ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK

024464  104414                  $DB2O:  SAVREG                  ;;SAVE ALL REGISTERS
024466  016601  000002                  MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
024472  012705  024603                  MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
024476  012704  000014                  MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
024502  012703  177770                  MOV     #^C7,R3         ;;MASK
024506  012100                          MOV     (R1)+,R0        ;;LOWER WORD
024510  012101                          MOV     (R1)+,R1        ;;HIGH WORD
024512  005002                          CLR     R2              ;;TERMINATOR
024514  110245                  1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
024516  010002                          MOV     R0,R2           ;;GET THIS DIGIT
024520  005304                          DEC     R4              ;;COUNT THIS CHARACTER
024522  003007                          BGT     3$              ;;BR IF NOT THE LAST DIGIT
024524  001405                          BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
024526  005205                          INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
024530  010566  000002                  MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
024534  104415                          RESREG                  ;;RESTORE ALL REGISTERS
024536  000207                          RTS     PC              ;;RETURN TO USER
024540  006203                  2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
024542  006001                  3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
024544  006000                          ROR     R0              ;;      THE NEXT OCTAL DIGIT
024546  006001                          ROR     R1
024550  006000                          ROR     R0
024552  006001                          ROR     R1
024554  006000                          ROR     R0
024556  040302                          BIC     R3,R2           ;;MASK OUT ALL JUNK
024560  062702  000060                  ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
024564  000753                          BR      1$              ;;GO PUT IT IN THE DATA TABLE
024566  000016                  $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
```

```
5614                                    .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
5615
5616                                    ;;**********************************************************************
5617                                    ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
5618                                    ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
5619                                    ;*POSITIVE.
5620                                    ;*CALL
5621                                    ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
5622                                    ;*      JSR     PC,@#$DB2D
5623                                    ;*      RETURN                  ;;THE FIRST ADDRESS OF ASCIZ
5624                                                                    ;;IS ON THE STACK
5625
5626
5627  024604  104414                    $DB2D:  SAVREG                  ;;SAVE REGISTERS
5628  024606  016602  000002                   MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
5629  024612  012700  024764                   MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
5630  024616  010066  000002                   MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
5631  024622  012201                           MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
5632  024624  012202                           MOV     (R2)+,R2
5633  024626  012737  000012  024702           MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
5634  024634  012704  024714                   MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
5635  024640  012705  024716                   MOV     #$TNPWR+2,R5
5636  024644  005003                    1$:     CLR     R3              ;;CLEAR PARTIAL
5637  024646  161401                    2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
5638  024650  005602                            SBC     R2
5639  024652  161502                            SUB     (R5),R2
5640  024654  002402                            BLT     3$              ;;BR IF TEN POWER TO LARGE
5641  024656  005203                            INC     R3              ;;ADD 1 TO PARTIAL
5642  024660  000772                            BR      2$              ;;LOOP
5643  024662  062401                    3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
5644  024664  005502                            ADC     R2
5645  024666  062402                            ADD     (R4)+,R2
5646  024670  022525                            CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
5647  024672  052703  000060                   BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
5648  024676  110320                            MOVB    R3,(R0)+        ;;SAVE IT
5649  024700  005327                            DEC     (PC)+           ;;DONE?
5650  024702  000000                    4$:     .WORD   0
5651  024704  001357                            BNE     1$              ;;BR IF NO
5652  024706  105020                            CLRB    (R0)+           ;;TERMINATOR
5653  024710  104415                            RESREG                  ;;RESTORE REGISTERS
5654  024712  000207                            RTS     PC              ;;RETURN
5655  024714  145000                    $TNPWR: 145000                  ;;1.0E09
5656  024716  035632                            35632
5657  024720  160400                            160400                  ;;1.0E08
5658  024722  002765                            2765
5659  024724  113200                            113200                  ;;1.0E07
5660  024726  000230                            230
5661  024730  041100                            041100                  ;;1.0E06
5662  024732  000017                            17
5663  024734  103240                            103240                  ;;1.0E05
5664  024736  000001                            1
5665  024740  023420                            23420                   ;;1.0E04
5666  024742  000000                            0
5667  024744  001750                            1750                    ;;1.0E03
5668  024746  000000                            0
5669  024750  000144                            144                     ;;1.0E02
```

```
5670  024752  000000                    0
5671  024754  000012                    12                  ;;1.0E01
5672  024756  000000                    0
5673  024760  000001                    1                   ;;1.0E00
5674  024762  000000                    0
5675  024764  000014  $DECVL:  .BLKB   12.                  ;;RESERVE STORAGE FOR ASCIZ STRING
```

```
5676                                    .SBTTL  SUPRS - TYPE NUMERICAL ASCIZ STRING, REPLACE LEADING 0'S BY BLANKS
5677                                    .SBTTL  SUPRSL - TYPE NUMERICAL ASCIZ STRING, LEFT JUSTIFY
5678                                    ;NOT FROM SYSMAC
5679
5680  025000  010046            SUPRSL: MOV     R0,-(SP)        ;SAVE R0
5681  025002  005037  025070            CLR     SUP2
5682  025006  016600  000004            MOV     4(SP),R0
5683  025012  000405                    BR      SUP1
5684
5685  025014  010046            SUPRS:  MOV     R0,-(SP)        ;SAVE R0
5686  025016  016600  000004            MOV     4(SP),R0        ;PICKUP THE POINTER
5687  025022  010037  025070            MOV     R0,SUP2         ;SAVE FOR TYPING
5688  025026                    SUP1:
5689  025026  105710            1$:     TSTB    (R0)            ;TERMINATOR?
5690  025030  001406                    BEQ     2$              ;BR IF YES
5691  025032  122710  000060            CMPB    #'0,(R0)        ;IS THIS AN ASCII "0"?
5692  025036  001006                    BNE     4$              ;NO
5693  025040  112720  000040            MOVB    #40,(R0)+       ;REPLACE IT WITH "BLANK"
5694  025044  000770                    BR      1$
5695  025046  005300            2$:     DEC     R0              ;BACKUP BY 1
5696  025050  112710  000060            MOVB    #'0,(R0)        ;ASCII "0"
5697  025054  005737  025070    4$:     TST     SUP2            ;LEFT JUSTIFY?
5698  025060  001002                    BNE     5$              ;NO
5699  025062  010037  025070            MOV     R0,SUP2         ;YES
5700  025066  104401            5$:     TYPE                    ;GO TYPE
5701  025070  000000            SUP2:   .WORD   0
5702  025072  012600                    MOV     (SP)+,R0        ;RESTORE R0
5703  025074  012616                    MOV     (SP)+,(SP)      ;RESTORE THE STACK
5704  025076  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

025100                  $MULT:
025100  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
025102  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
025104  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
025106  005046                  CLR     -(SP)           ;;CLEAR THE SIGN KEY
025110  016601  000012          MOV     12(SP),R1       ;;GET THE MULTIPLICAND
025114  100002                  BPL     1$              ;;BR IF PLUS
025116  005216                  INC     (SP)            ;;SET THE SIGN KEY
025120  005401                  NEG     R1              ;;MAKE THE MULTIPLICAND POSTIVE
025122  016602  000014  1$:     MOV     14(SP),R2       ;;GET THE MULTIPLIER
025126  100002                  BPL     2$              ;;BR IF PLUS
025130  005316                  DEC     (SP)            ;;UPDATE THE SIGN KEY
025132  005402                  NEG     R2              ;;MAKE THE MULTIPLIER POSTIVE
025134  012746  000021  2$:     MOV     #17.,-(SP)      ;;SET THE LOOP COUNT
025140  005000                  CLR     R0              ;;SETUP FOR THE MULTIPLY LOOP
025142  103001          3$:     BCC     4$              ;;DON'T ADD IF MULTIPLICAND = 0
025144  060200                  ADD     R2,R0
025146  006000          4$:     ROR     R0              ;;POSITION THE PARITIAL PRODUCT AND
025150  006001                  ROR     R1              ;;THE MULTIPLICAND
025152  005316                  DEC     (SP)            ;;HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
025154  001372                  BNE     3$              ;;BR IF NO
025156  022616                  CMP     (SP)+,(SP)      ;;SHOULD PRODUCT BE NEGATIVE?
025160  001403                  BEQ     5$              ;;GO TO EXIT IF NO
025162  005400                  NEG     R0              ;;YES--SO MAKE IT SO
025164  005401                  NEG     R1
025166  005600                  SBC     R0
025170  005726          5$:     TST     (SP)+           ;;CLEAR SIGN INFO. OFF OF STACK
025172  010066  000012          MOV     R0,12(SP)       ;;PUT THE PRODUCT ON THE STACK (MSB'S
025176  010166  000010          MOV     R1,10(SP)       ;;LSB'S
025202  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
025204  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
025206  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
025210  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
025212  013746  000034          $DIV:   MOV     34,-(SP)        ;SAVE CURRENT TRAP VECTOR
025216  012737  025226  000034        MOV     #1$,34          ;SET UP TRAP VECTOR
025224  104400                        TRAP
025226  012716  025250          1$:     MOV     #2$,(SP)        ;REPLACE NEW PC
025232  016637  000004  000034        MOV     4(SP),34        ;RESTORE OLD TRAP VECT
025240  016666  000002  000004        MOV     2(SP),4(SP)     ;SAVE PSW
025246  000002                        RTI                     ;RESTORE PSW

025250  042716  000017          2$:     BIC     #17,(SP)        ;STRIP AWAY CONDITION CODES
025254  010046                        MOV     R0,-(SP)        ;PUSH R0 ON STACK
025256  010146                        MOV     R1,-(SP)        ;PUSH R1 ON STACK
025260  010246                        MOV     R2,-(SP)        ;PUSH R2 ON STACK
025262  010346                        MOV     R3,-(SP)        ;PUSH R3 ON STACK
025264  005046                        CLR     -(SP)           ;SAVE A PLACE FOR SIGNS
025266  012746  000021                MOV     #17.,-(SP)      ;SETUP THE ITERATION COUNTER
025272  016601  000024                MOV     24(SP),R1       ;PICKUP THE DIVIDEND
025276  016600  000022                MOV     22(SP),R0
025302  100005                        BPL     3$              ;CHECK THE SIGN
025304  105366  000003                DECB    3(SP)           ;KEEP TRACK OF THE SIGN
025310  005400                        NEG     R0              ;AND NEGATE THE ORIGINAL
025312  005401                        NEG     R1              ;NUMBER
025314  005600                        SBC     R0
025316  016602  000020          3$:     MOV     20(SP),R2       ;PICKUP THE DIVISOR
025322  022702  000001                CMP     #1,R2           ;IF THE DIVISOR IS 1 SKIP THE REST
025326  001463                        BEQ     13$             ;YES
025330  005702                        TST     R2
025332  002407                        BLT     4$              ;CHECK THE SIGN
025334  003011                        BGT     5$              ;DIVISOR OF 0 IS A NO-NO
025336  052766  000003  000014        BIS     #3,14(SP)       ;SET "V" & "C"
025344  012700  177777                MOV     #-1,R0          ;SET REMAINDER TO ALL ONES
025350  000424                        BR      9$              ;EXIT
025352  005266  000002          4$:     INC     2(SP)           ;KEEP TRACK OF DIVISORS SIGN
025356  000401                        BR      6$
025360  005402                  5$:     NEG     R2              ;NEGATE THE ORIGINAL NUMBER
025362  000241                  6$:     CLC                     ;CLEAR "C"
025364  000405                        BR      8$              ;START FORMING QUOTIENT
025366  006100                  7$:     ROL     R0              ;POSITION MSB'S
```

```
5808  025370  010003                        MOV     R0,R3           ;COPY
5809  025372  060203                        ADD     R2,R3           ;COMPARE DIVIDEND & DIVISOR
5810  025374  103001                        BCC     8$              ;BR IF DIVIDEND > DIVISOR
5811  025376  010300                        MOV     R3,R0           ;REMAINDER AFTER THIS LOOP
5812  025400  006101                8$:     ROL     R1              ;QUOTIENT BIT ENTERS HERE
5813  025402  005316                        DEC     (SP)            ;DONE?
5814  025404  001370                        BNE     7$              ;BR IF NO
5815  025406  005701                        TST     R1              ;OVERFLOW?
5816  025410  100005                        BPL     10$             ;BR IF NO
5817  025412  052766  000002  000014        BIS     #2,14(SP)       ;SET "V" IN RETURN STATUS WORD
5818  025420  005000                        CLR     R0              ;SET REMAINDER TO ALL ZEROS
5819  025422  010001                9$:     MOV     R0,R1           ;COPY REMAINDER INTO QUOTIENT
5820  025424  005726                10$:    TST     (SP)+           ;CLEAR COUNTER FROM STACK
5821  025426  005716                        TST     (SP)            ;REMAINDER SIGN CORRECTION NEEDED?
5822  025430  002004                        BGE     11$             ;BR IF NO
5823  025432  005400                        NEG     R0              ;NEGATE REMAINDER
5824  025434  105066  000001                CLRB    1(SP)           ;CLEAR SIGN
5825  025440  005316                        DEC     (SP)            ;BUT DON'T FORGET QUOTIENT
5826  025442  005726                11$:    TST     (SP)+           ;QUOTIENT SIGN CORRECTION NEEDED?
5827  025444  001401                        BEQ     12$             ;BR IF NO
5828  025446  005401                        NEG     R1              ;NEGATE QUOTIENT
5829  025450  010166  000020        12$:    MOV     R1,20(SP)       ;RETURN QUOTIENT AND
5830  025454  010066  000016                MOV     R0,16(SP)       ;REMAINDER TO USER
5831  025460  012603                        MOV     (SP)+,R3        ;POP STACK INTO R3
5832  025462  012602                        MOV     (SP)+,R2        ;POP STACK INTO R2
5833  025464  012601                        MOV     (SP)+,R1        ;POP STACK INTO R1
5834  025466  012600                        MOV     (SP)+,R0        ;POP STACK INTO R0
5835  025470  012666  000002                MOV     (SP)+,2(SP)     ;SETUP TO RETURN CONDITION CODES
5836  025474  000002                        RTI                     ;RETURN
5837  025476  022626                13$:    CMP     (SP)+,(SP)+     ;POP THE STACK
5838  025500  000763                        BR      12$
```

```
5839                                    .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
5840
5841                                    ;*********************************************************************
5842                                    ;*SAVE R0-R5
5843                                    ;*CALL:
5844                                    ;*      SAVREG
5845                                    ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
5846                                    ;*
5847                                    ;*TOP---(+16)
5848                                    ;* +2---(+18)
5849                                    ;* +4---R5
5850                                    ;* +6---R4
5851                                    ;* +8---R3
5852                                    ;*+10---R2
5853                                    ;*+12---R1
5854                                    ;*+14---R0
5855
5856  025502                            $SAVREG:
5857  025502  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
5858  025504  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
5859  025506  010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
5860  025510  010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5861  025512  010446                           MOV     R4,-(SP)        ;;PUSH R4 ON STACK
5862  025514  010546                           MOV     R5,-(SP)        ;;PUSH R5 ON STACK
5863  025516  016646  000022                   MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
5864  025522  016646  000022                   MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
5865  025526  016646  000022                   MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
5866  025532  016646  000022                   MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
5867  025536  000002                           RTI
5868
5869                                    ;*RESTORE R0-R5
5870                                    ;*CALL:
5871                                    ;*      RESREG
5872  025540                            $RESREG:
5873  025540  012666  000022                   MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
5874  025544  012666  000022                   MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
5875  025550  012666  000022                   MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
5876  025554  012666  000022                   MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
5877  025560  012605                           MOV     (SP)+,R5        ;;POP STACK INTO R5
5878  025562  012604                           MOV     (SP)+,R4        ;;POP STACK INTO R4
5879  025564  012603                           MOV     (SP)+,R3        ;;POP STACK INTO R3
5880  025566  012602                           MOV     (SP)+,R2        ;;POP STACK INTO R2
5881  025570  012601                           MOV     (SP)+,R1        ;;POP STACK INTO R1
5882  025572  012600                           MOV     (SP)+,R0        ;;POP STACK INTO R0
5883  025574  000002                           RTI
```

```
5884                                    .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
5885                                    ;*CALL:
5886                                    ;*      JSR     PC,$RAND        ;CALL THE ROUTINE
5887                                    ;*      RETURN                  ;RETURN HERE THE RANDOM
5888                                    ;*                              ;NUMBER WILL BE IN
5889                                    ;*                              ;$HINUM,$LONUM
5890  025576                            $RAND:
5891  025576  010046                           MOV     R0,-(SP)        ;PUSH R0 ON STACK
5892  025600  010146                           MOV     R1,-(SP)        ;PUSH R1 ON STACK
5893  025602  010246                           MOV     R2,-(SP)        ;PUSH R2 ON STACK
5894  025604  010346                           MOV     R3,-(SP)        ;PUSH R3 ON STACK
5895  025606  010446                           MOV     R4,-(SP)        ;PUSH R4 ON STACK
5896  025610  017604  000012                   MOV     @12(SP),R4      ;GET POINTER TO THE SAVED SEEDS
5897                                                                    ;FOR GENERATING THIS RANDOM NUMBER
5898  025614  011400                           MOV     (R4),R0         ;GET LO NUMBER SEED
5899  025616  016401  000002                   MOV     2(R4),R1        ;GET HIGH NUMBER SEED
5900  025622  012703  177771                   MOV     #-7,R3          ;SET SHIFT COUNT
5901  025626  005002                           CLR     R2              ;ZERO R2
5902  025630  006300                    1$:    ASL     R0              ;SHIFT R0 LEFT AND
5903  025632  006101                           ROL     R1              ;ROTATE CARRY INTO R1 AND
5904  025634  006102                           ROL     R2              ;ROTATE CARRY INTO R2
5905  025636  005203                           INC     R3              ;CHECK FOR DONE
5906  025640  001373                           BNE     1$              ;CONTINUE SHIFT LOOP
5907  025642  061400                           ADD     (R4),R0         ;ADD NUMBER TO MAKE X 129
5908  025644  005501                           ADC     R1              ;PROPOGATE CARRY
5909  025646  066401  000002                   ADD     2(R4),R1        ;ADD NUMBER TO MAKE X 129
5910  025652  005502                           ADC     R2              ;PROPOGATE CARRY
5911  025654  062700  001057                   ADD     #1057,R0        ;ADD LOW CONSTANT
5912  025660  005501                           ADC     R1              ;PROPOGATE CARRY
5913  025662  005502                           ADC     R2              ;PROPOGATE CARRY
5914  025664  062701  047401                   ADD     #47401,R1       ;ADD HIGH CONSTANT
5915  025670  005502                           ADC     R2              ;PROPOGATE CARRY
5916  025672  062702  000006                   ADD     #6,R2           ;ADD HIGHEST CONSTART
5917  025676  060200                           ADD     R2,R0           ;REPRIME R0 WITH HIGHEST DIGIT
5918  025700  005501                           ADC     R1              ;PROPOGATE CARRY
5919  025702  010014                           MOV     R0,(R4)         ;SAVE R0-$LONUM (FOR USE NXT TIME)
5920  025704  010164  000002                   MOV     R1,2(R4)        ;SAVE R1-$HINUM (FOR USE NXT TIME)
5921  025710  012604                           MOV     (SP)+,R4
5922  025712  012603                           MOV     (SP)+,R3        ;POP STACK INTO R3
5923  025714  012602                           MOV     (SP)+,R2        ;POP STACK INTO R2
5924  025716  012601                           MOV     (SP)+,R1        ;POP STACK INTO R1
5925  025720  012600                           MOV     (SP)+,R0        ;POP STACK INTO R0
5926  025722  062716  000002                   ADD     #2,(SP)         ;ADJUST SP FOR CORRECT RETURN
5927  025726  000207                           RTS     PC              ;RETURN
5928  025730  123456                    RSDRVL: 123456                  ;RANDOM SEED FOR DRIVE SELECTION (LO)
5929  025732  176543                    RSDRVH: 176543                  ;  "    "       (HI)
5930  025734  001201                    RSFUNL: 1201                    ;RANDOM SEED FOR FUNCTION
5931  025736  062465                    RSFUNH: 62465                   ;  "    "       (HI)
5932  025740  176105                    RSCYLL: 176105                  ;RANDOM SEED FOR CYLINDER (LO)
5933  025742  174532                    RSCYLH: 174532                  ;  "    "       (HI)
5934  025744  157650                    RSBAL:  157650                  ;RANDOM SEED FOR BUS †ADDRESS (LO)
5935  025746  030753                    RSBAH:  30753                   ;  "    "       (HI)
5936  025750  131547                    RSWCL:  131547                  ;RANDOM SEED FOR WORD COUNT (LO)
5937  025752  032070                    RSWCH:  32070                   ;  "    "       (HI)
5938  025754  123456                    RSDTL:  123456                  ;RANDOM SEED FOR DATA (LO)
5939  025756  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

 025760  017646  000000        $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
 025764  116637  000001 026203          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
 025772  112637  026205                 MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
 025776  062716  000002                 ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
 026002  000406                         BR      $TYPON
 026004  112737  000001 026203 $TYPOC:  MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
 026012  112737  000006 026205         MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
 026020  112737  000005 026202 $TYPON:  MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
 026026  010346                         MOV     R3,-(SP)        ;;SAVE R3
 026030  010446                         MOV     R4,-(SP)        ;;SAVE R4
 026032  010546                         MOV     R5,-(SP)        ;;SAVE R5
 026034  113704  026205                 MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
 026040  005404                         NEG     R4
 026042  062704  000006                 ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
 026046  110437  026204                 MOVB    R4,$OMODE       ;;SAVE IT FOR USE
 026052  113704  026203                 MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
 026056  016605  000012                 MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
 026062  005003                         CLR     R3              ;;CLEAR THE OUTPUT WORD
 026064  006105                 1$:     ROL     R5              ;;ROTATE MSB INTO "C"
 026066  000404                         BR      3$              ;;GO DO MSB
 026070  006105                 2$:     ROL     R5              ;;FORM THIS DIGIT
 026072  006105                         ROL     R5
 026074  006105                         ROL     R5
 026076  010503                         MOV     R5,R3
 026100  006103                 3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
 026102  105337  026204                 DECB    $OMODE          ;;TYPE THIS DIGIT?
 026106  100016                         BPL     7$              ;;BR IF NO
 026110  042703  177770                 BIC     #177770,R3      ;;GET RID OF JUNK
 026114  001002                         BNE     4$              ;;TEST FOR 0
 026116  005704                         TST     R4              ;;SUPPRESS THIS 0?
 026120  001403                         BEQ     5$              ;;BR IF YES
```

```
 5996  026122  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
 5997  026124  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
 5998  026130  052703  000040            5$:   BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
 5999  026134  110337  026200                  MOVB    R3,9$           ;;SAVE FOR TYPING
 6000  026140  104401  026200                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
 6001  026144  105337  026202            7$:   DECB    $OCNT           ;;COUNT BY 1
 6002  026150  003347                        BGT     2$              ;;BR IF MORE TO DO
 6003  026152  002402                        BLT     6$              ;;BR IF DONE
 6004  026154  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
 6005  026156  000744                        BR      2$              ;;GO DO THE LAST DIGIT
 6006  026160  012605                  6$:   MOV     (SP)+,R5        ;;RESTORE R5
 6007  026162  012604                        MOV     (SP)+,R4        ;;RESTORE R4
 6008  026164  012603                        MOV     (SP)+,R3        ;;RESTORE R3
 6009  026166  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
 6010  026174  012616                        MOV     (SP)+,(SP)
 6011  026176  000002                        RTI                     ;;RETURN
 6012  026200     000                  8$:   .BYTE   0               ;;STORAGE FOR ASCII DIGIT
 6013  026201     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
 6014  026202     000                  $OCNT:  .BYTE   0             ;;OCTAL DIGIT COUNTER
 6015  026203     000                  $OFILL: .BYTE   0             ;;ZERO FILL SWITCH
 6016  026204  000006                  $OMODE: .WORD   0             ;;NUMBER OF DIGITS TO TYPE
 6017
```

```
6018                                    .SBTTL  ERROR HANDLER ROUTINE
6019
6020                                    ;*SW15=1        HALT ON ERROR
6021                                    ;*SW13=1        INHIBIT ERROR TYPEOUTS
6022                                    ;*SW12=1        TYPE OUT THE ERROR HISTORY, THE FUNCTION THAT
6023                                    ;               WAS BEING PERFORMED ON RK AT THE TIME OF ERROR AND
6024                                    ;               THE FUNCTION PERFORMED PRIOR TO THAT.
6025                                    ;*NOTE THIS SWITCH OPTION (12) IS MEANINGFUL ONLY FOR ERRORS OCCURING IN THE
6026                                    ;*EXERCISER PART OF THE PROGRAM.
6027                                    ;*SW11=1        DUMP OUT ALL RK REGISTERS
6028                                    ;*SW10=1        BELL ON ERROR
6029                                    ;*SW09=1        LOOP ON ERROR
6030                                    ;*SW03=1        TYPE OUT TIME AT WHICH ERROR OCCURED
6031                                    ;*SW02=1        DROP THE DRIVE AFTER MAXM ERORS ON THIS DRIVE
6032                                    ;*SW01=1        TYPE OUT THE SERIAL NUMBER OF THE ERRORING DRIVE
6033                                    ;*GO TO $ERRTYP ON ERROR
6034
6035
6036  026206  104407                    $ERROR: CKSWR                   ;LOOK FOR A 'CONTROL G'
6037  026210  105237  001103                    INCB    $ERFLG          ;SET THE ERROR FLAG
6038  026214  001774                            BEQ     $ERROR          ;DON'T LET THE FLAG GO TO ZERO
6039  026216  013777  001102  152716            MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
6040  026224  032777  002000  152706            BIT     #SW10,@SWR      ;BELL ON ERROR?
6041  026232  001402                            BEQ     1$              ;NO - SKIP
6042  026234  104401  001206                    TYPE    ,$BELL          ;RING BELL
6043  026240  005237  001112            1$:     INC     $ERTTL          ;COUNT THE NUMBER OF ERRORS
6044
6045  026244  032777  000004  152666            BIT     #SW2,@SWR       ;COUNT # OF ERORS & DROP DRVE?
6046  026252  001415                            BEQ     3$              ;NO
6047                                                                    ;YES
6048  026254  010146                            MOV     R1,-(SP)        ;SAVE R1
6049  026256  013701  001250                    MOV     $RDRV,R1        ;GET ERRORING DRIVE #
6050  026262  100410                            BMI     2$              ;IF (SRDRV)= -1, SKIP(BECAUSE THE
6051                                                                    ;EROR WAS NOT ATTRIBUTABLE TO ANY
6052                                                                    ;SPECIFIC DRIVE)
6053  026264  105261  001542                    INCB    ERDRV(R1)       ;COUNT # OF ERORS ON THIS DRIVE
6054  026270  126127  001542  000003            CMPB    ERDRV(R1),#3    ;# OF ERORS GREATER THAN ALLOWABLE?
6055  026276  101402                            BLOS    2$              ;NO
6056  026300  000137  016312                    JMP     DSELCT          ;DROP THE DRIVE
6057
6058  026304  012601                    2$:     MOV     (SP)+,R1        ;RESTORE R1
6059
6060  026306  011637  001116            3$:     MOV     (SP),$ERRPC     ;GET ADDRESS OF ERROR INSTRUCTION
6061  026312  162737  000002  001116            SUB     #2,$ERRPC
6062  026320  005046                            CLR     -(SP)
6063  026322  117716  152570                    MOVB    @$ERRPC,(SP)    ;STRIP AND SAVE THE ERROR ITEM CODE
6064  026326  121627  000100                    CMPB    (SP),#100       ;FORM THE CO4$ECT ITEMB IF THIS IS AN
6065  026332  002402                            BLT     4$              ;EROR MESAGE EQUAL OR ABOVE 100.
6066  026334  162716  000040                    SUB     #40,(SP)        ;NOTE THERE R 2 CLASSES OF ERRORS:
6067                                                                    ;1) $ITEMB'S BELOW 100
6068                                                                    ;2) $ITEMB'S ABOVE 100
6069                                                                    ;SUBTRACTION FACTOR HAS TOBE SUCH THAT
6070                                                                    ;THE CO4$ECT OFFSET IS SELECTED. THIS
6071                                                                    ;FACTOR WILL CHANGE IF THE TOTAL # OF
6072                                                                    ;ERROR MESSAGES IN CLASS 1 CHANGES.
6073                                                                    ;#=100 -LAST ITEM IN # CLASS 1 - 1
```

```
6074  026340  012637  001114          4$:   MOV   (SP)+,$ITEMB
6075  026344  032777  020000  152566        BIT   #SW13,@SWR      ;SKIP TYPEOUT IF SET
6076  026352  001012                        BNE   5$              ;SKIP TYPEOUTS
6077  026354  004737  026472                JSR   PC,@#$ERRTYP    ;GO TO USER ERROR ROUTINE
6078  026360  104401  001213                TYPE  ,$CRLF
6079
6080  026364  032777  010000  152546        BIT   #SW12,@SWR      ;TYPE ERROR HISTORY?
6081  026372  001402                        BEQ   5$              ;NO
6082  026374  004737  020426                JSR   PC,HISTRY       ;YES
6083
6084  026400  023737  000042  000046  5$:   CMP   @#42,@#46       ;ACT11 AUTOMATIC MODE?
6085  026406  001403                        BEQ   .+10            ;YES, HALT ON ERROR
6086  026410  005777  152524                TST   @SWR            ;HALT ON ERROR
6087  026414  100002                        BPL   6$              ;SKIP IF CONTINUE
6088  026416  000000                        HALT                  ;HALT ON ERROR!
6089  026420  104407                        CKSWR                 ;LOOK FOR A 'CONTROL G'
6090  026422  032777  001000  152510  6$:   BIT   #SW09,@SWR      ;LOOP ON ERROR SWITCH SET?
6091  026430  001411                        BEQ   7$              ;BR IF NO
6092  026432  123727  001114  000040        CMPB  $ITEMB,#40      ;THERE R 37 ERROR MESSAGES IN CLASS 1
6093  026440  103011                        BHIS  8$
6094  026442  013746  001244                MOV   PPRLVL,-(SP)    ;LOCK OUT ALL INTERUPTS ON RETURN
6095                                                              ;FROM THIS EROR HANDLER, IF THE EROR
6096                                                              ;IS IN EXERCISER & LOOPING IS TO
6097                                                              ;BE DONE
6098  026446  005726                        TST   (SP)+
6099  026450  012716  015726                MOV   #EXCRLUP,(SP)   ;IF THIS ERROR CALL WAS FROM EXERCISER
6100                                                              ;PART OF THE PROGRAM, GO TO 'EXCRLUP'
6101                                                              ;OTHERWISE RETURN THRU '$LUPERR'
6102
6103  026454  012737  177777  001250  7$:   MOV   #-1,SRDRV       ;RESET SERIAL NO FLAG
6104                                                              ;IF (SRDRV)=-1, THEN THE SERIAL
6105                                                              ;NO OF THE DRIVE WILL NOT BE
6106                                                              ;TYPED OUT.
6107                                                              ;OTHERWISE, SERIAL NO FOR THE DRIVE
6108                                                              ;# IN 'SRDRV' WILL BE TYPED.
6109  026462  000002                        RTI
6110  026464  013716  001110          8$:   MOV   $LPERR,(SP)     ;FUDGE RETURN FOR LOOPING
6111  026470  000771                        BR    7$              ;RETURN
```

```
                        SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

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

026472 104401 001213    $ERRTYP:TYPE    $CRLF           ;"CARRIAGE RETURN" & "LINE FEED"
026476 010046                   MOV     R0,-(SP)        ;SAVE R0
026500 005000                   CLR     R0              ;PICKUP THE ITEM INDEX
026502 153700 001114            BISB    @#$ITEMB,R0
026506 001004                   BNE     1$              ;IF ITEM NUMBER IS ZERO, JUST
026510 013746 001116            MOV     $ERRPC,-(SP)    ;TYPE THE PC OF THE ERROR
026514 104402                   TYPOC
026516 000425                   BR      5$              ;GET OUT
026520 005300           1$:     DEC     R0              ;ADJUST THE INDEX SO THAT IT WIL
026522 006300                   ASL     R0              ;WORK FOR THE ERROR TABLE
026524 006300                   ASL     R0
026526 006300                   ASL     R0
026530 062700 002666            ADD     #$ERRTB,R0      ;FORM TABLE POINTER
026534 012037 026544            MOV     (R0)+,2$        ;PICKUP "ERROR MESSAGE" POINTER
026540 001404                   BEQ     4$              ;SKIP TYPEOUT IF NO POINTER
026542 104401                   TYPE                    ;TYPE THE "ERROR MESSAGE"
026544 000000           2$:     .WORD   0               ;"ERROR MESSAGE" POINTER GOES HERE
026546 104401 001213    3$:     TYPE    $CRLF           ;"CARRIAGE RETURN" & "LINE FEED"
026552 032777 004000 152360 4$: BIT     #SW11,@SWR      ;DUMP OUT RK REGISTERS?
026560 001404                   BEQ     5$
026562 004737 027032            JSR     PC,DMPREG       ;GO TYPE OUT RK REGISTERS
026566 104401 001213            TYPE    ,$CRLF

026572 012037 026602    5$:     MOV     (R0)+,6$        ;PICKUP "DATA HEADER" POINTER
026576 001404                   BEQ     7$              ;SKIP TYPEOUT IF 0
026600 104401                   TYPE                    ;TYPE THE "DATA HEADER"
026602 000000           6$:     .WORD   0               ;"DATA HEADER" POINTER GOES HERE
026604 104401 001213            TYPE    $CRLF           ;"CARRIAGE RETURN" & "LINE FEED"
026610 011000           7$:     MOV     (R0),R0         ;PICKUP "DATA TABLE" POINTER
026612 001406                   BEQ     9$

                                                        ;TYPE AN OCTAL NUMBER
026614 013046           8$:     MOV     @(R0)+,-(SP)    ;SAVE @(R0)+ FOR TYPEOUT
026616 104402                   TYPOC                   ;GO TYPE--OCTAL ASCII(ALL DIGITS.
026620 104401 002663            TYPE    ,BLNKS2         ;TYPE TWO(2) SPACES
026624 005710                   TST     (R0)            ;IS THERE ANOTHER NUMBER?
026626 001372                   BNE     8$              ;BR IF YES

026630 012600           9$:     MOV     (SP)+,R0        ;RESTORE R0
026632 104401 001213            TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
026636 123727 001114 000040     CMPB    $ITEMB,#40      ;SKIP TIME & SERIAL # FOR
                                                        ;NON-EXERCISER ERORS
026644 103004                   BHIS    10$
026646 004737 026660            JSR     PC,TIMTYP       ;TYPE OUT TIME IF SW 3 IS SET
026652 004737 026766            JSR     PC,SNOTYP       ;GO TYPE OUT THE SERIAL # OF THE
                                                        ;ERRORING DRIVE, IF SW 1 IS SET.
026656 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.
                                
026660  032777  000010  152252  TIMTYP: BIT     #SW3,@SWR       ;IS SW 3 SET?
026666  001434                          BEQ     4$              ;IF NOT SKIP TYPING TIME
026670  104401  002652                  TYPE    ,MSG29          ;'TIME'
026674  104401  002662                  TYPE    ,BLNKS3
026700  104414                          SAVREG                  ;SAVE R0-R4
026702  012700  001552                  MOV     #KWHR,R0        ;INITIALIZE POINTER
026706  012001                          MOV     (R0)+,R1
026710  000404                          BR      2$
026712  012001                  1$:     MOV     (R0)+,R1        ;TYPE OUT
026714  001402                          BEQ     2$
026716  062701  000074                  ADD     #60.,R1
026722  010137  026762          2$:     MOV     R1,5$
026726  012746  026762                  MOV     #5$,-(SP)       ;HOURS:MINS:SECS
026732  004737  024604                  JSR     PC,@#$DB2D      ;CONVERT TO ASCIZ STRING
026736  004737  025000                  JSR     PC,@#SUPRSL     ;GO TYPE
026742  020027  001560                  CMP     R0,#KWSEC+2     ;ALL DONE?
026746  001403                          BEQ     3$
026750  104401  002334                  TYPE    ,MSG18
026754  000756                          BR      1$
026756  104415                  3$:     RESREG                  ;RESTORE R0-R4
026760  000207                  4$:     RTS     PC              ;RETURN
026762  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.

026766  032777  000002  152144  SNOTYP: BIT     #SW1,@SWR       ;TYPE OUT SERIAL #?
026774  001415                          BEQ     2$              ;NO
026776  010146                          MOV     R1,-(SP)        ;SAVE R1
027000  013701  001250                  MOV     SRDRV,R1        ;GET ERRORING DRIVE #
027004  006301                          ASL     R1              ;IF (SRDRV)= -1, SKIP (BECAUSE
027006  100407                          BMI     1$              ;THE ERROR WAS NOT ATTRIBUTABLE
                                                                ;TO A SPECIFIC DRIVE)
027010  104401  001213                  TYPE    ,$CRLF
027014  104401  002326                  TYPE    ,MSG17          ;TYPE "SR. NO:"
027020  016146  001266                  MOV     SRNO(R1),-(SP)  ;GET THE SERIAL #
027024  104405                          TYPDS                   ;TYPE IT OUT (DECIMAL)

027026  012601                  1$:     MOV     (SP)+,R1        ;RESTORE R1
027030  000207                  2$:     RTS     PC              ;RETURN



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

027032                          DMPREG:
027032  104401  027040                  TYPE    ,65$            ;;TYPE ASCIZ STRING
027036  000441                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/  PC     RKDS    RKER    RKCS    RKWC    RKBA    RKDA    RKDB/
027142                          64$:
027142  013746  001116                  MOV     $ERRPC,-(SP)
027146  104402                          TYPOC
027150  104401  002663                  TYPE    ,BLNKS2
027154  010046                          MOV     R0,-(SP)
027156  012700  001216                  MOV     #RKDS,R0
027162  013046                  1$:     MOV     @(R0)+,-(SP)
027164  104402                          TYPOC
027166  104401  002663                  TYPE    ,BLNKS2
027172  020027  001232                  CMP     R0,#RKDB
027176  003771                          BLE     1$
027200  012600                          MOV     (SP)+,R0
027202  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

027204                          $SCOPE:
027204  104407                         CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
027206  032777  040000  151724  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
027214  001047                         BNE     $OVER           ;;YES IF SW14=1
                                ;*****START OF CODE FOR THE XOR TESTER*****
027216  000416                $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
027220  013746  000004                 MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
027224  012737  027244  000004         MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
027232  005737  177060                 TST     @#177060        ;;TIME OUT ON XOR?
027236  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
027242  000421                         BR      $SVLAD          ;;GO TO THE NEXT TEST
027244  022626                 5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
027246  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
027252  000407                         BR      7$              ;;LOOP ON THE PRESENT TEST
027254                          6$:;*****END OF CODE FOR THE XOR TESTER*****
027254  105737  001102         2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
027260  001412                         BEQ     $SVLAD          ;;BR IF NO
027262  032777  001000  151650         BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
027270  001404                         BEQ     4$              ;;BR IF NO
027272  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
027300  000415                         BR      $OVER
027302  105037  001103         4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
027306  105237  001102         $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
027312  011637  001106                 MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
027316  011637  001110                 MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
027322  005037  001204                 CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
027326  112737  000001  001115         MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON ME T TEST
027334  013777  001102  151600 $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
027342  013716  001106                 MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
027346  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.

027350  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
027352  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
027356  005740                          TST     -(R0)           ;;BACKUP BY 2
027360  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
027362  006300                          ASL     R0              ;;POSITION FOR INDEXING
027364  016000  027404                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
027370  000200                          RTS     R0              ;;GO TO ROUTINE


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

027372  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
027374  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
027402  000002                          RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------
027404  027372                  $TRPAD: .WORD   $TRAP2
027406  024244                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
027410  026004                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS.
027412  025760                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
027414  026020                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL.
027416  024020                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

027420  023034                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

027422  022764                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
027424  023246                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
027426  023366                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
027430  023540                          $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
027432  023642                          $RDDEC  ;;CALL=RDDEC    TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY
027434  025502                          $SAVREG ;;CALL=SAVREG   TRAP+14(104414) SAVE R0-R5 ROUTINE
027436  025540                          $RESREG ;;CALL=RESREG   TRAP+15(104415) RESTORE R0-R5 ROUTINE
027440  022356                          CN.RST  ;;CALL=CON.RESET        TRAP+16(104416) CONTROL RESET ROUTINE
027442  022364                          CN.RDY  ;;CALL=CON.RDY  TRAP+17(104417) WAIT FOR CONTROL READY
027444  022244                          DR.RST  ;;CALL=DRV.RESET        TRAP+20(104420) DRIVE RESET ROUTINE
027446  022522                          TY.MSG  ;;CALL=TYPMSG   TRAP+21(104421) TYPE MESSAGE ROUTINE. SW13
```

```
6330                                    .SBTTL  POWER DOWN AND UP ROUTINES
6331
6332                                    ;;**********************************************************************
6333                                    ;POWER DOWN ROUTINE
6334  027450  012737  027614  000024    $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
6335  027456  012737  000340  000026            MOV     #340,@#PWRVEC+2 ;;PRIO:7
6336  027464  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
6337  027466  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
6338  027470  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
6339  027472  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
6340  027474  010446                            MOV     R4,-(SP)        ;;PUSH R4 ON STACK
6341  027476  010546                            MOV     R5,-(SP)        ;;PUSH R5 ON STACK
6342  027500  017746  151434                    MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
6343  027504  010637  027620                    MOV     SP,$SAVR6       ;;SAVE SP
6344  027510  012737  027522  000024            MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
6345  027516  000000                            HALT
6346  027520  000776                            BR      .-2             ;;HANG UP
6347
6348                                    ;;**********************************************************************
6349                                    ;POWER UP ROUTINE
6350  027522  012737  027614  000024    $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
6351  027530  013706  027620                    MOV     $SAVR6,SP       ;;GET SP
6352  027534  005037  027620                    CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
6353  027540  005237  027620            1$:     INC     $SAVR6          ;;WAIT FOR THE INC
6354  027544  001375                            BNE     1$              ;;OF  WORD
6355  027546  012677  151366                    MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
6356  027552  012605                            MOV     (SP)+,R5        ;;POP STACK INTO R5
6357  027554  012604                            MOV     (SP)+,R4        ;;POP STACK INTO R4
6358  027556  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
6359  027560  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
6360  027562  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
6361  027564  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
6362  027566  012737  027450  000024            MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
6363  027574  012737  000340  000026            MOV     #340,@#PWRVEC+2 ;;PRIO:7
6364  027602  104401                            TYPE                    ;REPORT THE POWER FAILURE
6365  027604  027622                    $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
6366  027606  012716                            MOV     (PC)+,(SP)      ;;RESTART AT PFSTRT
6367  027610  003366                    $PWRAD: .WORD   PFSTRT          ;;RESTART ADDRESS
6368  027612  000002                            RTI
6369  027614  000000                    $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
6370  027616  000776                            BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
6371  027620  000000                    $SAVR6: 0                       ;;PUT THE SP HERE
6372  027622  005015  047520  042527    $POWER: .ASCIZ  <15><12>"POWER"
6373  027630  000122
6374                                            .EVEN
```

```
                                        ;ERROR MESSAGES

027632 051105 051117 047440  EM1:    .ASCIZ  /EROR ON WRITE/
027640 020116 051127 052111
027646 000105

027650 052101 046505 052120  EM2:    .ASCIZ  /ATEMPT TO INITIATE FUNCTION ON 'BUSY' DRVE/
027656 052040 020117 047111
027664 052111 040511 042524
027672 043040 047125 052103
027700 047511 020116 047117
027706 023440 052502 054523
027714 020047 051104 042526
027722    000
027723    103 052116 047522  EM3:    .ASCIZ  /CNTROL RDY NOT SET/
027730 020114 042122 020131
027736 047516 020124 042523
027744 000124

027746 051057 053457 051457  EM4:    .ASCIZ  "/R/W/S RDY NOT SET"
027754 051040 054504 047040
027762 052117 051440 052105
027770    000
027771    103 052116 047522  EM5:    .ASCIZ  /CNTROL RDY NOT SET AFTER 1ST INTRUPT ON ISSUING SEEK/
027776 020114 042122 020131
030004 047516 020124 042523
030012 020124 043101 042524
030020 020122 051461 020124
030026 047111 051124 050125
030034 020124 047117 044440
030042 051523 044525 043516
030050 051440 042505 000113

030056 051127 047117 020107  EM6:    .ASCIZ  /WRONG BITS IN RKCS, EXPCT SEEK/
030064 044502 051524 044440
030072 020116 045522 051503
030100 020054 054105 041520
030106 020124 042523 045505
030114    000
030115    047 052502 054523  EM7:    .ASCIZ  /'BUSY' FLAG CLEAR ON INTRUPTING DRVE/
030122 020047 046106 043501
030130 041440 042514 051101
030136 047440 020116 047111
030144 051124 050125 044524
030152 043516 042040 053122
030160 000105

030162 050047 051517 052111  EM10:   .ASCIZ  /'POSITIONING' FLAG FOR INTRUPTING DRVE CLEAR/
030170 047511 044516 043516
030176 020047 046106 043501
030204 043040 051117 044440
030212 052116 052522 052120
030220 047111 020107 051104
030226 042526 041440 042514
030234 051101    000
030237    047 051105 023522  EM11:   .ASCIZ  /'ERR'OR SET AFTER 1ST INTERRUPT ON ISSUING SEEK
030244 051117 051440 052105
030252 040440 052106 051105
030260 030440 052123 044440
030266 052116 051105 052522
```

```
030274  052120  047440  020116
030302  051511  052523  047111
030310  020107  042523  045505
030316     000
030317     123  050103  051440  EM12:   .ASCIZ  /SCP SET AFTER 1ST INTRUPT ON ISSUING SEEK/
030324  052105  040440  052106
030332  051105  030440  052123
030340  044440  052116  052522
030346  052120  047440  020116
030354  051511  052523  047111
030362  020107  042523  045505
030370     000
030371     103  052116  047522  EM13:   .ASCIZ  /CNTROL RDY NOT SET AFTER SEEK DONE INTRUPT/
030376  020114  042122  020131
030404  047516  020124  042523
030412  020124  043101  042524
030420  020122  042523  045505
030426  042040  047117  020105
030434  047111  051124  050125
030442  000124
030444  047111  051124  050125  EM14:   .ASCIZ  /INTRUPTING DRVE (SEEK DONE) WAS NOT 'BUSY'/
030452  044524  043516  042040
030460  053122  020105  051450
030466  042505  020113  047504
030474  042516  020051  040527
030502  020123  047516  020124
030510  041047  051525  023531
030516     000
030517     122  053457  051457  EM15:   .ASCIZ  "R/W/S READY NOT SET FOR INTRUPTING DRVE (SEEK DONE "
030524  051040  040505  054504
030532  047040  052117  051440
030540  052105  043040  051117
030546  044440  052116  052522
030554  052120  047111  020107
030562  051104  042526  024040
030570  042523  045505  042040
030576  047117  024505     000
030603     123  047111  042440  EM16:   .ASCIZ  /SIN EROR/
030610  047522  000122
030614  042447  051122  047447  EM17:   .ASCIZ  /'ERR'OR ON DOING SEEK/
030622  020122  047117  042040
030630  044517  043516  051440
030636  042505  000113
030642  041523  020120  044504  EM20:   .ASCIZ  /SCP DID NOT SET AFTER SEEK WAS DONE/
030650  020104  047516  020124
030656  042523  020124  043101
030664  042524  020122  042523
030672  045505  053440  051501
030700  042040  047117  000105
030706  047523  052106  042440  EM21:   .ASCIZ  /SOFT EROR/
030714  047522  000122
030720  040504  040524  024040  EM23:   .ASCIZ  /DATA (COMPARISON) EROR/
030726  047503  050115  051101
030734  051511  047117  020051
030742  051105  051117     000
030747     103  052116  047522  EM24:   .ASCIZ  /CNTROL RDY CLR ON INTRUPT AFTER RK FUNCTION
```

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

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

```
                                                    ;DATA HEADERS
031730  020040  041520  020040  DH1:    .ASCIZ  /  PC     RKCS    RKER    RKDS    RKDA/
031736  020040  051040  041513
031744  020123  020040  051040
031752  042513  020122  020040
031760  051040  042113  020123
031766  020040  051040  042113
031774  000101
                                                    
031776  020040  041520  020040  DH2:    .ASCIZ  /  PC     DRV#/
032004  020040  051104  021526
032012     000
                                                    
032013     040  050040  020103  DH21:   .ASCIZ  /  PC     RKCS    RKER    RKDS    DRIVE   CYL     SUR     SEC/
032020  020040  020040  045522
032026  051503  020040  020040
032034  045522  051105  020040
032042  020040  045522  051504
032050  020040  042040  044522
032056  042526  020040  020040
032064  054503  020114  020040
032072  020040  052523  020122
032100  020040  020040  042523
032106  000103
                                                    
032110  020040  041520  020040  DH23:   .ASCIZ  /  PC     RKBA    EXPCT   RECVD   RKDA/
032116  020040  045522  040502
032124  020040  020040  054105
032132  041520  020124  020040
032140  042522  053103  020104
032146  020040  045522  040504
032154     000
032155     040  050040  020103  DH25:   .ASCIZ  /  PC     RKCS    RKER    RKDS    RKDA    DRVE#/
032162  020040  020040  045522
032170  051503  020040  051040
032176  042513  020122  020040
032204  051040  042113  020123
032212  020040  051040  042113
032220  020101  020040  051104
032226  042526  000043
                                                    
032232  020040  041520  020040  DH27:   .ASCIZ  /  PC     KEY     FNCTN CODE/
032240  020040  045440  054505
032246  020040  020040  047106
032254  052103  020116  047503
032262  042504     000
032265     040  050040  020103  DH103:  .ASCIZ  /  PC     EXPCT   RECVD/
032272  020040  042440  050130
032300  052103  020040  051040
032306  041505  042126     000
                                                    
032313     040  050040  020103  DH30:   .ASCIZ  /  PC     KEY/
032320  020040  045440  054505
032326     000
032327     040  050040  020103  DH105:  .ASCIZ  /  PC     RKDA    RKWC/
032334  020040  020040  045522
```

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

```
ABORT   001672
ABRT    012046
ABRT1   012060
BAMAP   001530
BASEBA  002052
BCTST   005352
BEGEX1  010612
BEGNEX  010630
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BLNKS1  002664
BLNKS2  002663
BLNKS3  002662
BPTVEC= 000014
BUSY    001426
CBSY    021474
CHDPRS  022656
CHFAFN  011154
CHKBA   020150
CHKCRD  020272
CHKCS   020112
CHKDA   020126
CHKDRV  016540
CHKMEX  020204
CHKRWS  020254
CHKWC   020230
CH1     011210
CH2     011310
CICNT   001466
CICNT1  001470
CRSWR = 104407
CLEANB  006434
CLRERR  016006
CLRFLG  015770
CLRSIN  016152
CMNABT  014216
CMND    001326
CMNERR  014160
CNOBSY  021712
CN.RDY  022364
CN.RST  022356
COMRET  020306
CON.RD= 104417
CON.RE= 104416
CR    = 000015
CRCMND  020340
CRLF  = 000200
CROTLF  022170
CSE   = 000002
CSECN   001652
CYLMAP  001522
DATCHK  017110
DATER   001712
DBUF    032514
DDISP = 177570
DH1     031730
DH103   032265
DH105   032327
DH110   032354
DH2     031776
DH21    032013
DH23    032110
DH25    032155
DH27    032232
DH30    032313
DISPLA  001142
DISPRE  000174
DMPREG  027032
DOREAD  006424
DOWRIT  006304
DOXFER  006312
DPL   = 010000
DRCMND  020314
DRMAP   001520
DRU   = 002000
DRVABT  014426
DRVCNT  001500
DRVPRS  001264
DRVPTR  001476
DRV.RE= 104420
DRY   = 000200
DR.RST  022244
DSELCT  016312
DSWR = 177570
DT1     032422
DT103   032504
DT2     032436
DT21    032444
DT25    032466
ECOUNT  001540
EMTVEC= 000030
EM1     027632
EM10    030162
EM101   031513
EM102   031553
EM103   031567
EM104   031615
EM105   031643
EM106   031673
EM11    030237
EM110   031712
EM12    030317
EM13    030371
EM14    030444
EM15    030517
EM16    030603
EM17    030614
EM2     027650
EM20    030642
EM21    030706
EM23    030720
EM24    030747
EM26    031023
EM27    031076
EM3     027723
EM30    031137
EM31    031232
EM32    031323
EM33    031355
EM34    031432
EM35    031451
EM36    031467
EM37    031503
EM4     027746
EM5     027771
EM6     030056
EM7     030115
ERCODE  001474
ERDRV   001542
ERINF1  005720
ERR   = 100000
ERRVEC= 000004
EXCRLU  015726
EXCUT   011612
EXNSK   011470
EXRCSR  007772
FNCMND  020372
FNMAP   001526
FRSTRT  001253
FTITLE  001252
GENBUF  016704
GEN1    014724
GENBRQ  014500
GETINF  022064
GTSDRV  022226
GTSWR = 104406
GT3RG   022040
GT4RG   022032
HE    = 040000
HECN    001562
HISTRY  020426
HRDERR  014016
HT    = 000011
INTFLG  001534
INTHND  013256
INT1FL  001535
INT1SK  012376
IOTVEC= 000020
KDPAR0= 172360
KDPAR1= 172362
KDPAR2= 172364
KDPAR3= 172366
KDPAR4= 172370
KDPAR5= 172372
KDPAR6= 172374
KDPAR7= 172376
KDPDR0= 172320
KDPDR1= 172322
KDPDR2= 172324
KDPDR3= 172326
KDPDR4= 172330
KDPDR5= 172332
KDPDR6= 172334
KDPDR7= 172336
KEY     001306
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KWCOUN  001560
KWHR    001552
KWLS    001234
KWLVEC= 000100
KWMIN   001554
KWPLVL  001246
KWSEC   001556
KWSRVE  022552
LF    = 000012
MAXBA   002054
MH1     020746
MH2     020762
MH3     020772
MH4     020775
MMVEC = 000250
MSG1    002062
MSG10   002165
MSG11   002201
MSG12   002206
MSG13   002214
MSG14   002225
MSG15   002245
MSG16   002305
MSG17   002326
MSG18   002334
MSG19   002336
MSG2    002070
MSG20   002362
MSG24   002372
MSG25   002375
MSG26   002377
MSG26A  002474
MSG27   002532
MSG28   002604
MSG29   002652
MSG3    002076
MSG4    002104
MSG5    002120
MSG6    002133
MSG7    002141
MSG8    002146
MSG9    002156
NIEROR  021516
NOEROR  014300
NRDH    001774
NRDL    001772
NWRFNC  011766
NWRTH   001734
NWRTL   001732
NXTDRV  005366
```

```
ODDEVN  015724
PCMND   002032
PCNTR   001236
PDR     001254
PFSTRT  003366
PFTERR  013214
PGEND   032514
PIRQ    = 177772
PIRQVE= 000240
PORKER  013222
POS     001436
POSCMN  020350
POSK    012164
POSTIO  011414
PPRLVL  001244
PRICMN  011410
PRSFNC  001462
PR0     = 000000
PR1     = 000040
PR2     = 000100
PR3     = 000140
PR4     = 000200
PR5     = 000240
PR6     = 000300
PR7     = 000340
PS      = 177776
PSINER  013070
PSRTRY  013130
PSTFNC  001464
PSW     = 177776
PWRVEC= 000024
P1      020376
P2      020410
QBUSAD  001514
QCYL    001504
QDRV    001502
QERDR   021610
QFNC    001512
QMNGER  010650
QSCNT   001460
QSEC    001510
QSUR    001506
QWRCNT  001516
RCNT    017104
RDCHR   = 104410
RDDEC   = 104413
RDLIN   = 104411
RDOCT   = 104412
REPCNT  002056
REPSTA  021116
RESREG= 104415
RESVEC= 000010
RETRY   001446
RG4SDR  022222
RKBA    001226
RKCS    001222
RKDA    001230
RKDB    001232
RKDS    001216
RKER    001220
RKSTAT  001242
RKVEC   001240
RKWC    001224
RNUM    017106
RSBAH   025746
RSBAL   025744
RSCYLH  025742
RSCYLL  025740
RSDRVH  025732
RSDRVL  025730
RSDTH   025756
RSDTL   025754
RSFUNH  025736
RSFUNL  025734
RSWCH   025752
RSWCL   025750
RTIPC7  021732
RWS     = 000100
R6      =%000006
R7      =%000007
SAVKEY  001536
SAVREG= 104414
SCP     = 020000
SECMAP  001524
SEXIT   021720
SFTERR  013636
SIN     = 001000
SINCN   001622
SINCNT  016264
SKCMP   012704
SKE     = 010000
SKECN   001602
SNOTYP  026766
SRDRV   001250
SRNO    001266
SR0     = 177572
SR1     = 177574
SR2     = 177576
SR3     = 172516
STACK   = 001100
START   003376
STARTA  003444
STATST  021006
STATUS  021436
STKLMT= 177774
ST2     004400
ST3     005322
ST4     004752
SUPRS   025014
SUPRSL  025000
SUP1    025026
SUP2    025070
SWR     001140
SWREG   000176
SW0     = 000001
SW00    = 000001
SW01    = 000002
SW02    = 000004
SW03    = 000010
SW04    = 000020
SW05    = 000040
SW06    = 000100
SW07    = 000200
SW08    = 000400
SW09    = 001000
SW1     = 000002
SW10    = 002000
SW11    = 004000
SW12    = 010000
SW13    = 020000
SW14    = 040000
SW15    = 100000
SW2     = 000004
SW3     = 000010
SW4     = 000020
SW5     = 000040
SW6     = 000100
SW7     = 000200
SW8     = 000400
SW9     = 001000
TBITVE= 000014
TIMER   001472
TIMOUT  022354
TIMTYP  026660
TKVEC   = 000060
TPVEC   = 000064
TRAPVE= 000034
TRTVEC= 000014
TST1    005414
TST2    005526
TST3    005772
TST4    006106
TST5    006456
TST6    007040
TST7    007374
TYPDBO  017774
TYPDS   = 104405
TYPE    = 104401
TYPFN   021736
TYPMSG= 104421
TYPOC   = 104402
TYPON   = 104404
TYPOS   = 104403
TY.MSG  022522
WATIME  022630
WCE     = 000001
WCECN   001632
WCFLG   001456
WCMAP   001532
WPS     = 000040
WRDSK   010266
WRFNC   012064
XFR     007514
XFR16K  007504
XXDPMD  002060
$AUTOB  001134
$BDADR  001122
$BDDAT  001126
$BELL   001206
$CHARC  024460
$CKSWR  022764
$CMTAG  001100
$CM1    = 000011
$CM2    = 000022
$CM3    = 000011
$CNTLG  023511
$CNTLU  023504
$CORE   017674
$CRLF   001213
$CROUT  017724
$DBLK   024234
$DB2D   024604
$DB2O   024464
$DECVL  024764
$DIV    025212
$DOAGN  022760
$DTBL   024224
$ENDAD  022750
$ENDCT  022734
$EOP    022704
$EOPCT  022726
$ERFLG  001103
$ERMAX  001115
$ERROR  026206
$ERRPC  001116
$ERRTB  002666
$ERRTY  026472
$ERTTL  001112
$ESCAP  001204
$FILLC  001156
$FILLS  001155
$GDADR  001120
$GDDAT  001124
$GET42  022740
$GTSWR  023034
$HD     = 000000
$HIOCT  023640
$ICNT   001104
$ILLUP  027614
$INTAG  001135
$ITEMB  001114
$KTNEX  017666
$KTOUT  017656
$KTI1   017524
$LF     001214
$LPADR  001106
$LPERR  001110
$LSTAD  017770
$LSTBK  017772
$MNEW   023527
$MSWR   023516
$MUL    = 000003
$MULT   025100
$NULL   001154
$NWTST= 000001
$OCNT   026202
$OCTVL  024566
$OMODE  026204
$OVER   027234
$PASS   001100
$POWER  027622
$PWRAD  027610
$PWRDN  027450
$PWRMG  027604
$PWRUP  027522
$QUES   001212
$RAND   025576
$ROCHR  023246
$RODEC  023642
$ROLIN  023366
$ROOCT  023540
$ROSZ   = 000010
$REGAD  001160
$REG0   001162
$REG1   001164
$REG10  001202
$REG2   001166
$REG3   001170
$REG4   001172
$REG5   001174
$REG6   001176
$REG7   001200
$RESRE  025540
$RTNAD  022762
$SAVRE  025502
```

```
$SAVR6  027620          $SVPC = 000220          $TPB    001152          $TSTNM  001102          $TYPON  026020
$SCOPE  027204          $SWR  = 143000          $TPFLG  001157          $TTYIN  023474          $TYPOS  025760
$SETUP= 000115          $SWRMK= 000000          $TPS    001150          $TYPDS  024020          $XTSTR  027216
$SIZE   017466          $TKB    001146          $TRAP   027350          $TYPE   024244          $$GET4= 000000
$SIZEX  017730          $TKS    001144          $TRAP2  027372          $TYPEC  024414          $OFILL  026203
$STUP = 177777          $TN   = 000010          $TRP  = 000022          $TYPEX  024462          .       = 032516
$SVLAD  027306          $TNPWR  024714          $TRPAD  027404          $TYPOC  026004

. ABS.  032516      000


 ERRORS DETECTED:  0

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