RM80

FCTNL TEST PT 4
CZRNGAO

AH-T117A-MC
FICHE 1 OF 1

```
                        .REM    a
```

IDENTIFICATION

PRODUCT CODE: AC-T116A-MC

PRODUCT NAME: CZRNGA0 RM80 FUNCTIONAL TEST, PT 4

PRODUCT DATE: APRIL 1, 1982

MAINTAINER: CX DIAGNOSTIC GROUP

AUTHOR: MIKE LEAVITT

CONTENTS

1. ABSTRACT

THIS PROGRAM CONTAINS A SERIES OF TESTS THAT WILL VERIFY THAT THE DISK IS CAPABLE OF PERFORMING SEEKS, THAT THE SEEK TIMES ARE WITHIN TOLERANCE AND THAT THE TRACK/SECTOR ADDRESSING CIRCUITRY OPERATES PROPERLY. THIS PROGRAM IS PRIMARILY USED TO DETERMINE IF THE SEEK TIMES ARE WITHIN TOLERANCES AND NOT FOR DATA RELIABLITY. THERE IS ABSOLUTELY NO WRITTING OF DATA INVOLVED IN THIS PROGRAM. HOWEVER, THIS PROGRAM DOES PERFORM READ HEADER AND DATA COMMANDS TO DETERMINE IF THE SEEKS PERFORMED ARE VALID.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
12K MEMORY
TERMINAL
PROGRAM LOADING DEVICE
KW11-L OR KW11-P (THE KW11-P IS REQUIRED FOR THE TIMING TESTS)
RH70 CONTROLLER
1 TO 8 RM80 DISK DRIVES

2.2 PRELIMINARY PROGRAMS

RM80 DISKLESS TEST, PART 1 & 2

RM80 FUNCTIONAL TEST, PART 1, 2 & 3

2.3 MEDIA

THE PROGRAM REQUIRES THAT EACH DRIVE TO BE TESTED HAS A FORMATTED DISK. THE DISK MAY BE FORMATTED IN EITHER 16-BIT OR 18-BIT MODE, DEPENDING ON THE TESTING REQUIREMENTS. NOTE THAT THE PROGRAM WILL NOT TEST A MIXTURE OF DRIVES WITH BOTH 16 AND 18 BIT MODE PACKS.

3. LOADING PROCEDURE

THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE 'XXDP' MEDIA USING THE ASSOCAITED LOADER. THE PROGRAM MAY BE INCLUDED IN AN 'XXDP' CHAIN.

4. STARTING PROCEDURE

4.1 STARTING ADDRESSES

200 NORMAL STARTING ADDRESS
204 SELECT OPERATING PARAMETERS
210 SELECT RH CONTROLLER ADDRESSES
214 COMBINATION OF 204 AND 210

NOTE: STARTING ADDRESSES 210 AND 214 ARE AVAILABLE WHEN THE PROGRAM IS INITIALLY STARTED; THESE STARTING ADDRESSES ARE TREATED AS ADDRESSES 200 OR 204 RESPECTIVELY ON RESTARTS.

4.2 OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3.)
2. BRING DRIVE(S) TO ONLINE STATE, WRITE ENABLED, AND LOCKED ON PORT.
3. LOAD ADDRESS 200.
4. SET SWITCHES (SEE SECTION 5.)
5. PRESS START.
6. THE PROGRAM WILL TYPEOUT THE STATUS OF THE DRIVES ATTACHED TO THE SELECTED MASSBUS SUBSYSTEM. TO INHIBIT THIS TYPEOUT, DO NOT RESTART THE PROGRAM FROM ANY OF THE STARTING ADDRESSES; INSTEAD TYPE A 'CONTROL C' ON THE KEYBOARD TO RETURN THE PROGRAM TO COMMAND ENTRY MODE.

4.3 PROGRAM ACTION

IN AN EFFORT TO ALLOW CONVERSATION WITH A PROGRAM FOR THE PURPOSE OF CONTROLLING ITS OPERATION AND PARAMETERS THE FOLLOWING CONSTRUCTIONS HAVE BEEN ADOPTED.

NOTE1: IN ALL EXAMPLES BRACKETS ARE USED FOR CLARITY AND ARE NOT TYPED BY THE USER.

NOTE2: THE CARRRIAGE RETURN TYPED BY THE USER IS INDICATED BY <CR> AND WILL BE ECHOED AS A 'CARRIAGE RETURN-LINE FEED'.

<.><CR> PERIOD

A STATEMENT TERMINATOR: WHEN TYPED AT THE END OF A LINE (LEGAL ON ALL LINES) IT TELLS THE PARAMETER STRING INTERPRETER (PSI) THIS IS THE END OF CHANGES TO THE CURRENT PARAMETER STRING.

<..><CR> PERIOD PERIOD

THE 'PERIOD PERIOD' TERMINATOR IS TYPED TO INDICATE THE END OF TEST PARAMETER MODIFICATION AND TO SIGNAL THE START OF TEST EXECUTION.

<,><CR> COMMA

THE COMMA IS USED AS A SEPARATOR BETWEEN DRIVE NUMBERS AND TEST NUMBERS.

</> SLASH

A MODIFICATION INDICATOR: IF A SLASH FOLLOWS A TEST NUMBER, THE PROGRAM WILL OPEN THAT TEST FOR PARAMETER MODIFICATION.

<^U> CONTROL-U

DELETE THE PRESENT INPUT STRING AND START A NEW LINE. TYPED BY DEPRESSING THE "CONTROL KEY" (CTRL) AND THEN STRIKING THE "U".

<\> RUBOUT

DELETE THE LAST CHARACTER FROM THE INPUT STRING. TYPED BY STRIKING THE "RUBOUT" KEY. WHICH WILL BE ECHOED BY A BACKSLASH (\) FOLLOWED BY THE CHARACTER DELETED.

4.3.1 CONTROL SWITCH SELECTION

STARTING THE PROGRAM AT ANY OF THE POSSIBLE STARTING ADDRESSES WITH SW<07>=1 WILL RESULT IN ENTERING THE "CONTROL SWITCH SETTING" MODE. THUS, ALLOWING THE OPERATOR TO SPECIFY THE DESIRED STATE OF "C.SWR".

CONTROL SWITCH SELECTION EXAMPLES:

EXAMPLE #1

```
SET SW<07>=0
C.SWR=000000 / 400..
```

EXAMPLE #2

```
SET SW<07>=0
C.SWR=000000 / 220.
C.SWR=000000 / 220..
```

4.3.2 RH CONTROLLER ADDRESS SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECT OF THE DEFAULT VALUES OF BUS ADDRESS (RMCS1), AND VECTOR ADDRESS OF THE RH CONTROLLER. IF THE DEFAULT VAULE OF THE BUS ADDRESS DOES NOT RESPOND (TIMES OUT) WHEN ADDRESSED, AN ERROR IS REPORTED. AFTER THE ERROR IS REPORTED ONE OF TWO COURSES OF ACTION WILL BE TAKEN:

1. IF THERE IS A MONITOR -- RETURN TO THE MONITOR

2. IF THERE ISN'T A MONITOR -- ASK FOR NEW ADDRESSES

STARTING THE PROGRAM AT 210 OR 214 ALLOWS THE OPERATOR TO CHANGE THE ADDRESS OF THE RH CONTROLLER AND THE VECTOR ADDRESS.

ADDRESS SELECTION EXAMPLES

```
                                EXAMPLE #1

                                  RMCS1=176700 / 177200.

                                EXAMPLE #2

                                  RMCS1=176700 / 176300<CR>
                                  RMVEC=254 / 260<CR>

                                EXAMPLE #3

                                  RMCS1=176700<CR>
                                  RMVEC=254 / 260.

                                EXAMPLE #4

                                  RH/RM FAILED TO RESPOND TO ADDRESSING
                                  RMCS1   ERR PC
                                  176300  XXXXXX
                                  RMCS1=176300 / 176700.

                                EXAMPLE #5

                                  RMCS1=176700 / 1776\67\6300<CR>
                                  RMVEC=254<CR>
                                  RMCS1=176300.

        4.3.3   DRIVE AND PARAMETER SELECTION

                STARTING THE PROGRAM AT 200 OR 210 WILL RESULT IN AUTOMATIC
                SELECTION OF THE DRIVES TO TEST AND THE TESTS TO RUN.

                STARTING THE PROGRAM AT 204 OR 214 ALLOWS THE OPERATOR
                TO SELECT THE DRIVE(S) TO BE TESTED, THE TESTS TO BE EXECUTED,
                AND THE PARAMETERS TO USE.

        4.3.3.1 DRIVE AND PARAMETER SELECTION DESCRIPTION

                THE FOLLOWING IS A TABLE OF TERMS USED BY THE PSI.

                        "R"     REPEATS (ITERATIONS)
                        "FC"    FIRST CYLINDER ADDRESS
                        "LC"    LAST CYLINDER ADDRESS
                        "IC"    INCREMENT CYLINDER
                        "FT"    FIRST TRACK ADDRESS
                        "LT"    LAST TRACK ADDRESS
                        "IT"    INCREMENT TRACK
                        "FS"    FIRST SECTOR ADDRESS
                        "LS"    LAST SECTOR ADDRESS
                        "PAT"   PATTERN (USED FOR DATA TEST) (NOT USED)
                        "WDX"   WORD OF PATTERN 0 WHERE X IS 1 TO 16 (NOT USED)

                       *"S"     ALL SEEK TESTS (TESTS 0 - 7)
                       *"T"     ALL TIMING TESTS (TESTS 11 - 14)

                        * USED BY THE OPERATOR TO SELECT TEST GROUPS
```

SPECIAL CASES OF CONTROL CHARACTERS

IF <..> IS TYPED WHILE A TEST IS OPEN FOR MODIFICATION (</>) AND OTHER TESTS IN THE "TEST COMMAND" STRING ARE TO BE MODIFIED, THE REMAINING TESTS WILL BE UNCHANGED.

WHEN THE PROGRAM IS STARTED FROM LOCATION 200 OR 210, TESTS 0-7 AND 11, 12 & 14 WILL BE RUN USING ALL AVAILABLE, ONLINE DRIVES. IF THE OPERATOR WISHES TO SELECT THE DRIVES TO BE TESTED, THE TESTS TO BE PERFORMED, OR THE PARAMETERS TO BE USED, THE CONVERSATION MODE MAY BE ENTERED BY TYPING A 'CONTROL C' OR BY STARTING THE PROGRAM FROM EITHER LOCATION 204 OR 214.

THE PROGRAM WILL THEN RESPOND WITH:

```
DRIVE(S)=
```

THE FOLLOWING EXAMPLES ASSUME THAT THE OPERATOR IS TO TEST DRIVE #3 USING TESTS 2 THRU 7 AND TEST 11 AND DOES NOT DESIRE TO CHANGE THE PARAMETERS (INITIAL CYLINDER ADDRESS, FINAL CYLINDER ADDRESS, ETC.). THE USER WOULD TYPE '3<CR>' WHICH SAYS 'THIS IS THE END OF DRIVE ENTRY'. THE PROGRAM WILL THEN REQUEST TEST NUMBERS.

THE TRANSACTION APPEARS AS FOLLOWS:

```
DRIVE(S)=3<CR>
TEST=
```

THE OPERATOR MAY NOW ENTER DESIRED TEST NUMBERS. IN THE EXAMPLE, HE WANTS TESTS 2 THRU 7 AND TEST 11 SO HE TYPES 2-7<,> (THE 'COMMA' SEPARATES ENTRIES), 11<.><CR> ('PERIOD' 'CARRIAGE RETURN' - END OF CHANGES, START TEST EXECUTION.)

IT NOW LOOKS LIKE THIS

```
DRIVE(S)=3<CR>
TEST=2-7,11.<CR>
```

IN THE NEXT EXAMPLE, IT IS ASSUMED THAT THE OPERATOR WISHES TO TEST DRIVE 4 AND TO RUN TESTS 1 AND 3 THRU 11, MODIFYING THE PARAMETERS FOR TESTS 3 AND 10.

THE TRANSACTION WOULD BE AS FOLLOWS:

```
DRIVE(S)=4<CR>
TEST=
```

THE OPERATOR NOW ENTERS THE TEST NUMBERS. THE TRANSACTION IS GIVEN BELOW:

```
DRIVE(S)=4<CR>
TEST=1,3/4-7,10/11<CR>
```

NOTICE THIS SAYS SELECT TEST 1, CONTINUE<,>; SELECT TEST 3, OPEN</>;

SELECT TESTS 4-7, CONTINUE<,>; SELECT TEST 10, OPEN</>; SELECT TEST 11, END OF INPUT <.>.

THE PROGRAM SCANS THE TEST NUMBER INPUT AND DETERMINES THAT THE PARAMETERS FOR TEST 3 AND TEST 10 ARE TO BE CHANGED. THE OTHER TESTS WILL NOT BE ALTERED.

(THE ENTIRE TRANSACTION IS REPEATED FOR CLARITY)

```
DRIVE(S)=4<CR>
TEST=1,3/4-7,10/11<CR>
  TEST 3
    R=X /                    ;WHERE X IS ITERATION
```

THE NEW VALUE FOR 'R' MAY BE ENTERED. TERMINATING THE ENTRY WITH A <.> (PERIOD) WILL TERMINATE THE CHANGES FOR THIS TEST; TYPING A <CR> OR TERMINATING THE ENTRY WITH A <CR> WILL CAUSE THE PROGRAM TO MOVE TO THE NEXT PARAMETER.

```
DRIVE(S)=4<CR>
TEST=1,3/4-7,10/11<CR>
  TEST 3
    R=1 / <CR>               ;DO NOT ALTER-BUT CONTINUE
    FC=N /                   ;WHERE 'N' IS FIRST CYLINDER ADDRESS
```

IF THE OPERATOR DOES NOT WISH TO CHANGE 'FC', THE FOLLOWING OCCURS:

```
DRIVE(S)=4<CR>
TEST=1,3/4-7,10/11.<CR>
  TEST 3
    R=1 / <CR>               ;DO NOT ALTER THIS LINE BUT CONTINUE
    FC=0 / <CR>              ;DO NOT ALTER THIS LINE BUT CONTINUE
    LC=560 /
```

THE PROGRAM RESPONDS WITH THE PREVIOUSLY ASSIGNED PARAMETER FOR LAST CYLINDER ADDRESS IN THIS CASE USING 560 AS THE EXAMPLE. THIS IS WHAT THE OPERATOR INTENDED TO MODIFY AND IS WHY TEST 3 WAS OPENED. TO CHANGE THE VALUE TO '20', THE NEW VALUE IS TYPED FOLLOWED BY A 'PERIOD' TERMINATOR (<.><CR>).

THE TOTAL TRANSACTION AND RESPONSE:

```
DRIVE(S)=4<CR>
TEST=1,3/4-7,10/11<CR>
  TEST 3
    R=1 / <CR>
    FC=0 / <CR>
    LC= 560 / 20.<CR>
  TEST 10
    R=1 /
```

THE PROGRAM HAS LOADED TEST 3 WITH ITS NEW PARAMETERS AND THE PROGRAM IS WAITING FOR CHANGES TO TEST 10'S PARAMETERS.

```
DRIVE(S)=4<CR>
TEST=1,3/4-7,10/11<CR>
  TEST 3
```

```
                                        R=1 / <CR>
                                        FC=0 / <CR>
                                        LC= 560 / 20.<CR>
                                      TEST 10
                                        R=1 / 10.<CR>
```

THE OPERATOR TYPES THE NEW VALUE (10) AND TERMINATES THE ENTRY WITH A 'PERIOD' 'CARRIAGE RETURN'.

THE PROGRAM NOW LOADS TEST 10 WITH THE NEW PARAMETERS (TEST 11 RETAINS THE PREVIOUSLY ASSIGNED PARAMETERS) AND RESPONDS WITH:

```
        DRIVE(S)=
```

SINCE THE USER DID NOT END THE CONVERSATION MODE WITH A 'PERIOD PERIOD', THE PROGRAM HAS LOOPED BACK TO THE BEGINNING LOOKING FOR MORE CHANGES. THAT IS TO SAY, AFTER THE ENTRY FOR DRIVE SELECTION, A <,><CR> WILL CAUSE THE TEST MESSAGE TO BE REPEATED AND FURTHER CHANGES CAN BE MADE. HOWEVER, AT SOME POINT IN ORDER TO EXECUTE THE PROGRAM, A 'PERIOD PERIOD' MUST BE TYPED.

IF A SINGLE 'PERIOD' IS TYPED WHILE DRIVE OR TEST NUMBERS ARE BEING ENTERED, THE PROGRAM WILL START EXECUTION IMMEDIATELY. A 'PERIOD PERIOD' MUST BE TYPED BEFORE THE PROGRAM WILL EXIT TEST PARAMETER CHANGE MODE TO GO TO EXECUTION.

4.3.3.2 DRIVE AND PARAMETER SELECTION EXAMPLES

EXAMPLE #1

```
DRIVE=4.<CR>              ;SELECT DRIVE #4, TERMINATE AND
                          ;BEGIN EXECUTION USING PREVIOUSLY ASSIGNED
                          ;PARAMETERS
```

EXAMPLE #2

```
DRIVE=0<CR>               ;SELECT DRIVE #0 AND MAKE CHANGES ","
TEST=1-5.<CR>             ;RUN TEST 1 THRU 5 ONLY, USE DEFAULT
                          ;PARAMETERS AND TERMINATE AND EXECUTE."
```

EXAMPLE #3

```
DRIVE=2<CR>               ;SELECT DRIVE #2 AND MAKE CHANGES ","
TEST=1-5,6/7/10/<CR>      ;RUN TEST 1-5 WITH DEFAULT PARAMETERS, OPEN
  TEST 6                  ;TEST 6,7 AND 10 FOR CHANGES
    R=1 / <CR>            ;LEAVE 'R' AS IS AND MOVE TO NEXT PARAMETER
    FC=0 / 10.<CR>        ;SET 'FC' CYLINDER ADDRESS TO 10, END CHANGES
                          ;TO TEST 6.

  TEST 7
    R=1 / 50<CR>          ;50 ITERATIONS, MOVE TO NEXT PARAMETER
    FC=^ / <CR>           ;DO NOT CHANGE 'FC' CYLINDER ADDRESS BUT CONTINUE
    LC=560 / 50..<CR>     ;TEST 10 IS STILL PENDING AND WILL BE
```

```
                                                    ;RETAIN ITS PRESENT PARAMETERS.


                                  EXAMPLE #4
                                  ----------

                                  DRIVE=0<CR>       ;SELECT DRIVE #0 AND MAKE CHANGES
                                  TEST=S.<CR>       ;RUN ALL SEEK TESTS

                                  EXAMPLE #5
                                  ----------

                                  DRIVE=1<CR>
                                  TEST=S/T<CR>      ;RUN ALL SEEK TESTS (OPEN FOR CHANGES) AND
                                    TEST 0          ;THE TIMING TESTS (WITH DEFAULT PARAMETERS).
                                      R=10 / <CR>   ;RUN WITH 10 ITERATIONS
                                      FC=0 / 10..<CR>   ;CHANGE FIRST CYLINDER ADDRESS
                                                        ;AND START EXECUTION
                                                        ;TESTS 1 - 10 WILL RETAIN THEIR PREVIOUSLY
                                                        ;ASSIGNED PARAMETERS.

                                  EXAMPLE #6
                                  ----------

                                  DRIVE=1<CR>
                                  TEST=S/<CR>       ;OPEN THE SEEK TESTS (TESTS 0-7)
                                    TEST 0
                                      R=10 / 100.<CR>   ;CHANGE TO 100 ITERATIONS, TO TO THE NEXT TEST
                                    TEST 1
                                      R=100 / 1000.<CR> ;CHANGE 'R' TO 1000 ITERATIONS, MOVE TO NEXT TEST
                                    TEST 2
                                      R=1 / 10<CR>      ;CHANGE 'R' TO 10 ITERATIONS, GO TO NEXT PARAMETER
                                      FC=0 / 50<CR>     ;CHANGE 'FC' TO 50, GO TO NEXT PARAMETER
                                      LC=560 / 51.<CR>  ;CHANGE 'LC' TO 51, GO TO THE NEXT TEST
                                    TEST 3
                                      R=1.<CR>          ;MOVE TO NEXT TEST
                                    TEST 4
                                      R=1..<CR>         :USE TEST 4'S PARAMETERS AND START PROGRAM EXECUTION

                                  EXAMPLE #7
                                  ----------

                                  DRIVE=0,1,4<CR>   ;TEST DRIVES 0,1, AND 4 IN SEQUENCE
                                  TEST=0-5/<CR>     ;CHANGE TEST 5
                                    TEST 0
                                      R=10 / <CR>
                                      FC=0 / <CR>
                                      LC=560 / 1..<CR>  ;CHANGE LAST CYLINDER FROM 560 TO 1
                                                        ;START PROGRAM EXECUTION.


5.      SWITCH SETTINGS
        ---------------

5.1     OPERATIONAL SWITCH SETTINGS
```

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR 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 . THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS IN KEYBOARD ENTRY MODE, OR IS AT A HIGHER PRIORITY PROCESSING AN DRIVE INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

EACH TIME SWITCH SETTINGS 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 1'S IN THE SWITCHES. ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

THE SWITCH SETTINGS ARE:

SW<15>=1	HALT ON ERROR
SW<14>=1	LOOP ON TEST
SW<13>=1	INHIBIT ERROR TYPEOUTS
SW<12>=1	TYPE TEST NUMBER
SW<11>=1	INHIBIT ITERATIONS
SW<10>=1	RING BELL ON ERROR
SW<09>=1	LOOP ON ERROR
SW<08>=1	PRINT ERROR MESSAGE ON LINE PRINTER
SW<07>=1	READ 'C.SWR' SETTINGS FROM TTY
SW<06>=1	INHIBIT TIME REPORTS (TESTS 11-15)

5.2 CONTROL SWITCH SETTINGS

THE CONTROL SWITCH SETTINGS ARE ENTERED THROUGH THE KEYBOARD.

TO ENTER THE CONTROL SWITCH SETTING MODE PLACE SW<07>=1 BEFORE PRESSING START. THEN UPON STARTING THE PROGRAM IT WILL TYPE THE PRESENT CONTENTS OF THE CONTROL SWITCH REGISTER (C.SWR) AND WAIT FOR THE NEW SETTING TO BE INPUT. THE INPUT STRING MUST CONSIST OF 1 TO 6 OCTAL DIGITS, TWO PERIODS (..), AND A CARRIAGE RETURN.

THE C.SWR SETTINGS ARE:

C.SWR<14>=0	NO STALL BETWEEN DRIVE FUNCTIONS
=1	STALL AFTER EVERY DRIVE FUNCTION
C.SWR<13>=0	USE SPECIFIC STALL TIMES
=1	USE RANDOM STALL TIMES
C.SWR<12>=0	NO INCREMENTING STALLS IN TEST 4
=1	PERFORM INCREMENTING STALLS IN TEST 4

```
                        C.SWR<08>=0     DO IMPLIED SEEKS WITH DATA TRANSFERS
                                =1      DO EXPLICIT SEEKS BEFORE DATA TRANSFERS
                        C.SWR<07>=0     DO READ HEADER AND  DATA COMMANDS IN TESTS 0-7
                                =1      DO EXPLICIT SEEK COMMANDS IN TESTS 0-7
                        C.SWR<06>=0     60 HZ POWER SOURCE
                                =1      50 HZ POWER SOURCE
                        C.SWR<05>=0     ALLOW SOFTWARE TIMEOUTS(ENABLE WATCHDOG TIMER)
                                =1      INHIBIT SOFTWARE TIMEOUTS(DISABLE WATCHDOG TIMER)
                        C.SWR<00>=0     OPERATE IN 32. SECTOR (16 BIT) MODE
                                =1      OPERATE IN 30. SECTOR (18 BIT) MODE

                        THE DEFAULT CONDITION OF C.SWR<15:00>=0.

                        REFER TO 4.3.1 FOR C.SWR SELECTION



                6.      ERRORS
                        ------

                        THERE ARE ANUMBER OF ERRORS THAT CAN OCCUR IN THIS PROGRAM.
                        WHEN AN ERROR IS ENCOUNTERED, THE CALL TO THE ERROR ROUTINE
                        IS MADE AND IF SW<13> IS NOT SET, AN ERROR MESSAGE PERTAINING
                        TO THE ERROR WILL BE TYPED.  EACH ERROR TYPEOUT WILL CONTAIN
                        THE FOLLOWING:

                        1.  AN ERROR MESSAGE
                        2.  A DATA HEADER
                        3.  A DATA STRING

                        REFER TO THE FOLLOWING SECTION FOR THE DIFFERENT ERRORS
                        THAT CAN OCCUR.

                6.1     ERROR TYPES

                        THE ERRORS THAT OCCUR IN THIS PROGRAM FALL INTO THREE
                        (3) CATEGORIES DEFINED AND EXPLAINED AS FOLLOWS:

                6.1.1   DRIVER ERROR

                        THESE ERRORS WILL BE DETECTED BY THE RH/RM DRIVER.
                        THERE ARE TWO CLASSES OF DRIVER ERRORS; THOSE THAT
                        CAN NOT BE IDENTIFIED IN A MANNER THAT ALLOWS THE
                        INFORMATION TO BE RETURNED TO A "DATA PARAMETER BLOCK"
                        (DPB) AND THOSE THAT CAN. THE FIRST CLASS WILL BE
                        REPORTED BY ERROR CALLS (EMT'S) 1-5 WITHIN THE DRIVER.
                        THE SECOND CLASS WILL PASS THE ERROR CODES TO THE
                        STATUS/ERROR WORD (DPB+16) OF THE PROPER DPB.

                6.1.2   NON-FATAL ERRORS

                        THESE ERRORS WILL BE DUE TO "DISK" OR "DATA" FAILURES
                        WHICH WILL BE REPORTED AS THEY OCCUR.  AFTER REPORTING
                        THE ERROR THE PROGRAM WILL CONTINUE TESTING.

                6.1.3   FATAL ERRORS
```

THIS TYPE OF ERROR WILL BE THE RESULT OF ANY KIND
OF ERROR THAT INHIBITS THE PROGRAM FROM TESTING THE DISK.

THIS ERROR WILL BE REPORTED WHEN IT OCCURS, THEN THE PROGRAM
WILL ABORT THE TEST AND GO TO THE END OF PROGRAM.

6.2 ERROR RECOVERY

6.2.1 PRETEST ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED. THEN DEPENDING
ON HOW THE PROGRAM WAS STARTED IT WILL ASK FOR THE DRIVES AND
ADDRESSES FOR TESTING OR RETURN TO MONITOR.

6.2.2 NON-FATAL ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED AND
THE PROGRAM WILL CONTINUE IN TEST.

6.2.3 FATAL ERROR

WHEN THIS TYPE OF EPROR OCCURS IT WILL BE REPORTED. THE
PROGRAM WILL ABORT THE TEST AND GO TO THE END OF PROGRAM.

7. RESTRICTIONS

THE PROGRAM WILL TEST THE DRIVES IN EITHER 16 BIT MODE OR IN 18
BIT MODE DEPENDING ON THE SETTING OF 'S.SWR<00>'. IF 'C.SWR<00>'
IS 0, ALL OF THE DRIVES WILL BE TESTED IN 16 BIT MODE; IF 'C.SWR<00>
IS 1, ALL OF THE DRIVES WILL BE TESTED IN 18 BIT MODE. THE PROGRAM
HAS NO PROVISIONS FOR TESTING DRIVES WITH INTERMIXED PACKS OR TESTING
BOTH 16 BIT MODE AND 18 BIT MODE DRIVES ON THE SAME SYSTEM.
ACT11 AUTOMATIC MODE ASSUMES 16 BIT MODE.

BEFORE THE PROGRAM IS STARTED, PROPERLY FORMATTED PACKS MUST BE MOUNTED
ON THE DRIVES WHICH WILL BE TESTED. THE PROGRAM ASSUMES A
PROPERLY FORMATTED PACK. THE FORMAT OF THE PACK IS NOT ALTERED
BY THE PROGRAM.

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE PROGRAM REQUIRES APPROXIMATELY 10 MINUTES TO MAKE ONE PASS WITH
RM80 DRIVES. THIS ASSUMES THE DEFAULT TEST SEQUENCE (TESTS 0-7 AND
11, 12 & 14) AND DEFAULT TEST PARAMETERS.

8.2 STACK POINTER

THE STACK POINTER IS INITIALLY SET TO 1100.

8.3 TIMING TESTS (TESTS 11-15) PRINTOUTS

AT THE COMPLETION OF EACH OF THE TIMING TESTS THE TIME OF THE MINIMUM SEEK, MAXIMUM SEEK, AND THE AVERAGE OF ALL OF THE SEEKS PERFORMED ARE TYPED ON THE TTY. THE NUMBER OF SEEKS THAT HAD TIMES BELOW THE MINIMUM TIME ALLOWED WILL BE TYPED ON THE SAME LINE AS THE MINIMUM TIME. THE NUMBER ABOVE THE MAXIMUM WILL BE TYPED ON THE SAME LINE AS THE MAXIMUM TIME, AND THE TOTAL NUMBER OF SEEKS PERFORMED WILL BE ON THE SAME LINE AS THE AVERAGE.

8.3.1 TIMING TOLERANCES

1. TEST 11 -- ROTATIONAL SPEED TIMES

   60 HZ
   MINIMUM=15970 US
   MAXIMUM=17270 US
   NOMINAL=16670 US

   50 HZ
   MINIMUM=15970 US
   MAXIMUM=17270 US
   NOMINAL=16670 US

2. TEST 12 -- ONE CYLINDER SEEK TIMES

   MAXIMUM=6000 US

3. TEST 13 -- AVERAGE SEEK TIME TEST (NOT DEFAULT)

   MAXIMUM=25000 US

   ** THERE ARE NO SPECIFICATIONS GIVEN FOR AN AVERAGE SEEK TIME **
   ** ON THIS PARTICULAR DRIVE. THEREFORE, THIS TEST SHOULD BE **
   ** USED FOR REFERENCE ONLY. **

4. TEST 14 -- MAXIMUM SEEK TIMES

   MAXIMUM (FORWARD)=50000 US

   ** THERE IS NO SPECIFICATION GIVEN FOR THE MAXIMUM REVERSE **
   ** SEEK TIME ON THIS PARTICULAR DRIVE. THEREFORE, ANY REVERSE **
   ** SEEK TIMES ABOVE THE MAXIMUM TIME OF 50.0 MS WILL NOT BE **
   ** TYPED IN THE TIMING REPORT. HOWEVER, THE TIMING REPORT **
   ** WILL STILL TYPE THE MINIMUM, MAXIMUM AND AVERAGE TIMES **
   ** FOR THE REVERSE SEEKS. (SEE SECTION 8.3.2, EX. 2) **

5. TEST 15 -- AVERAGE SEEK TIME MEASUREMENT (NOT DEFAULT)

   MAXIMUM=25000 US

   ** WARNING - THIS TEST TAKES APPROXIMATELY 3.0 HOURS TO RUN **

8.3.2. TIMING TESTS PRINTOUT EXAMPLES

EXAMPLE #1

```
ROTATIONAL SPEED TIMES
MIN=16670 US
MAX=16690 US
AVG=16680 US  10 SEARCHES TIMED

ALLOWABLE ROTATIONAL SPEED LIMITS FOR RM80
MIN=15970 US
MAX=17270 US

ONE CYLINDER SEEK TIMES
 * FORWARD
MIN=5350 US
MAX=5920 US
AVG=5550 US  559 SEEKS TIMED
 * REVERSE
MIN=5040 US
MAX=5970 US
AVG=5330 US  560 SEEKS TIMED

ALLOWABLE ONE CYLINDER SEEK LIMIT
MAX=6000 US

AVERAGE SEEK TIMES
 * FORWARD
MIN=23770 US
MAX=24640 US
AVG=24230 US  128 SEEKS TIMED
 * REVERSE
MIN=23990 US
MAX=24550 US
AVG=24220 US  128 SEEKS TIMED

ALLOWABLE AVERAGE TIME LIMIT
MAX=25000 US

MAXIMUM SEEK TIMES
 * FORWARD
MIN=47990 US
MAX=49980 US
AVG=48010 US  128 SEEKS TIMED
 * REVERSE
MIN=47120 US
MAX=49650 US
AVG=48340 US  128 SEEKS TIMED

ALLOWABLE MAXIMUM (FORWARD) SEEK TIME LIMIT
MAX=50000 US

EXAMPLE #2
```

```
                                    ----------

                                    ROTATIONAL SPEED TIMES
                                    MIN=16670 US
                                    MAX=16690 US
                                    AVG=16680 US  10 SEARCHES TIMED

                                    ALLOWABLE ROTATIONAL SPEED LIMITS FOR RM80
                                    MIN=15970 US
                                    MAX=17270 US

                                    ONE CYLINDER SEEK TIMES
                                     * FORWARD
                                    MIN=5470 US
                                    MAX=7940 US  3 ABOVE THE MAXIMUM OF 6000 US
                                    AVG=5830 US  559 SEEKS TIMED
                                     * REVERSE
                                    MIN=5040 US
                                    MAX=5970 US
                                    AVG=5330 US  560 SEEKS TIMED

                                    ALLOWABLE ONE CYLINDER SEEK LIMIT
                                    MAX=6000 US

                                    AVERAGE SEEK TIMES
                                     * FORWARD
                                    MIN=24730 US
                                    MAX=32620 US  73 ABOVE THE MAXIMUM OF 25000 US
                                    AVG=30320 US  128 SEEKS TIMED
                                     * REVERSE
                                    MIN=25620
                                    MAX=32230 US  128 ABOVE THE MAXIMUM OF 25000 US
                                    AVG=32800 US  128 SEEKS TIMED

                                    ALLOWABLE AVERAGE TIME LIMIT
                                    MAX=25000 US

                                    MAXIMUM SEEK TIMES
                                     * FORWARD
                                    MIN=50510 US
                                    MAX=57240 US  128 ABOVE THE MAXIMUM OF 50000 US
                                    AVG=56020 US  128 SEEKS TIMED
                                     * REVERSE
                                    MIN=50050 US
                                    MAX=57550 US
                                    AVG=56210 US  128 SEEKS TIMED

                                    ALLOWABLE MAXIMUM (FORWARD) SEEK TIME LIMIT
                                    MAX=50000 US


                        8.4         END OF TEST

                                    WITH ALL SWITCHES ON A "0" AN "END OF PASS" MESSAGE WILL BE
                                    TYPED AT THE COMPLETION OF TESTING A DRIVE AND THE "END OF TEST"
                                    TYPEOUT WILL OCCUR WHEN ALL DRIVES HAVE BEEN TESTED. ALSO, THE
                                    END OF TEST COULD OCCUR ON A DRIVE, IF THE MAXIMUM ERROR LIMIT
```

IN LOCATION 'ERMAX' IS EXCEEDED.

9. PROGRAM DESCRIPTION

THIS PROGRAM CONTAINS 16 TESTS, NUMBERED 0-15 IN OCTAL. TESTS 0-7 WILL READ THE CYLINDER, TRACK, AND SECTOR INFORMATION FROM THE HEADER, USING A "READ HEADER AND DATA" COMMAND, AND THEN CHECK THE INFORMATION FOR VALIDITY. THUS, INSURING THE SEEK OPERATION FUNCTIONS PROPERLY. TESTS 11-14 WILL MEASURE THE ROTATIONAL SPEED, THE ONE CYLINDER SEEK, THE AVERAGE SEEK TIME, AND THE MAXIMUM SEEK TIMES TO ENSURE THEY ARE ALL WITHIN THE TOLERANCES ALLOWED. TEST 15 MEASURES THE AVERAGE SEEK TIME FOR UNIQUE NUMBER OF SEEKS, BUT DOES NOT HAVE ANY PASS/FAIL LIMITS.

THE PROGRAM WILL START BY IDENTIFING ITSELF AND DETERMINING ALL DRIVES THAT ARE AVAILABLE FOR TESTING. THEN BEGINNING WITH THE LOWEST NUMERICAL DRIVE AND PROCEEDING IN SEQUENTIAL ORDER. ALL OF THE DRIVES WILL BE TESTED. ONE PASS THROUGH THE TEST SEQUENCE (TESTS 0-7 AND 11, 12 & 14) WILL BE PERFORMED ON EACH DRIVE BEFORE MOVING TO THE NEXT DRIVE IN SEQUENCE. DRIVE TO BE TESTED WILL BE TYPED AT THE BEGINNING OF EACH PASS, AN "END OF PASS" MESSAGE WILL BE TYPED AT THE COMPLETION OF EACH PASS, AND AN "END OF TEST" MESSAGE WILL BE TYPED AFTER TESTING ALL DRIVES.

REFER TO THE FOLLOWING SECTIONS FOR DETAILED DESCRIPTIONS OF EACH TEST.

9.1 TEST 0 - RECAL/RANDOM SEEK TEST

THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A RECALIBRATE COMMAND AND THEN SEEK TO A RANDOM CYLINDER BETWEEN "FC" AND "LC". AT THE COMPLETION OF BOTH COMMANDS, STATUS INDICATORS ARE CHECKED TO ENSURE THAT NO ERRORS OCCURRED.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	200
FC	-	0
LC	-	560
FT	-	0
FS	-	0

9.2 TEST 1 - SEEK/SEEK TEST

THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A FORWARD SEEK CYCLE TO "LC", "LT", "LS" FOLLOWED BY A REVERSE SEEK CYCLE TO "FC", "FT", "FS". AT THE COMPLETION OF EACH SEEK, THE PROPER INDICATORS ARE EXAMINED TO ENSURE PROPER OPERATION.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	100
FC	-	0

```
                                        LC      -       256
                                        IC      -         0
                                        FT      -         0
                                        LT      -         0
                                        FS      -         0
                                        LS      -         0

                9.3     TEST 2 - INCREMENTAL SEEK TEST

                        THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
                        CYLINDER ADDRESS FROM "FC" TO "LC" BY THE INCREMENT "IC".
                        WHEN THE RESJLTANT CYLINDER ADDRESS (NC) EXCEEDS
                        "LC" REVERSE SEEK CYCLES ARE INITIATED; STARTING
                        AT THE LAST LEGAL "NC" AND DECREMENTING BY "IC"
                        UNTIL "NC" IS LESS THAN "FC". AT THE COMPLETION OF EACH
                        SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
                        ENSURE PROPER OPERATION.

                        THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

                                        R       -         1
                                        FC      -         0
                                        LC      -       560
                                        IC      -         1
                                        FT      -         0
                                        FS      -         0

                9.4     TEST 3 - STEPPING SEEK TEST

                        THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0, 1, 2, 4, 8,
                        16, 32, 64, 128, 256 AND 512.  AT THE COMPLETION OF EACH SEEK
                        COMMAND THE PROPER INDICATORS ARE EXAMINED TO VERIFY PROPER
                        OPERATION.

                        THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

                                        R       -         8
                                        FC      -         0
                                        LC      -       512
                                        IC      -         1
                                        FT      -         0
                                        FS      -         0

                9.5     TEST 4 - OSCILLATING SEEK TEST

                        THIS TEST WILL COMMAND SEEK CYCLES FROM "FC" TO "NC" AND BACK
                        TO "FC".  "NC" STARTS AT "FC" AND INCREMENTS BY "IC" UP TO CYLINDER
                        "LC", THEN IS DECREMENTED BY "IC" BACK TO CYLINDER "FC".  AT THE
                        COMPLETION OF EVERY SEEK COMMAND THE PROPER INDICATORS ARE
                        EXAMINED TO ENSURE PROPER OPERATION.

                        THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

                                        R       -         1
                                        FC      -         0
                                        LC      -       560
                                        IC      -         1
```

```
                                        FT      -       0
                                        FS      -       0


        9.6     TEST 5 - CONVERGING/DIVERGING SEEK TEST

                THIS TEST WILL CAUSE THE DRIVE TO EXECUTE FORWARD AND REVERSE
                SEEKS FROM 'NC1" AND 'NC2" RESPECTIVELY. 'NC1" WILL BE INCREMENTED
                BY "IC" AND 'NC2" WILL BE DECREMENTED BY "IC" UNTIL 'NC1" IS
                GREATER THAN THE INITIAL VALUE OF 'NC2" AND 'NC2" IS LESS THAN
                THE INITIAL VALUE OF 'NC1". AT THE COMPLETION OF EACH SEEK
                COMMAND THE PROPER INDICATORS ARE EXAMINED TO ENSURE PROPER
                OPERATION.  'NC1" AND 'NC2" DEFAULT TO "FC" AND "LC" RESPECTIVELY.

                THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

                        R       -       1
                        FC      -       0
                        LC      -       560
                        IC      -       1
                        FT      -       0
                        FS      -       0

        9.7     TEST 6 - SERVO ADDRESSING LOGIC NOISE GENERATOR TEST

                IN THIS TEST A SEEK IS DONE TO CYL 'NC" THEN A SEEK TO
                NC+4 THEN NC+1 THEN NC+3 THEN NC+2 THEN NC+5.  NOW 'NC" IS UPDATED
                BY "IC" AND THE ABOVE SEQUENCE IS REPEATED UNITL 'LC" IS
                EXCEEDED BY ANY OF THE ABOVE VALUES.  THE INITIAL VALUE OF 'NC"
                IS "FC".  AT THE COMPLETION OF EACH SEEK COMMAND THE
                PROPER INDICATORS ARE EXAMINED TO ENSURE PROPER OPERATION.

                THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

                        R       -       1
                        FC      -       0
                        LC      -       560
                        IC      -       1
                        FT      -       0
                        FS      -       0

        9.8     TEST 7 - RANDOM SEEK TEST (NOT DEFAULT)

                THIS TEST PERFORMS RANDOM SEEK OPERATIONS BETWEEN CYLINDERS 'FC'
                'LC'.  AFTER EACH SEEK, THE POSITION OF THE DRIVE IS VERIFIED BY
                READING A SECTOR FROM THE CURRENTLY ADDRESSED CYLINDER AND TRACK.
                THE TRACK ADDRESS IS INCREMENTED FOR EACH SEEK SO THAT VERIFICATION
                OF POSITIONING OCCURS USING EACH HEAD.  TRACK ADDRESSES ARE INCREMENTED
                BETWEEN PARAMTERS 'FT' AND 'LT'.

                THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

                        R       -       5000
                        FC      -       0
                        LC      -       560
                        FT      -       0
                        LT      -       4
```

9.9 TEST 10 - ALL SEEKS TEST

THIS TEST VERIFIES THAT THE DISK DRIVE CAN SEEK FROM EACH CYLINDER TO ALL OTHER CYLINDERS.

BEGINNING WITH CYLINDER 'FC', THE TEST SEEKS TO EACH CYLINDER BETWEEN 'FC' AND 'LC' FROM CYINDER 'FC'. THE BEGINNING CYLINDER ADDRESS IS INCREMENTED AND THE TEST SEEKS BETWEEN THE NEW CYLINDER ADDRESS AND ALL CYLINDERS BETWEEN 'FC' AND 'LC'. THE SEQUENCE CONTINUES UNTIL ALL CYLINDERS HAVE BEEN CHECKED.

THE FOLLOWING PARAMETERS ARE USED BY THIS TEST:

R	-	1
FC	-	0
LC	-	560
IC	-	1
FT	-	0
FS	-	0

9.10 TEST 11 - ROTATIONAL SPEED TIMING TEST

THIS TEST WILL START A SEARCH TO CYLINDER 'FC', TRACK 'FT', SECTOR 'FS'. AS SOON AS THE INTERRUPT OCCURS, THE GO BIT IS SET AGAIN AND THE OPERATION IS TIMED. THIS PROCEDURE IS REPEATED 10 TIMES THEN THE AVERAGE TIME IS CALCULATED AND CHECKED TO ENSURE IT IS WITHIN TOLERANCE:

LO LIMIT (16.67MS - 4.3%) = 15.97MS IF 50/60 HZ.
HI LIMIT (16.67MS + 3.5%) = 17.27MS IF 50/60 HZ.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
FT	-	0
FS	-	0

9.11 TEST 12 - ONE CYLINDER SEEK TIMING TEST

THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE CYLINDER BY ONE UNTIL THE INCREMENT IS GREATER THAN THE CYLINDER 'LC', THEN REVERSE SEEK TO CYLINDER 'FC'. THE TIME TO PERFORM EACH SEEK IS CHECKED TO ENSURE IT DOES NOT EXCEED THE MAXIMUM TIME PERMITTED FOR A ONE CYLINDER SEEK. MAXIMUM TIME IS 6.0 MS

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	560

9.12 TEST 13 - AVERAGE SEEK TIMING TEST (NOT DEFAULT)

THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 'FC' TO CYLINDER 'LC', THEN A REVERSE SEEK FROM CYLINDER 'LC' TO

```
1027                    CYLINDER 'FC'.  BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE THEY
1028                    ARE WITHIN THE TOLERANCE ALLOWED FOR THE AVERAGE SEEK TIME.
1029                    THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL OF 256 SEEKS).
1030                    MAXIMUM TIME IS 25.0 MS
1031
1032                    ** THERE ARE NO SPECIFICATIONS GIVEN  FOR AN AVERAGE SEEK TIME **
1033                    ** ON  THIS PARTICULAR  DRIVE.  THEREFORE, THIS TEST SHOULD BE **
1034                    ** USED FOR REFERENCE ONLY.                                    **
1035
1036                    THE TEST USES THE FOLLOWING PARAMETERS:
1037
1038                            R       -       1
1039                            FC      -       0
1040                            LC      -     155
1041
1042            9.13    TEST 14 - MAXIMUM SEEK TIMING TEST
1043
1044                    THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 'FC' TO
1045                    CYLINDER 'LC', THEN A REVERSE SEEK FROM CYLINDER 'LC' TO
1046                    CYLINDER 'FC'.  BOTH SEEKS ARE TIMED, BUT ONLY THE FORWARD SEEKS
1047                    ARE CHECKED TO ENSURE THEY ARE WITHIN THE TOLERANCE ALLOWED FOR
1048                    THE MAXIMUM SEEK TIME.  THIS SEQUENCE IS REPEATED 128 TIMES (FOR
1049                    A TOTAL OF 256. SEEKS). MAXIMUM (FORWARD) TIME IS 50.0 MS
1050
1051                    ** THERE IS  NO SPECIFICATION  GIVEN FOR THE  MAXIMUM REVERSE **
1052                    ** SEEK TIME ON THIS PARTICULAR DRIVE. THEREFORE, ANY REVERSE **
1053                    ** SEEK TIMES ABOVE THE  MAXIMUM TIME OF 50.0 MS WILL  NOT BE **
1054                    ** TYPED  IN THE  TIMING REPORT.  HOWEVER,  THE TIMING REPORT **
1055                    ** WILL STILL  TYPE THE  MINIMUM, MAXIMUM  AND  AVERAGE TIMES **
1056                    ** FOR THE REVERSE SEEKS.                                     **
1057
1058                    THE TEST USES THE FOLLOWING PARAMETERS:
1059
1060                            R       -       1
1061                            FC      -       0
1062                            LC      -     560
1063
1064            9.14    TEST 15 - AVERAGE SEEK TIME MEASUREMENT TEST (NOT DEFAULT)
1065
1066                    THIS TEST MEASURES AVERAGE SEEK TIME BY DOING ALL UNIQUE SEEKS
1067                    BETWEEN 'FC' AND 'LC' AND MEASURING THE SEEK TIMES.  THE TOTAL
1068                    SEEK TIME FOR ALL SEEKS IS THEN DIVIDED BY THE NUMBER OF SEEKS
1069                    PERFORMED TO  PROVIDE AN AVERAGE SEEK TIME VALUE FOR THE RANGE
1070                    OF SEEKS TIMED. MAXIMUM TIME IS 25.0 MS
1071
1072                    ** WARNING - THIS TEST TAKES APPROXIMATELY 3.0 HOURS TO RUN **
1073
1074                    THE TEST USES THE FOLLOWING PARAMETERS:
1075
1076                            R       -       1
1077                            FC      -       0
1078                            LC      -     560
1079
1080            10.     PROGRAM LISTING
1081                    ---------------
1082            a
```

```
                                ;*LAST REVISION 24-SEP-81
                                .TITLE  CZRNGA0 RM80 FCTNL PT4
                                ;*COPYRIGHT (C) 1982
                                ;*DIGITAL EQUIPMENT CORPORATION
                                ;*COLORADO SPGS., CO. 80919
                                ;*
                                ;*PROGRAM BY MIKE LEAVITT
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81
                                ;*
                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        12            TYPE TEST NUMBER
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            PRINT ERROR MESSAGE ON LINE PRINTER
                                ;*         7            READ 'C.SWR' SETTINGS FROM TTY
                                ;*         6            INHIBIT TIME REPORTS (TESTS 11-14)

                                .SBTTL  CONTROL SWITCH SETTINGS
                                ;*
                                ;*    SWITCH  STATE             USE
                                ;*    ------  -----     ------------------------------------
                                ;*      14      0       NO STALL BETWEEN DRIVE FUNCTIONS
                                ;*              1       STALL AFTER EVERY DRIVE FUNCTION
                                ;*      13      0       USE SPECIFIC STALL TIME
                                ;*              1       USE RANDOM STALL
                                ;*      12      0       NO INCREMENTING STALL IN TEST 4
                                ;*              1       DO INCREMENTING STALL IN TEST 4
                                ;*       7      0       DO "READ HEADER AND DATA" IN TESTS 0-10
                                ;*              1       DO EXPLICIT SEEKS IN TESTS 0-10
                                ;*       6      0       60 HZ
                                ;*              1       50 HZ
                                ;*       5      0       RUN WATCHDOG TIMER
                                ;*              1       INHIBIT WATCHDOG TIMER
                                ;*       0      0       TEST DRIVE(S) IN 31. SECTOR (16 BIT) MODE
                                ;*              1       TEST DRIVE(S) IN 30. SECTOR (18 BIT) MODE

                                .SBTTL  BASIC DEFINITIONS

                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                          STACK   = 1100
104000                          ERROR   = EMT           ;;BASIC DEFINITION OF FRROR CALL
000004                          SCOPE   = IOT           ;;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
```

BASIC DEFINITIONS

```
177776                  PS      = 177776        ;;PROCESSOR STATUS WORD
177776                  PSW=PS
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
001000                  SW9=SW09
000400                  SW8=SW08
000200                  SW7=SW07
000100                  SW6=SW06
000040                  SW5=SW05
000020                  SW4=SW04
000010                  SW3=SW03
000004                  SW2=SW02
000002                  SW1=SW01
000001                  SW0=SW00
```

```
                        ;*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
001000                  BIT9=BIT09
000400                  BIT8=BIT08
000200                  BIT7=BIT07
000100                  BIT6=BIT06
000040                  BIT5=BIT05
000020                  BIT4=BIT04
000010                  BIT3=BIT03
000004                  BIT2=BIT02
000002                  BIT1=BIT01
000001                  BIT0=BIT00

                        ;*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
81
82                      ;CONTROL AND STATUS REGISTER 1 (RMCS1)
83
84      000100          IE      = 100                   ;INTERRUPT ENABLE (BIT #6)
85      000200          RDY     = 200                   ;READY (BIT #7)
86      000400          A16     = 400                   ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
87      001000          A17     = 1000                  ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
88      002000          PSEL    = 2000                  ;PORT SELECT (BIT #10)
89      020000          MCPE    = 20000                 ;MASSBUSS PARITY ERROR (BIT #13)
90      040000          TRE     = 40000                 ;TRANSFER ERROR (BIT #14)
91                      ;SC     = 100000                ;SPECIAL CONDITION (BIT #15)
92
93                      ;WORD COUNT REGISTER (RMWC)
94                      ;(EACH BIT IS CALLED BY BIT NUMBER)
95
96                      ;BUS ADDRESS REGISTER (RMBA)
```

```
                                ;(EACH BIT IS CALLED BY BIT NUMBER)

                                ;CONTROL AND STATUS REGISTER 2 (RMCS2)

000001                          US1     = 1                     ;UNIT SELECT (BIT #0)
000002                          US2     = 2                     ;UNIT SELECT (BIT #1)
000004                          US4     = 4                     ;UNIT SELECT (BIT #2)
000010                          BAI     = 10                    ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
000040                          CLR     = 40                    ;CLEAR (BIT #5)
000100                          IR      = 100                   ;INPUT READY (BIT #6)
000200                          OR      = 200                   ;OUTPUT READY (BIT #7)
000400                          MPE     = 400                   ;MASS BUS PARITY ERROR (BIT #8)
001000                          MXF     = 1000                  ;MISSED TRANSFER ERROR (BIT #9)
002000                          PGE     = 2000                  ;PROGRAM ERROR (BIT #10)
004000                          NEM     = 4000                  ;NON EXISTENT MEMORY (BIT #11)
010000                          NED     = 10000                 ;NON EXISTENT DRIVE (BIT #12)
020000                          UPE     = 20000                 ;UNIBUS PARITY ERROR (BIT #13)
040000                          WCE     = 40000                 ;WRITE CHECK ERROR (BIT #14)
100000                          DLT     = 100000                ;DATA LATE (BIT #15)

                                ;DATA BUFFER REGISTER (RMDB)
                                ;(EACH BIT IS CALLED BY BIT NUMBER)

                                .SBTTL  RM80 REGISTERS

                                ;CONTROL AND STATUS 1 REGISTER. (#00)

000001                          GO      = 1                     ;GO BIT (BIT #0)
000002                          F1      = 2                     ;FUNCTION CODE BIT #1
000004                          F2      = 4                     ;FUNCTION CODE BIT #2
000010                          F3      = 10                    ;FUNCTION CODE BIT #3
000020                          F4      = 20                    ;FUNCTION CODE BIT #4
000040                          F5      = 40                    ;FUNCTION CODE BIT #5
004000                          DVA     = 4000                  ;DEVICE AVAILABLE (BIT #11)

                                ;DRIVE STATUS REGISTER (RMDS) (#01)

000001                          OM      = 1                     ;OFFSET MODE
000100                          VV      = 100                   ;VOLUME VALID (BIT #6)
000200                          DRY     = 200                   ;DRIVE READY (BIT #7)
000400                          DPR     = 400                   ;DRIVE PRESENT (BIT #8)
001000                          PGM     = 1000                  ;PROGRAMABLE (BIT #9)
002000                          LST     = 2000                  ;LAST SECTOR TRANSFERRED (BIT #10)
004000                          WRL     = 4000                  ;WRITE LOCK (BIT #11)
010000                          MOL     = 10000                 ;MEDIUM ON-LINE (BIT #12)
020000                          PIP     = 20000                 ;POSITIONING OPERATION IN PROGRESS (BIT #13)
040000                          ERR     = 40000                 ;COMPOSITE ERROR (BIT #14)
100000                          ATA     = 100000                ;ATTENTION ACTIVE (BIT #15)

                                ;ERROR REGISTER #01 (RMER1) (#02)

000001                          ILF     = 1                     ;ILLEGAL FUNCTION (BIT #0)
000002                          ILR     = 2                     ;ILLEGAL REGISTER (BIT #1)
000004                          RMR     = 4                     ;REGISTER MODIFICATION REFUSED (BIT #2)
000010                          PAR     = 10                    ;PARITY ERROR (BIT #3)
000020                          FER     = 20                    ;FORMAT ERROR (BIT #4)
000040                          WCF     = 40                    ;WRITE CLOCK FAIL (BIT #5)
```

RM80 REGISTERS

```
                                ECH     = 100                   ;ECC HARD ERROR (BIT #6)
                                HCE     = 200                   ;HEADER COMPARE ERROR (BIT #7)
                                HCRC    = 400                   ;HEADER CRC ERROR (BIT #8)
                                AOE     = 1000                  ;ADDRESS OVERFLOW ERROR (BIT #9)
                                IAE     = 2000                  ;INVALID ADDRESS ERROR (BIT #10)
                                WLE     = 4000                  ;WRITE LOCK ERROR (BIT #11)
                                DTE     = 10000                 ;DRIVE TIMING ERROR (BIT #12)
                                OPI     = 20000                 ;OPERATION INCOMPLETE (BIT #13)
                                UNS     = 40000                 ;DRIVE UNSAFE (BIT #14)
                                DCK     = 100000                ;DATA CHECK ERROR (BIT 15)

                                ;MAINTAINABILITY REGISTER #01 (RMMR1)(#03) - READ ONLY BITS

                                DMD     = 1                     ;DIAGNOSTIC MODE
                                LSIT    = 2
                                WD      = 10
                                EECC    = 20
                                WC      = 40
                                CONT    = 100
                                PHA     = 200
                                PDA     = 400
                                ECRC    = 1000
                                PLFS    = 2000
                                ESRC    = 4000
                                REX     = 10000
                                EBL     = 20000
                                OCC     = 100000

                                ;MAINTAINABILITY REGISTER #01 (RMMR1) (#03) - WRITE ONLY BITS

                                DMD     = 1                     ;DIAGNOSTIC MODE BIT
                                MSC     = 2
                                MI      = 4
                                MWP     = 10
                                DTG     = 20
                                MS      = 40
                                MDF     = 100
                                MSER    = 200
                                MOC     = 400
                                MUR     = 1000
                                MRD     = 2000
                                MCLK    = 4000
                                MSEN    = 10000
                                DTO     = 20000
                                OBEN    = 40000
                                OBCK    = 100000

                                ;ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)

                                AT0     = 1                     ;DEVICE 0 (BIT #0)
                                AT1     = 2                     ;DEVICE 1 (BIT #1)
                                AT2     = 4                     ;DEVICE 2 (BIT #2)
                                AT3     = 10                    ;DEVICE 3 (BIT #3)
                                AT4     = 20                    ;DEVICE 4 (BIT #4)
                                AT5     = 40                    ;DEVICE 5 (BIT #5)
                                AT6     = 100                   ;DEVICE 6 (BIT #6)
                                AT7     = 200                   ;DEVICE 7 (BIT #7)
```

RM80 REGISTERS

```
                                ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)
                                ;(EACH BIT IS CALLED BY BIT NUMBER)

                                ;DRIVE TYPE REGISTER (RMDT) (#06)

000001                          DT00    = 1                     ;DRIVE TYPE NUMBER BIT 1
000002                          DT01    = 2                     ;DRIVE TYPE NUMBER BIT 2
000004                          DT02    = 4                     ;DRIVE TYPE NUMBER BIT 3
000010                          DT03    = 10                    ;DRIVE TYPE NUMBER BIT 4
000020                          DT04    = 20                    ;DRIVE TYPE NUMBER BIT 5
000040                          DT05    = 40                    ;DRIVE TYPE NUMBER BIT 6
000100                          DT06    = 100                   ;DRIVE TYPE NUMBER BIT 7
000200                          DT07    = 200                   ;DRIVE TYPE NUMBER BIT 8
000400                          DT08    = 400                   ;DRIVE TYPE NUMBER BIT 9
004000                          DRQ     = 4000                  ;DRIVE REQUEST REQUIRED (BIT #11)
020000                          MOH     = 20000                 ;MOVING HEAD (BIT #13)
040000                          TAP     = 40000                 ;TAPE DRIVE (BIT #14)
100000                          NBA     = 100000                ;NOT BLOCK ADDRESSED (BIT #15)

                                ;LOOK-AHEAD REGISTER (RMLA) (#07)

000100                          SC0     = 100                   ;SECTOR COUNT FIELD 0 (BIT #6)
000200                          SC1     = 200                   ;SECTOR COUNT FIELD 1 (BIT #7)
000400                          SC2     = 400                   ;SECTOR COUNT FIELD 2 (BIT #8)

                                ;RM MAINTAINABILITY REGISTER #2 (RMMR2) (#10)

                                ;RM ERROR REGISTER #02 (RMMR2) (#10)

                                ;OFFSET REGISTER (RMOF) (#11)

000200                          OFD     = 200                   ;OFFSET DIRECTION (BIT #07)
001000                          SSEI    = 1000                  ;SKIP SECTOR INHIBIT (BIT #09)
002000                          HCI     = 2000                  ;HEADER COMPARE INHIBIT (BIT #10)
004000                          ECI     = 4000                  ;ERROR CORRECTION CODE INHIBIT (BIT #11)
010000                          FMT16   = 10000                 ;FORMAT BIT (BIT #12)

                                ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
                                ;(EACH BIT IS CALLED BY BIT NUMBER)

                                ;CURRENT CYLINDER ADDRESS (RMHR) (#13)
                                ;(EACH BIT IS CALLED BY BIT NUMBER)

                                ;SERIAL NUMBER REGISTER (RMSN) (#14)
                                ;(EACH IS CALLED BY BIT NUMBER)

                                ;RM ERROR REGISTER #02 (RMER2) (#15)

000010                          DPE     = 10                    ;DATA PARITY ERROR (BIT #3)
000040                          SSE     = 40                    ;SKIP SECTOR ERROR (BIT #5)
000200                          DVC     = 200                   ;DEVICE CHECK (BIT #7)
002000                          LBC     = 2000                  ;LOSS OF BIT CLOCK (BIT #10)
004000                          LSC     = 4000                  ;LOSS OF SYSTEM CLOCK (BIT #11)
010000                          IVC     = 10000                 ;INVLAID COMMAND ERROR (BIT #12)
020000                          OPE     = 20000                 ;OPERATOR PLUG ERROR (BIT #13)
040000                          SKI     = 40000                 ;SEEK INCOMPLETE (BIT #14)
```

```
                100000          BSE     = 100000                ;BAD SECTOR ERROR (BIT #15)

                                ;ECC POSITION REGISTER (RMEC1) (#16)
                                ;(EACH BIT IS CALLED BY BIT NUMBER)

                                ;ECC PATTERN REGISTER (RMEC2) (#17)
                                ;(EACH BIT IS CALLED BY BIT NUMBER)

                                ;;*****************************************************************

                                ;DRIVER OPERATION CODE DEFINITIONS
                000105          SEEK    = 105
                000107          RECAL   = 107
                000111          DRVCLR  = 111
                000113          RELEASE = 113
                000115          OFFSET  = 115
                000117          RTC     = 117
                000121          READIN  = 121
                000123          PACK    = 123
                000131          SEARCH  = 131
                000151          WRCKD   = 151
                000153          WRCKHD  = 153
                000161          WRITE   = 161
                000163          WRTHD   = 163
                000171          READ    = 171
                000173          READHD  = 173
                000141          GETREG  = 141
                000143          SETFORM = 143
                000145          SELDRV  = 145

                                ;OTHER EQUATES

                177400          SCTRWC  = -256.         ;WORD COUNT FOR SECTOR
```

```
                                .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  003270        JMP     @#START1                ;;JUMP TO STARTING ADDRESS OF PROGRAM
2
3 000204  000137  003312        JMP     @#START2        ;SELECT OPERATING PARAMETERS
4
5 000210  000137  003260        JMP     @#START3        ;SELECT RH/RM ADDRESSES
6
7 000214  000137  003302        JMP     @#START4        ;COMBINATION OF 204 AND 210
8
9                               .SBTTL  ACT11 HOOKS

                                ;;******************************************************************
                                ;HOOKS REQUIRED BY ACT11
                        000220          $SVPC=.                 ;SAVE PC
                        000046          .=46
  000046  015366                        $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
                        000052          .=52
  000052  020000                        .WORD   20000           ;;2)SET LOC.52 TO 20000
                        000220          .=$SVPC                 ;; RESTORE PC
10
11                      001100  .=1100
12                              .SBTTL  APT PARAMETER BLOCK

                                ;;******************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;******************************************************************
                        001100          .$X=.   ;;SAVE CURRENT LOCATION
                        000024          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
  000024  000200                        200     ;;FOR APT START UP
                        000044          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
  000044  001100                        $APTHDR ;;POINT TO APT HEADER BLOCK
                        001100          .=.$X   ;;RESET LOCATION COUNTER
                                ;;******************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

  001100                        $APTHD:
  001100  000000        $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
  001102  001234        $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
  001104  001034        $TSTM:  .WORD   540.    ;;RUN TIM OF LONGEST TEST
  001106  001034        $PASTM: .WORD   540.    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
  001110  001034        $UNITM: .WORD   540.    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
  001112  000030                .WORD   $ETEND-$MAIL/2  ;;LENGTH MAILBOX-ETABLE(WORDS)
13        001114        TAB.XY=.        ;COMMON TAG STARTING ADDRESS
14
```

```
      0                                 .SBTTL  COMMON TAGS

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

                001114                          .=TAB.XY
001114                          $CMTAG:                         ;;START OF COMMON TAGS
001114  000000                          .WORD   0
001116     000                  $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001117     000                  $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001120  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001122  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001124  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001126  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001130     000                  $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001131     001                  $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001132  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001134  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001136  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001140  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001142  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001144  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
001146  000000                          .WORD   0
001150     000                  $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001151     000                  $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001152  000000                          .WORD   0
001154  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001156  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001160  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001162  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001164  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001166  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001170     000                  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001171     002                  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001172     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
001173     000                  $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001174  000000                  $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                                ;;WHICH  ($REG0) WAS OBTAINED
001176  000000                  $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001200  000000                  $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001202  000000                  $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001204  000000                  $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001206  000000                  $REG4:  .WORD   0               ;;CONTAINS (($REGAD)+10)
001210  000000                  $REG5:  .WORD   0               ;;CONTAINS (($REGAD)+12)
001212  000000                  $TMP0:  .WORD   0               ;;USER DEFINED
001214  000000                  $TMP1:  .WORD   0               ;;USER DEFINED
001216  000000                  $TMP2:  .WORD   0               ;;USER DEFINED
001220  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001222  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001224     207     377     377  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001230     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
001231     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001232     012     000          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                        ;;**********************************************************************
                                        .SBTTL  APT MAILBOX-ETABLE
```

```
                                ;;*****************************************************************
                                .EVEN
001234                          $MAIL:                          ;;APT MAILBOX
001234  000000                  $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001236  000000                  $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001240  000000                  $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001242  000000                  $PASS:  .WORD   APASS           ;;PASS COUNT
001244  000000                  $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
001246  000000                  $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
001250  000000                  $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
001252  000000                  $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
001254                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
001254     000                  $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
001255     000                  $ENVM:  .BYTE   AENVM           ;;ENVIRONMENT MODE BITS
001256  000000                  $SWREG: .WORD   ASWREG          ;;APT SWITCH REGISTER
001260  000000                  $USWR:  .WORD   AUSWR           ;;USER SWITCHES
001262  000000                  $CPUOP: .WORD   ACPUOP          ;;CPU TYPE,OPTIONS
                                ;*                              BITS 15-11=CPU TYPE
                                ;*                                      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                ;*                                      11/70=06,PDQ=07,Q=10
                                ;*                              BIT 10=REAL TIME CLOCK
                                ;*                              BIT  9=FLOATING POINT PROCESSOR
                                ;*                              BIT  8=MEMORY MANAGEMENT
001264     000                  $MAMS1: .BYTE   AMAMS1          ;;HIGH ADDRESS,M.S. BYTE
001265     000                  $MTYP1: .BYTE   AMTYP1          ;;MEM. TYPE,BLK#1
                                ;*                              MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                                      900 NSEC CORE=001
                                ;*                                      300 NSEC BIPOLAR=002
                                ;*                                      500 NSEC MOS=003
001266  000000                  $MADR1: .WORD   AMADR1          ;;HIGH ADDRESS,BLK#1
                                ;*                                MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE'' ABOVE
001270     000                  $MAMS2: .BYTE   AMAMS2          ;;HIGH ADDRESS,M.S. BYTE
001271     000                  $MTYP2: .BYTE   AMTYP2          ;;MEM.TYPE,BLK#2
001272  000000                  $MADR2: .WORD   AMADR2          ;;MEM.LAST ADDRESS,BLK#2
001274     000                  $MAMS3: .BYTE   AMAMS3          ;;HIGH ADDRESS,M.S.BYTE
001275     000                  $MTYP3: .BYTE   AMTYP3          ;;MEM.TYPE,BLK#3
001276  000000                  $MADR3: .WORD   AMADR3          ;;MEM.LAST ADDRESS,BLK#3
001300     000                  $MAMS4: .BYTE   AMAMS4          ;;HIGH ADDRESS,M.S.BYTE
001301     000                  $MTYP4: .BYTE   AMTYP4          ;;MEM.TYPE,BLK#4
001302  000000                  $MADR4: .WORD   AMADR4          ;;MEM.LAST ADDRESS,BLK#4
001304  000000                  $VECT1: .WORD   AVECT1          ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001306  000000                  $VECT2: .WORD   AVECT2          ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001310  000000                  $BASE:  .WORD   ABASE           ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001312  000000                  $DEVM:  .WORD   ADEVM           ;;DEVICE MAP
001314                          $ETEND:
                                .MEXIT
```

```
                                .SBTTL  USER DEFINED TAGS

001314  000000                  C.SWR:  .WORD   0               ;CONTROL SWITCHES
001316  000031                  ERMAX:  .WORD   25.             ;MAXIMUM NUMBER OF ERRORS ALLOWED PER DRIVE
001320  000000                  SAVCSW: .WORD   0               ;PREVIOUS CONTENTS OF 'C.SWR'
001322  000000                  CNTRLC: .WORD   0               ;CONTROL "C" FLAG
001324  000000                  BUSADR: .WORD   0               ;GET RM ADDRESSES FLAG (0=NO, -1=YES)
001326  000000                  LPTAVL: .WORD   0               ;LPT AVAILABLE STATUS (0=NO,1=YES)
001330  000000                  DRVSEL: .WORD   0               ;DRIVES SELECTED FOR TESTING
001332  013377  000000          TSTNMS: .WORD   13377,0         ;DEFAULT IS RUN TESTS 0-7 & 11,12 & 14
001336  000000  000000          OPNFLG: .WORD   0,0             ;MODIFY TEST PARAMETER FLAGS
001342  000000                  CLKSTA: .WORD   0               ;CLOCK STATUS (0=NO CLOCK,+1=KW11-P,
                                                                ;AND -1=KW11-L)
001344  000020                  TICKMS: .WORD   16.             ;16 MILLISECONDS PER CLOCK TICK
001346  040432                  TICKUS: .WORD   16666.          ;16666 MIRCOSECONDS PER CLOCK TICK
001350  000000                  BYPASS: .WORD   0
001352  000000                  CHKDRV: .WORD   0               ;DRIVE UNDER TEST
001354  000000                  DRVMSK: .WORD   0               ;DRIVE MASK BIT
001356  000000                  SVSTAT: .WORD   0               ;STATUS/ERROR INDICATOR IS
                                                                ;SAVED HERE ON AN ERROR
001360  000000                  CYL.RD: .WORD   0               ;CYLINDER READ
001362  000000                  TRK.RD: .WORD   0               ;TRACK READ
001364  000000                  SEC.RD: .WORD   0               ;SECTOR READ
001366  000000                  CYL.DS: .WORD   0               ;CYLINDER DESIRED
001370  000000                  SEC.DS: .WORD   0               ;SECTOR DESIRED
001372  000000                  TRK.DS: .WORD   0               ;TRACK DESIRED
001374  000000                  LSTRK:  .WORD   0               ;CONTAINS THE LAST TRACK OF THE UNIT UNDER
                                                                ;TEST. RM80 = 13.
001376  000000                  TIM.UP: .WORD   0               ;MINIMUM TIME
001400  000000                          .WORD   0               ;NUMBER OF COUNTS BELOW MIN. LIMIT
001402  000000                          .WORD   0               ;MAXIMUM TIME
001404  000000                          .WORD   0               ;NUMBER OF COUNTS ABOVE MAX. LIMIT
001406  000000  000000                  .WORD   0,0             ;TOTAL TIME OF ALL SEEKS
001412  000000                          .WORD   0               ;NUMBER OF SEEKS PERFORMED
001414  000000                  TIM.DN: .WORD   0               ;MINIMUM TIME
001416  000000                          .WORD   0               ;NUMBER OF COUNTS BELOW MIN. LIMIT
001420  000000                          .WORD   0               ;MAXIMUM TIME
001422  000000                          .WORD   0               ;NUMBER OF COUNTS ABOVE MAX. LIMIT
001424  000000  000000                  .WORD   0,0             ;TOTAL TIME OF ALL SEEKS
001430  000000                          .WORD   0               ;NUMBER OF SEEKS PERFORMED
001432  000000                  TIM.PT: .WORD   0               ;POINTS TO TABLE OF TIMES
001434  000000                  WCEFLG: .WORD   0               ;FATAL WRITE CHECK ERROR FLAG
001436  000000                  STALL0: .WORD   0               ;VARIABLE STALL (TEST 4)
001440  000000  000000          SVADR:  .WORD   0,0             ;SAVE DISK ADDRESS
001444  000000                  SEKTMR: .WORD   0               ;SEEK TIMER
001446  000000                  SEKCNT: .WORD   0               ;SEEK COUNTER
001450  000000                  DELTA:  .WORD   0               ;TESTING RANGE FOR SERVO SETTLE DOWN TEST
001452  160400                  TRCKWC: .WORD   -<256.*31.>     ;WORD COUNT FOR A FULL TRACK IN 16 BIT MODE
001454  000012                  STALL1: .WORD   10.             ;10 MILLISECONDS STALL (TEST 0-10)
001456  000012                  STALL2: .WORD   10.             ;10 MILLISECONDS STALL
001460  011610                  STALL3: .WORD   5000.           ;5 SEC STALL
001462  000031                  MXSTAL: .WORD   25.             ;MAX. INCREMENTING STALL ALLOWED IN TEST 4
001464     020                  ERR.CT: .BYTE   16.             ;NUMBER OF ERRORS ALLOWED IN TESTS
                                                                ;BEFORE GOING TO THE NEXT TEST
001465     000                          .BYTE   0               ;RESERVED
001466  000000                  BASFLG: .WORD   0               ;FLAG FOR DETECTING BAD SECTOR
001470  000000                  XXDP:   .WORD   0               ;THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH
```

USER DEFINED TAGS

```
                                                        ;THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE
                                                        ;'XXDP' DEVICE CODE FOR THE RM80.

                                    ;THE FOLLOWING 8. LOCATIONS ARE USED IN THE AVERAGE SEEK TIME
                                    ;MEASUREMENT TEST ONLY. (TEST 15)
001472  000000  000000  000000  SKTIM:  .WORD   0,0,0,0          ;ACCUMULATED SEEK TIME
001502  000000  000000          SKCNT:  .WORD   0,0             ;NUMBER OF SEEKS
001506  000000                  SKSIZ:  .WORD   0               ;SIZE OF CURRENT SEEK CYCLE
001510  000000                  CYINC:  .WORD   0               ;CYLINDER OFFSET VALUE

001512                          ERRCN:  .BLKB   8.              ;TOTAL ERROR COUNT FOR DRIVES 0-7.

001522  176700                  RH.ADR: .WORD   176700          ;RH/RM UNIBUS ADDRESS
001524  000254  000240          RHVEC:  .WORD   254,240         ;VECTOR ADDRESS AND PRIORITY
001530  000104  000106          PKV:    .WORD   104,106         ;KW11-P VECTOR ADDRESS
001534  172540                  PKCS:   .WORD   172540          ;KW11-P CONTROL AND STATUS REG.
001536  172542                  PKB:    .WORD   172542          ;KW11-P COUNT SET BUFFER
001540  172544                  PKC:    .WORD   172544          ;KW11-P COUNTER
001542  000100  000102          LKV:    .WORD   100,102         ;KW11-L VECTOR ADDRESS
001546  177546                  LKS:    .WORD   177546          ;KW11-L STATUS REGISTER
001550  177564                  TPS:    .WORD   177564          ;TTY PRINTER STATUS
001552  177566                  TPB:    .WORD   177566          ;TTY PRINTER BUFFER
001554  177514                  LPS:    .WORD   177514          ;LINE PRINTER STATUS
001556  177516                  LPB:    .WORD   177516          ;LINE PRINTER BUFFER

                                    ;BIT TABLE
001560  000001                  BITS:   .WORD   BIT00
001562  000002                          .WORD   BIT01
001564  000004                          .WORD   BIT02
001566  000010                          .WORD   BIT03
001570  000020                          .WORD   BIT04
001572  000040                          .WORD   BIT05
001574  000100                          .WORD   BIT06
001576  000200                          .WORD   BIT07
001600  000400                          .WORD   BIT08
001602  001000                          .WORD   BIT09
001604  002000                          .WORD   BIT10
001606  004000                          .WORD   BIT11
001610  010000                          .WORD   BIT12
001612  020000                          .WORD   BIT13
001614  040000                          .WORD   BIT14
001616  100000                          .WORD   BIT15
001620  000001                          .WORD   BIT00
001622  000002                          .WORD   BIT01
001624  000004                          .WORD   BIT02
001626  000010                          .WORD   BIT03
001630  000020                          .WORD   BIT04
001632  000040                          .WORD   BIT05
001634  000100                          .WORD   BIT06
001636  000200                          .WORD   BIT07

                                    .SBTTL  TIMING LIMITS

                                    ;ROTATIONAL TEST TABLES
                                    ;60HZ TABLE
001640  037422                  111A:   .WORD   ROTATE
001642  000000                          .WORD   0
```

```
001644  003075                  .WORD   1597.          ;LO LIMIT (16.67MS - 4.3%)
001646  003277                  .WORD   1727.          ;HI LIMIT (16.67MS + 3.5%)

                        ;50HZ TABLE
001650  037422          T11B:   .WORD   ROTATE
001652  000000                  .WORD   0
001654  003075                  .WORD   1597.          ;LO LIMIT (16.67MS - 4.3%)
001656  003277                  .WORD   1727.          ;HI LIMIT (16.67MS + 3.5%)

                        ;SEEK TEST TABLES
001660  037453          T12:    .WORD   ONECYL         ;FORWARD
001662  037620                  .WORD   REV            ;REVERSE
001664  000000                  .WORD   0              ;NO LOWER LIMIT
001666  001130                  .WORD   600.           ;HI LIMIT (6.0MS)

001670  037520          T13:    .WORD   AVERGE         ;FORWARD
001672  037620                  .WORD   REV            ;REVERSE
001674  000000                  .WORD   0              ;NO LOWER LIMIT
001676  004704                  .WORD   2500.          ;HI LIMIT (25.0MS)

001700  037560          T14:    .WORD   MXSEEK         ;FORWARD
001702  037620                  .WORD   REV            ;REVERSE
001704  000000                  .WORD   0              ;NO LOWER LIMIT
001706  011610                  .WORD   5000.          ;HI LIMIT (50.0MS)


                        ;SPECS. MESSAGE TABLES FOR ROTATIONAL AND TIMING TESTS

                        ;ROTATIONAL MESSAGE AND LO/HI LIMITS
                        ;60HZ TABLE
001710  040056          SP11A:  .WORD   MSG11X
001712  003075                  .WORD   1597.          ;LO LIMIT (16.67MS - 4.3%)
001714  003277                  .WORD   1727.          ;HI LIMIT (16.67MS + 3.5%)

                        ;50HZ TABLE
001716  040056          SP11B:  .WORD   MSG11X
001720  003075                  .WORD   1597.          ;LO LIMIT (16.67MS - 4.3%)
001722  003277                  .WORD   1727.          ;HI LIMIT (16.67MS + 3.5%)

                        ;TIMING TEST MESSAGES AND LO/HI LIMITS
001724  040131          SP12:   .WORD   MSG12X
001726  000000                  .WORD   0              ;NO LOWER LIMIT
001730  001130                  .WORD   600.           ;HI LIMIT (6.0MS)

001732  040173          SP13:   .WORD   MSG13X
001734  000000                  .WORD   0              ;NO LOWER LIMIT
001736  004704                  .WORD   2500.          ;HI LIMIT (25.0MS)

001740  040235          SP14:   .WORD   MSG14X
001742  000000                  .WORD   0              ;NO LOWER LIMIT
001744  011610                  .WORD   5000.          ;HI LIMIT (50.0MS)

                        ;STATUS/ERROR MESSAGE POINTER TABLE
001746  041076          STATBL: .WORD   MSG814         ;OFFLINE OR UNSAFE DRIVE REQUESTED
001750  000000  000000  000000  .WORD   0,0,0,0        ;FILLER WORDS
001760  041140                  .WORD   MSG809         ;SOFTWARE TIMEOUT ON THIS DRIVE
001762  041177                  .WORD   MSG808         ;SOFTWARE TIMEOUT ON ANOTHER DRIVE
```

```
001764  041241                  .WORD   MSGB06          ;ERROR OCCURRED DURING I/O OPERATION
001766  041305                  .WORD   MSGB05          ;ERROR OCCURRED DURING NON-I/O OPERATION
001770  041355                  .WORD   MSGB04          ;UNSAFE OCCURRED
001772  000000                  .WORD   0               ;FILLER WORD
001774  041375                  .WORD   MSGB02          ;DRIVE HAS NOT RESPONDED TO PORT REQUEST
001776  041445                  .WORD   MSGB01          ;DRIVE HAS BECOME NONEXISTENT
```

ERROR POINTER TABLE

```
                                .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


   002000                       $ERRTB:

 4
 5                              ;*ERROR  1
 6
 7                              ;*    RH CONTROLLER INTERRUPT OCCURED (RMAS=0)
 8                              ;*    ERR PC  RMAS
 9                              ;*    $ERRPC  $REG3
10
11 002000  040311                      EM1
12 002002  041503                      DH1
13 002004  042754                      DT1
14 002006  043364                      DF1
15
16                              ;*ERROR  2
17
18                              ;*    UNEXPECTED ATTENTION OCCURRED
19                              ;*    ERR PC  DRIVE   RMAS    RMDS    RMER1     RMMR2    RMER2
20                              ;*    $ERRPC  $REG1   $REG3   RMERRS  RMERRS+2 RMERRS+4 RMERRS+4
21
22 002010  040363                      EM2
23 002012  041520                      DH2
24 002014  042760                      DT2
25 002016  043370                      DF2
26
27                              ;*ERROR  3 -- NOT USED
28
29 002020  000000                      0
30 002022  000000                      0
31 002024  000000                      0
32 002026  000000                      0
33
34                              ;*ERROR  4 -- NOT USED
35
36 002030  000000                      0
37 002032  000000                      0
38 002034  000000                      0
39 002036  000000                      0
40
41                              ;*ERROR  5
42
43                              ;*    ADDRESS PLUG CHANGE BIT SET
44                              ;*    ERR PC  DRIVE   RMAS    RMDS    RMER1     RMMR2    RMER2
45                              ;*    $ERRPC  $REG1   $REG3   RMERRS  RMERRS+2  RMERRS+4 RMERRS+6
```

ERROR POINTER TABLE

```
                                                EM5
002040  040421                                  DH2
002042  041520                                  DT2
002044  042760                                  DF2
002046  043370
                                        ;*ERROR  6 -- NOT USED
                                                0
002050  000000                                  0
002052  000000                                  0
002054  000000                                  0
002056  000000
                                        ;*ERROR  7 -- NOT USED
                                                0
002060  000000                                  0
002062  000000                                  0
002064  000000                                  0
002066  000000
                                        ;*ERROR  10
                                        ;*      RH CONTROLLER FAILED TO RESPOND TO ADDRESSING
                                        ;*      RMCS1   ERR PC
                                        ;*      RH.ADR  $ERRPC
                                                EM10
002070  040455                                 DH10
002072  041605                                 DT10
002074  043016                                 DF10
002076  043404
                                        ;*ERROR  11
                                        ;*      DRIVE SELECTED IS NOT ONLINE
                                        ;*      DRIVE   ERR PC
                                        ;*      $REG2   $ERRPC
                                                EM11
002100  040533                                 DH11
002102  041624                                 DT11
002104  043022                                 DF11
002106  043410
                                        ;*ERROR  12
                                        ;*      IMPROPER HEADER DATA
                                        ;*      TEST    ERR PC  TST PC  DRIVE   CYLNDR  TRACK   SECTOR
                                        ;*      $TMP0   $ERRPC  $REG0   CHKDRV  CYL.DS  TRK.DS  SEC.DS
                                        ;*          GDCYL   GDTRK   GDSCTR  BDCYL   BDTRK   BDSCTR
                                        ;*          CYL.DS  TRK.DS  SEC.DS  CYL.RD  TRK.RD  SEC.RD
                                        ;*      CYLNDR, TRACK, AND SECTOR ARE DECIMAL
                                                EM12
002110  040570                                 DH12
002112  041643                                 DT12
002114  043026                                 DF12
002116  043414
                                        ;*ERROR  13
```

ERROR POINTER TABLE

```
                                        ;*    DATA COMPARE FAILURE
                                        ;*    TEST    ERR PC  TST PC  DRIVE   CYLNDR  TRACK   SECTOR
                                        ;*    $TMP0   $ERRPC  $REG0   CHKDRV  CYL.DS  TRK.DS  SEC.DS
                                        ;*        GDDAT   BDDAT   WRDCNT  GDADR   BDADR
                                        ;*        $GDDAT  $BDDAT  $REG4   $GDADR  $BDADR
                                        ;*    CYLNDR, TRACK, SECTOR, AND WRDCNT  ARE DECIMAL

002120  040615                            EM13
002122  041643                            DH12
002124  043060                            DT13
002126  043424                            DF13

                                        ;*ERROR  14 -- FOLLOWS #13

                                        ;*        $GDDAT  $BDDAT  $REG4   $GDADR  $BDADR

002130  000000                            0
002132  000000                            0
002134  043076                            DT13A
002136  043434                            DF14

                                        ;*ERROR  15

                                        ;*    DATA COMPARE FAILURE
                                        ;*    TEST    ERR PC  TST PC  DRIVE   CYLNDR  TRACK   SECTOR
                                        ;*    $TMP0   $ERRPC  $REG0   CHKDRV  CYL.DS  TRK.DS  SEC.DS
                                        ;*        GDDAT   BDDAT   WRDCNT  GDADR   BDADR
                                        ;*        $GDDAT  $BDDAT  $REG4   $GDADR  $BDADR
                                        ;*    CYLNDR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL

002140  040615                            EM13
002142  041643                            DH12
002144  043060                            DT13
002146  043424                            DF13

                                        ;*ERROR  16 -- FOLLOWS #15

                                        ;*  $GDDAT  $BDDAT  $REG4   $GDADR  $BDADR

002150  000000                            0
002152  000000                            0
002154  043076                            DT13A
002156  043434                            DF14

                                        ;*ERROR  17

                                        ;*    DISK ERROR IN TIMING TEST
                                        ;*    TEST    ERR PC  DRIVE   RMCS1   RMDS     RMER1     RMMR2     RMER2
                                        ;*    $TMP0   $ERRPC  CHKDRV  RM.REG  RM.REG+12 RM.REG+14 RM.REG+40 RM.REG+42

002160  040642                            EM17
002162  042057                            DH17
002164  043110                            DT17
002166  043440                            DF17

                                        ;*ERROR  20
```

ERROR POINTER T'BLE

```
                                ;*    CLOCK (KW11-P) OVERFLOW IN TIMING TEST
                                ;*    TEST    ERR PC  DRIVE   RMCS1   RMDS     RMER1     RMMR2     RMER2
                                ;*    $TMP0   $ERRPC  CHKDRV  RM.REG  RM.REG+12 RM.REG+14 RM.REG+40 RM.REG+42

002170  040674                    EM20
002172  042057                    DH17
002174  043110                    DT17
002176  043440                    DF17

                                ;*ERROR  21

                                ;*    DATA COMPARE FAILURE
                                ;*    TEST      ERR PC  TST PC  DRIVE   CYLNDR  TRACK
                                ;*    $TMP0     $ERRPC  $REG0   CHKDRV  CYL.DS  TRK.DS
                                ;*        GDDAT   BDDAT   WRDCNT  SECTOR
                                ;*        $REG1   $BDDAT  $REG4   $REG1
                                ;*    CYLINDR, TRACK, WRDCNT, AND SECTOR ARE DECIMAL

002200  040615                    EM13
002202  042154                    DH21
002204  043130                    DT21
002206  043444                    DF21

                                ;*ERROR  22--FOLLOWS #21

                                ;*    $REG1   $BDDAT   $REG4   $REG1

002210  000000                    0
002212  000000                    0
002214  043144                    DT21A
002216  043454                    DF22

                                ;*ERROR  23

                                ;*    DISK ERROR DURING SEEK
                                ;*    TEST      ERR PC  DRIVE   CYLNDR  RMCS1   RMCS2     RMDS
                                ;*    $TMP0     $ERRPC  CHKDRV  CYL.DS  RM.REG RM.REG+10 RM.REG+12
                                ;*        RMER1     RMMR2     RMER2     RMDC      RMHR
                                ;*        RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+34 RM.REG+36

002220  040743                    EM23
002222  042271                    DH23
002224  043154                    DT23
002226  043460                    DF23

                                ;*ERROR  24

                                ;*    SEEK NOT COMPLETE WITHIN 120 MS
                                ;*    TEST      ERR PC  DRIVE   CYLNDR  RMCS1   RMCS2     RMDS
                                ;*    $TMP0     $ERRPC  CHKDRV  CYL.DS  RM.REG RM.REG+10 RM.REG+12
                                ;*        RMER1     RMMR2     RMER2     RMDC      RMHR
                                ;*        RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+34 RM.REG+36

002230  040772                    EM24
002232  042271                    DH23
002234  043154                    DT23
```

ERROR POINTER TABLE

```
    217 002236  043460                        DF23
    218
    219                                 ;;*****************************************************************
    220                                 ;*  ERRORS 23-40 NOT USED
    221                                 ;*  ERRORS 41-46 WILL HAVE AN EM THAT
    222
    223                                 ;*  VARIES DEPENDING ON THE ERROR, IT WILL BE IN THE FORM:
    224                                 ;*  RH/RM ERROR (MESSAGE)
    225                                 ;*  WHERE MESSAGE WILL BE ONE OR MORE OF THE FOLLOWING:
    226                                 ;*  1)  OFFLINE DRIVE REQUESTED
    227                                 ;*  3)  SOFTWARE TIMEOUT ON THIS DRIVE
    228                                 ;*  4)  SOFTWARE TIMEOUT ON ANOTHER DRIVE
    229                                 ;*  5)  ERROR OCCURRED DURING I/O OPERATION
    230                                 ;*  6)  ERROR OCCURRED DURING NON-I/O OPERATION
    231                                 ;*  7)  UNSAFE OCCURED
    232                                 ;;*****************************************************************
    233
    234 002240                          ITEM41:
    235
    236                                 ;*ERROR  41
    237
    238                                 ;*    RH/RM ERROR (MESSAGE)
    239                                 ;*    TEST    ERR PC  TST PC  DRIVE
    240                                 ;*    $TMP0   $ERRPC  $REG0   CHKDRV
    241
    242 002240  041032                      EM41
    243 002242  042423                      DH41
    244 002244  043204                      DT41
    245 002246  043470                      DF41
    246
    247                                 ;*ERROR  42
    248
    249                                 ;*    RH/RM ERROR (MESSAGE)
    250                                 ;*    TEST    ERR PC  TST PC  DRIVE   RMCS1   RMCS2   RMDS
    251                                 ;*    $TMP0   $ERRPC  $REG0   CHKDRV  RM.REG  RM.REG+10 RM.REG+12
    252
    253 002250  041032                      EM41
    254 002252  042461                      DH42
    255 002254  043214                      DT42
    256 002256  043474                      DF42
    257
    258                                 ;*ERROR  43
    259
    260                                 ;*    RH/RM ERROR (MESSAGE)
    261                                 ;*    TEST    ERR PC  TST PC  DRIVE   RMCS1   RMCS2   RMDS
    262                                 ;*    $TMP0   $ERRPC  $REG0   CHKDRV  RM.REG  RM.REG+10 RM.REG+12
    263                                 ;*        RMER1     RMMR2     RMER2
    264                                 ;*        RM.REG+14 RM.REG+40 RM.REG+42
    265
    266 002260  041032                      EM41
    267 002262  042461                      DH42
    268 002264  043232                      DT43
    269 002266  043500                      DF43
    270
    271                                 ;*ERROR  44
    272
    273                                 ;*    RH/RM ERROR (MESSAGE)
```

ERROR POINTER TABLE

```
                                        ;*    TEST    ERR PC  TST PC  DRIVE   CYLNDR  TRACK     SECTOR
                                        ;*    $TMP0   $ERRPC  $REG0   CHKDRV  CYL.DS  TRK.DS    SEC.DS
                                        ;*        RMCS1     RMCS2     RMDS      RMHR     RMDC     RMDA
                                        ;*        RM.REG    RM.REG+10 RM.REG+12 RM.REG+36 RM.REG+34 RM.REG+06
                                        ;*    RMER1           RMMR2       RMER2
                                        ;*    RM.REG+14    RM.REG+40    RM.REG+42
                                        ;*    CYLNDR,TRACK, AND SECTOR ARE DECIMAL

 002270  041032                           EM41
 002272  041643                           DH12
 002274  043256                           DT44
 002276  043510                           DF44

                                       ;*ERROR  45

                                        ;*    RH/RM ERROR (MESSAGE)
                                        ;*    TEST    ERR PC  TST PC  DRIVE   CYLNDR  TRACK     SECTOR
                                        ;*    $TMP0   $ERRPC  $REG0   CHKDRV  CYL.DS  TRK.DS    SEC.DS
                                        ;*        RMCS1     RMCS2     RMDS      RMHR      RMDC      RMDA
                                        ;*        RM.REG    RM.REG+10 RM.REG+12 RM.REG+36 RM.REG+34 RM.REG+06
                                        ;*    RMER1           RMMR2       RMER2       RMWC       RMBA     RMDB
                                        ;*    RM.REG+14   RM.REG+40   RM.REG+42   RM.REG+2   RM.REG+4 RM.REG+22
                                        ;*    CYLNDR,TRACK, AND SECTOR ARE DECIMAL

 002300  041032                           EM41
 002302  041643                           DH12
 002304  043316                           DT45
 002306  043524                           DF45

                                       ;*ERROR  46

                                        ;*    FATAL WRITE CHECK ERROR (MESSAGE)
                                        ;*    TEST    ERR PC  TST PC  DRIVE   CYLNDR  TRACK     SECTOR
                                        ;*    $TMP0   $ERRPC  $REG0   CHKDRV  CYL.DS  TRK.DS    SEC.DS
                                        ;*        RMCS1     RMCS2     RMDS      RMHR      RMDC      RMDA
                                        ;*        RM.REG    RM.REG+10 RM.REG+12 RM.REG+36 RM.REG+34 RM.REG+06
                                        ;*    RMER1           RMMR2       RMER2       RMWC       RMBA     RMDB
                                        ;*    RM.REG+14   RM.REG+40   RM.REG+42   RM.REG+2   RM.REG+4 RM.REG+22
                                        ;*    CYLNDR,TRACK, AND SECTOR ARE DECIMAL

 002310  041046                           EM46
 002312  041643                           DH12
 002314  043316                           DT45
 002316  043524                           DF45
```

```
                                .SBTTL  TEST PARAMETER POINTERS AND TABLES

                                ;COMMON STORAGE FOR TEST PARAMETER
002320  000000                  PRM:    .WORD   0               ;THIS WORD TELLS WHICH OF THE
                                                                ;FOLLOWING PARAMETERS ARE TO BE USED
002322  000000                  RPT:    .WORD   0               ;REPEAT COUNTS FOR ALL TESTS
002324  000000                  FC:     .WORD   0               ;FIRST CYLINDER
002326  000000                  LC:     .WORD   0               ;LAST CYLINDER
002330  000000                  IC:     .WORD   0               ;INCREMENT CYLINDER
002332  000000                  FT:     .WORD   0               ;FIRST TRACK
002334  000000                  LT:     .WORD   0               ;LAST TRACK
002336  000000                  IT:     .WORD   0               ;INCREMENT TRACK
002340  000000                  FS:     .WORD   0               ;FIRST SECTOR
002342  000000                  LS:     .WORD   0               ;LAST SECTOR
002344  000000                  PAT:    .WORD   0               ;PATTERN CODE

002346  000000                  NC1:    .WORD   0               ;NEW CYLINDER ADDRESS
002350  000000                  NC2:    .WORD   0               ;NEW CYLINDER ADDRESS

                                ;TABLE OF PARAMETER POINTERS
002352  002726                  PRMPT:  .WORD   PRM0
002354  002742                          .WORD   PRM1
002356  002764                          .WORD   PRM2
002360  003002                          .WORD   PRM3
002362  003020                          .WORD   PRM4
002364  003036                          .WORD   PRM5
002366  003054                          .WORD   PRM6
002370  003072                          .WORD   PRM7
002372  003106                          .WORD   PRM10
002374  003124                          .WORD   PRM11
002376  003136                          .WORD   PRM12
002400  003146                          .WORD   PRM13
002402  003156                          .WORD   PRM14
002404  003166                          .WORD   PRM15
002406  000000                          .WORD   0               ;TERMINATOR

                                ;TABLE OF PARAMETER UPPER LIMITS
002410  077777                  PRMLMT: .WORD   32767.          ;'R'
002412  001060                          .WORD   560.            ;'FC'
002414  001060                          .WORD   560.            ;'LC'
002416  001060                          .WORD   560.            ;'IC'
002420  000015                          .WORD   13.             ;'FT'
002422  000015                          .WORD   13.             ;'LT'
002424  000015                          .WORD   13.             ;'IT'
002426  000036                          .WORD   30.             ;'FS'
002430  000036                          .WORD   30.             ;'LS'
002432  177777                          .WORD   177777          ;'PAT'

                                ;TABLE OF MESSAGE POINTERS
002434  036710                  PRMMSG: .WORD   MSG.R
002436  036712                          .WORD   MSG.FC
002440  036715                          .WORD   MSG.LC
002442  036720                          .WORD   MSG.IC
002444  036723                          .WORD   MSG.FT
002446  036726                          .WORD   MSG.LT
002450  036731                          .WORD   MSG.IT
002452  036734                          .WORD   MSG.FS
```

```
     48 002454  036737                          .WORD   MSG.LS
     49
     50                                 ;STATUS/ERROR INDICATOR MESSAGES POINTER TABLE
     51                                 ;DEFAULT VALUES OF TEST PARAMETERS
     52
     53 002456  000227  000310  000000  DFLT:   .WORD   227.200.,0.560.,0.0      ;RECAL/RANDOM SEEK (T0)
     54 002472  000677  000144  000000          .WORD   677.100.,0.256.,0.0.0.0.0       ;SEEK/SEEK (T1)
     55 002514  000237  000001  000000          .WORD   237.1.0.560.,1.0.0      ;INCREMENT SEEK (T2)
     56 002532  000237  000010  000000          .WORD   237.10.0.512.,1.0.0     ;STEPPING SEEK (T3)
     57 002550  000237  000001  000000          .WORD   237.1.0.560.,1.0.0      ;OSCILLATING SEEK (T4)
     58 002566  000237  000001  000000          .WORD   237.1.0.560.,1.0.0      ;CONVERGING/DIVERGING SEEK (T5)
     59 002604  000237  000001  000000          .WORD   237.1.0.560.,1.0.0      ;SERVO ADDRESSING LOGIC NOISE (T6)
     60 002622  000067  011610  000000          .WORD   67.5000.,0.560.,0.13.   ;RANDOM SEEK TEST (T7)
     61 002636  000237  000001  000000          .WORD   237.1.0.560.,1.0.0      ;ALL SEEKS TEST (T10)
     62 002654  000223  000001  000000          .WORD   223.1.0.0.0             ;ROTATIONAL SPEED TIMING TEST (T11)
     63 002666  000007  000001  000000          .WORD   7.1.0.560.              ;ONE CYLINDER SEEK TIMING TEST (T12)
     64 002676  000007  000001  000000          .WORD   7.1.0.155.              ;AVERAGE SEEK TIMING TEST (T13)
     65 002706  000007  000001  000000          .WORD   7.1.0.560.              ;MAXIMUM SEEK TIMING TEST (T14)
     66 002716  000007  000001  000000          .WORD   7.1.0.560.              ;AVG. SEEK TIME MEASUREMENT TEST (T15)
     67
     68                                 ;PARAMETER TABLES
     69
     70                                 ;RECAL/RANDOM SEEK (T0)
     71 002726  000227                  PRM0:   .WORD   227
     72 002730  000310                          .WORD   200.
     73 002732  000000                          .WORD   0
     74 002734  001060                          .WORD   560.
     75 002736  000000                          .WORD   0
     76 002740  000000                          .WORD   0
     77
     78                                 ;SEEK/SEEK (T1)
     79 002742  000677                  PRM1:   .WORD   677
     80 002744  000144                          .WORD   100.
     81 002746  000000                          .WORD   0
     82 002750  000400                          .WORD   256.
     83 002752  000000                          .WORD   0
     84 002754  000000                          .WORD   0
     85 002756  000000                          .WORD   0
     86 002760  000000                          .WORD   0
     87 002762  000000                          .WORD   0
     88
     89                                 ;INCREMENT SEEK (T2)
     90 002764  000237                  PRM2:   .WORD   237
     91 002766  000001                          .WORD   1
     92 002770  000000                          .WORD   0
     93 002772  001060                          .WORD   560.
     94 002774  000001                          .WORD   1
     95 002776  000000                          .WORD   0
     96 003000  000000                          .WORD   0
     97
     98                                 ;STEPPING SEEK (T3)
     99 003002  000237                  PRM3:   .WORD   237
    100 003004  000001                          .WORD   1
    101 003006  000000                          .WORD   0
    102 003010  001000                          .WORD   512.
    103 003012  000001                          .WORD   1
    104 003014  000000                          .WORD   0
```

```
    105 003016  000000                        .WORD   0
    106
    107                                     ;OSCILLATING SEEK (T4)
    108 003020  000237                      PRM4:   .WORD   237
    109 003022  000001                              .WORD   1
    110 003024  000000                              .WORD   0
    111 003026  001060                              .WORD   560.
    112 003030  000001                              .WORD   1
    113 003032  000000                              .WORD   0
    114 003034  000000                              .WORD   0
    115
    116                                     ;CONVERGING/DIVERGING SEEK (T5)
    117 003036  000237                      PRM5:   .WORD   237
    118 003040  000001                              .WORD   1
    119 003042  000000                              .WORD   0
    120 003044  001060                              .WORD   560.
    121 003046  000001                              .WORD   1
    122 003050  000000                              .WORD   0
    123 003052  000000                              .WORD   0
    124
    125                                     ;SERVO ADDRESSING LOGIC NOISE GENERATOR (T6)
    126 003054  000237                      PRM6:   .WORD   237
    127 003056  000001                              .WORD   1
    128 003060  000000                              .WORD   0
    129 003062  001060                              .WORD   560.
    130 003064  000001                              .WORD   1
    131 003066  000000                              .WORD   0
    132 003070  000000                              .WORD   0
    133
    134                                     ;RANDOM SEEK TEST (T7)
    135 003072  000067                      PRM7:   .WORD   67
    136 003074  011610                              .WORD   5000.
    137 003076  000000                              .WORD   0
    138 003100  001060                              .WORD   560.
    139 003102  000000                              .WORD   0
    140 003104  000015                              .WORD   13.
    141
    142                                     ;ALL SEEKS TEST (T10)
    143 003106  000237                      PRM10:  .WORD   237
    144 003110  000001                              .WORD   1
    145 003112  000000                              .WORD   0
    146 003114  001060                              .WORD   560.
    147 003116  000001                              .WORD   1
    148 003120  000000                              .WORD   0
    149 003122  000000                              .WORD   0
    150
    151                                     ;ROTATIONAL SPEED TIMING TEST (T11)
    152 003124  000223                      PRM11:  .WORD   223
    153 003126  000001                              .WORD   1
    154 003130  000000                              .WORD   0
    155 003132  000000                              .WORD   0
    156 003134  000000                              .WORD   0
    157
    158                                     ;ONE CYLINDER SEEK TIMING TEST (T12)
    159 003136  000007                      PRM12:  .WORD   7
    160 003140  000001                              .WORD   1
    161 003142  000000                              .WORD   0
```

```
    162 003144  001060                        .WORD   560.
    163
    164                                 ;AVERAGE SEEK TIMING TEST (T13)
    165 003146  000007                  PRM13:  .WORD   7
    166 003150  000001                          .WORD   1
    167 003152  000000                          .WORD   0
    168 003154  000233                          .WORD   155.
    169
    170                                 ;MAXIMUM SEEK TIMING TEST (T14)
    171 003156  000007                  PRM14:  .WORD   7
    172 003160  000001                          .WORD   1
    173 003162  000000                          .WORD   0
    174 003164  001060                          .WORD   560.
    175
    176                                 ;AVERAGE SEEK TIME MEASUREMENT TEST (T15)
    177 003166  000007                  PRM15:  .WORD   7
    178 003170  000001                          .WORD   1
    179 003172  000000                          .WORD   0
    180 003174  001060                          .WORD   560.
    181
    465
    473
    481
    486
    511
    529
    536
    540
    549
    553
    572
    581
    587
    588
    598
    608
```

```
                                        ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS

003176  011600                BADTMO: MOV     (SP),R0         ;SAVE PC WHERE THE TIME OUT OCCURED
003200  005740                        TST     -(R0)           ;ADJUST PC -2
003202  022626                        CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
003204  104401  003212                TYPE    ,65$            ;;TYPE ASCIZ STRING
003210  000417                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
003250                          64$:
003250  010046                        MOV     R0,-(SP)        ;SETUP FOR TYPING OUT PC
003252  104402                        TYPOC
003254  000240                        NOP                     ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
                                                                ;TO STOP ON UNEXPECTED TIMEOUT.
003256  000404                        BR      START1          ;BRANCH TO START1

                                .SBTTL  START OF PROGRAM

003260  012737  177777  001324  START3: MOV     #-1,BUSADR      ;GET BUSADR FLAG
003266  000402                        BR      STRT1A

003270  005037  001324          START1: CLR     BUSADR          ;CLR BUSADR FLAG
003274  005037  001322          STRT1A: CLR     CNTRLC          ;NO CONTROL "C"
003300  000411                        BR      START

003302  012737  177777  001324  START4: MOV     #-1,BUSADR      ;SET BUSADR FLAG
003310  000402                        BR      STRT2A

003312  005037  001324          START2: CLR     BUSADR          ;CLR BUSADR FLAG
003316  012737  177777  001322  STRT2A: MOV     #-1,CNTRLC      ;SET CONTROL "C" FLAG

003324  005227  000000          START:  INC     #0              ;TTY LOOP, WAIT FOR INCREMENT
003330  001375                        BNE     .-4             ;OF WORD
003332  000005                        RESET                   ;CLEAR THE WORLD

                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
003334  012706  001114                MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
003340  005026                        CLR     (R6)+           ;;CLEAR MEMORY LOCATION
003342  022706  001154                CMP     #SWR,R6         ;;DONE?
003346  001374                        BNE     .-6             ;;LOOP BACK IF NO
003350  012706  001100                MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
003354  012737  021432  000020        MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
003362  012737  000340  000022        MOV     #340,@#IOTVEC+2 ;;LEVEL 7
003370  012737  015674  000030        MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
003376  012737  000340  000032        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
003404  012737  022126  000034        MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
003412  012737  000340  000036        MOV     #340,@#TRAPVEC+2;LEVEL 7
003420  013737  015322  015314        MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
003426  012737  176543  022600        MOV     #176543,$HINUM  ;;PRIME THE RANDOM NUMBER GENERATOR
003434  012737  123456  022602        MOV     #123456,$LONUM  ;;BOTH HIGH AND LOW WORDS
003442  005037  001220                CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
003446  005037  001222                CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
003452  112737  000001  001131        MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
003460  012737  003460  001122        MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
003466  012737  003466  001124        MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
```

```
INITIALIZE THE COMMON TAGS

                                        ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
003474  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
003500  012737  003534  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
003506  012737  177570  001154          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
003514  012737  177570  001156          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
003522  022777  177777  175424          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
003530  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
003532  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
003534  012716  003542          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
003540  000002                          RTI
003542  012737  000176  001154  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
003550  012737  000174  001156          MOV     #DISPREG,DISPLAY
003556  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

003562  005037  001242                  CLR     $FASS           ;;CLEAR PASS COUNT
003566  132737  000200  001255          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
003574  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
003576  012737  001256  001154          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
003604                          67$:
   33                           ;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
   34 003604  012737  003176  000004          MOV     #BADTMO,ERRVEC  ;SETUP FOR UNEXPECTED TIMEOUT
   35 003612  012737  000300  000006          MOV     #PR6,ERRVEC+2   ;LEVEL 6
   36
   37                           .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
003620  005227  177777                  INC     #-1             ;;FIRST TIME?
003624  001033                          BNE     68$             ;;BRANCH IF NO
003626  022737  015366  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
003634  001427                          BEQ     68$             ;;BRANCH IF YES
003636  104401  003644                  TYPE    ,69$            ;;TYPE ASCIZ STRING
003642  000424                          BR      68$             ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  <CRLF>@CZRNGA0 - RM80 FUNCTIONAL TEST, PT 4@<CRLF>
003714                          68$:
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
003714  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
003720  001012                          BNE     70$             ;;BRANCH IF YES
003722  123727  001254  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
003730  001406                          BEQ     70$             ;;BRANCH IF YES
003732  023727  001154  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
003740  001005                          BNE     71$             ;;BRANCH IF NO
003742  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
003744  000403                          BR      71$
003746  112737  000001  001150  70$.    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
003754                          71$:
   38
   39 003754  012700  001174                  MOV     #$REGAD,R0      ;FIRST ADDRESS
   40 003760  005020                  1$:     CLR     (R0)+           ;CLEAR VARIABLE STORAGE
   41 003762  022700  001224                  CMP     #$BELL,R0       ;DONE?
   42 003766  001374                          BNE     1$              ;NO--BRANCH
   43 003770  013737  001550  001164          MOV     TPS,$TPS        ;SETUP THE STATUS AND BUFFER REG'S
   44 003776  013737  001552  001166          MOV     TPB,$TPB        ;FOR THE TYPE ROUTINE
   45
   46                           ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
   47                           ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
   48
   49 004004  005037  001470                  CLR     XXDP            ;CLEAR 'XXDP' LOAD DEVICE STORAGE
```

GET VALUE FOR SOFTWARE SWITCH REGISTER

```
     50 004010  122737  000016  000041          CMPB    #16,@#41        ;LOADED FROM AN RM80 ?
     51 004016  001121                          BNE     4$              ;BR IF NOT
     52 004020  013737  000040  001470          MOV     @#40,XXDP       ;GET DEVICE INDICATOR AND NUMBER
     53 004026  122737  000007  001470          CMPB    #7,XXDP         ;IS IT A VALID NUMBER ?
     54 004034  103002                          BHIS    2$              ;YES
     55 004036  105037  001470                  CLRB    XXDP            ;NO, DEFAULT TO DRIVE 0
     56 004042  005737  000042          2$:     TST     @#42            ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
     57 004046  001425                          BEQ     3$              ;BR IF NEITHER
     58 004050  104401  004056                  TYPE    ,73$            ;;TYPE ASCIZ STRING
        004054  000412                          BR      72$             ;;GET OVER THE ASCIZ
                                        ;;73$:  .ASCIZ  <CRLF>/NOT TESTING DRIVE /
        004102                          72$:
     59 004102  005046                          CLR     -(SP)           ;CLEAR WORD ON STACK
     60 004104  113716  001470                  MOVB    XXDP,(SP)       ;GET DRIVE ADDRESS
     61 004110  104403                          TYPOS                   ;TYPE THE ADDRESS
     62 004112     001                          .BYTE   1               ;ONLY 1 CHARACTER
     63 004113     000                          .BYTE   0               ;SUPRESS LEADING ZEROS
     64 004114  104401  001231                  TYPE    ,$CRLF          ;CR-LF
     65 004120  000460                          BR      4$              ;GET NUMBER OF DRIVES
     66
     67 004122  005227  177777          3$:     INC     #-1             ;FIRST TIME THRU HERE ?
     68 004126  001055                          BNE     4$              ;NO
     69 004130  104401  004136                  TYPE    ,75$            ;;TYPE ASCIZ STRING
        004134  000410                          BR      74$             ;;GET OVER THE ASCIZ
                                        ;;75$:  .ASCIZ  <CRLF>/TO TEST DRIVE /
        004156                          74$:
     70 004156  005046                          CLR     -(SP)           ;CLEAR WORD ON STACK
     71 004160  113716  001470                  MOVB    XXDP,(SP)       ;GET DRIVE ADDRESS
     72 004164  104403                          TYPOS                   ;TYPE DRIVE ADDRESS
     73 004166     001                          .BYTE   1               ;ONLY 1 CHARACTER
     74 004167     000                          .BYTE   0               ;SUPRESS LEADING ZEROS
     75 004170  104401  004176                  TYPE    ,76$            ;;TYPE ASCIZ STRING
        004174  000432                          BR      4$              ;;GET OVER THE ASCIZ
                                        ;;76$:  .ASCIZ  /, HALT PROGRAM, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>
        004262                          4$:
     79 004262  004737  020134                  JSR     PC,$TKINT       ;TURN ON THE TTY KEYBOARD INTERRUPT
     80 004266  005227  177777                  INC     #-1             ;SEE IF FIRST START
     81 004272  001002                          BNE     SRTINT          ;BR IF NOT
     82 004274  004737  036276                  JSR     PC,GETADR       ;GET OR CHECK THE RH/RM ADDRESS
     83
     84 004300  105037  001512          SRTINT: CLRB    ERRCN           ;CLEAR DRV 0 ERROR COUNT
     87 004304  105037  001513                  CLRB    ERRCN+1         ;CLEAR DRV 1 ERROR COUNT
        004310  105037  001514                  CLRB    ERRCN+2         ;CLEAR DRV 2 ERROR COUNT
        004314  105037  001515                  CLRB    ERRCN+3         ;CLEAR DRV 3 ERROR COUNT
        004320  105037  001516                  CLRB    ERRCN+4         ;CLEAR DRV 4 ERROR COUNT
        004324  105037  001517                  CLRB    ERRCN+5         ;CLEAR DRV 5 ERROR COUNT
        004330  105037  001520                  CLRB    ERRCN+6         ;CLEAR DRV 6 ERROR COUNT
        004334  105037  001521                  CLRB    ERRCN+7         ;CLEAR DRV 7 ERROR COUNT
     88 004340  004737  023176                  JSR     PC,LP.AVL       ;CHECK FOR A LINE PRINTER
     89 004344  012737  000001  001120          MOV     #1,$ICNT        ;SET ITERATION COUNT TO 1
     90 004352  004737  026656                  JSR     PC,GETSWR       ;GO CHECK FOR CONTROL SWITCHES
     91
     92 004356  004737  023240          SETVEC: JSR     PC,ST.CLK       ;INITIALIZE THE CLOCK
     93 004362  004737  031376                  JSR     PC,RMINIT       ;CHECK THE DRIVE STATUS
     94 004366  012737  177777  031330          MOV     #-1,SAVEFG      ;SET THE SAVE REGISTERS FLAG
     95 004374  005037  177776                  CLR     PS              ;ENSURE THE PRIORITY = 0
     96 004400  005227  177777                  INC     #-1             ;FIRST TIME THRU HERE ?
```

```
     97 004404  001403                    BEQ     1$              ;BR IF NO
     98 004406  005737  001322            TST     CNTRLC          ;CONTROL 'C' SWITCH SET ?
     99 004412  001066                    BNE     SRTDRV          ;CONTINUE IF YES
    100 004414  005004              1$:   CLR     R4              ;DRIVE TABLE POINTER
    101 004416  104401  036777            TYPE    ,UNSTAT         ;TYPE 'UNIT STATUS'
    102 004422  104401  001231        2$: TYPE    ,$CRLF          ;CR-LF
    103 004426  010446                    MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
        004430  104403                    TYPOS                   ;;GO TYPE--OCTAL ASCII
        004432     002                    .BYTE   2               ;;TYPE 2 DIGIT(S)
        004433     000                    .BYTE   0               ;;SUPPRESS LEADING ZEROS
    104 004434  104401  040051            TYPE    ,BLNKS4         ;TYPE 4 BLANKS
    105 004440  105764  031242            TSTB    DRVSTA(R4)      ;CHECK DRIVE'S STATUS
    106 004444  100416                    BMI     5$              ;BR IF UNSAFE
    107 004446  001020                    BNE     6$              ;BR IF ONLINE
    108 004450  105764  031252            TSTB    DRVTYP(R4)      ;SEE IF OFFLINE OR NONEXISTENT
    109 004454  001404                    BEQ     3$              ;BR IF NONEXISTENT
    110 004456  100006                    BPL     4$              ;BR IF OFFLINE
    111 004460  104401  037063            TYPE    ,NOTRM          ;DRIVE NOT AN RM80
    112 004464  000433                    BR      11$             ;CHECK NEXT DRIVE
    113
    114 004466  104401  037036        3$: TYPE    ,NOTPRS         ;DRIVE NOT PRESENT
    115 004472  000430                    BR      11$             ;CHECK NEXT DRIVE
    116
    117 004474  104401  037015        4$: TYPE    ,UNTOFF         ;DRIVE OFFLINE
    118 004500  000416                    BR      8$              ;PRINT DRIVE TYPE
    119
    120 004502  104401  037053        5$: TYPE    ,NOTSAF         ;DRIVE UNSAFE
    121 004506  000413                    BR      8$              ;PRINT DRIVE TYPE
    122
    123 004510  005737  001470        6$: TST     XXDP            ;LOADED FROM THIS DEVICE ?
    124 004514  001406                    BEQ     7$              ;BR IF NO
    125 004516  123704  00147C            CMPB    XXDP,R4         ;LOADED FROM THIS DRIVE ?
    126 004522  001003                    BNE     7$              ;BR IF NO
    127 004524  104401  037100            TYPE    ,LODEV          ;DRIVE IS LOAD DEVICE
    128 004530  000411                    BR      11$
    129 004532  104401  037026        7$: TYPE    ,UNTON          ;DRIVE ONLINE
    130 004536  104401  040053        8$: TYPE    ,BLNKS2         ;TYPE 2 BLANKS
    135 004542  012737  037115  004552    MOV     #$RM80,10$      ;ASSUME ADDRESS OF RM80 MESSAGE
    140
    141 004550  104401                9$: TYPE                    ;TYPE THE DRIVE TYPE MESSAGE
    142 004552  000000               10$: .WORD   0               ;MESSAGE ADDRESS HERE
    143
    144 004554  005204               11$: INC     R4              ;INCREMENT DRIVE NUMBER/TABLE POINTER
    145 004556  020427  000010            CMP     R4,#8.          ;FINISHED ?
    146 004562  001317                    BNE     2$              ;BR IF NOT
    147 004564  104401  001231            TYPE    ,$CRLF          ;CR-LF
```

GET UNIT STATUS

```
                                        .SBTTL  GET UNIT STATUS

                                SRTDRV: JSR     PC,CNTCLR       ;GO CLEAR MASSBUS CONTROLLER
004570  004737  023146                  TST     CNTRLC          ;CONTROL "C" START/RESTART?
004574  005737  001322                  BEQ     2$              ;NO--BRANCH
004600  001427                          MOV     SAVCSW,-(SP)    ;GET THE PREVIOUS 'C.SWR' CONTENTS
004602  013746  001320                  ADD     C.SWR,(SP)      ;SET UP TO SEE IF 'BIT00' IS DIFFERENT
004606  063716  001314                  BIT     #BIT00,(SP)+    ;IS 'BIT00' DIFFERENT ?
004612  032726  000001                  BEQ     1$              ;BR IF NOT
004616  001405                          MOV     C.SWR,SAVCSW    ;STORE PRESENT 'C.SWR' VALUE
004620  013737  001314  001320          JSR     PC,LODFLT       ;RESET DEFAULT PARAMETERS
004626  004737  023516                  JSR     PC,GT.PRM       ;GET PARAMETERS
004632  004737  027106          1$:     TST     TSTNMS          ;ANY TEST SELECTED THIS CYCLE?
004636  005737  001332                  BNE     6$              ;BR IF YES
004642  001034                          TST     TSTNMS+2        ;ANY TEST SELECTED THIS CYCLE ?
004644  005737  001334                  BNE     6$              ;BR IF YES
004650  001031                          TYPE    ,NOTEST         ;TYPE 'NO TESTS SPECIFIED'
004652  104401  037375                  BR      1$
004656  000765

004660  004737  023516          2$:     JSR     PC,LODFLT       ;SETUP DEFAULT PARAMETERS
004664  005037  001330                  CLR     DRVSEL          ;NO DRIVES SELECTED
004670  005000                          CLR     R0              ;DETERMINE THE DRIVES THAT
004672  012701  000001                  MOV     #1,R1           ;ARE AVAILABLE FOR TESTING

004676  105760  031242          3$:     TSTB    DRVSTA(R0)      ;IS DRIVE ON-LINE ?
004702  003411                          BLE     5$              ;BR IF NO
004704  005737  001470                  TST     XXDP            ;LOADED FROM THIS DEVICE ?
004710  001403                          BEQ     4$              ;BR IF NO
004712  123700  001470                  CMPB    XXDP,R0         ;LOADED FROM THIS DRIVE ?
004716  001403                          BEQ     5$              ;BR IF YES
004720  156037  031356  001330  4$:     BISB    ATABIT(R0),DRVSEL       ;YES, SELECT DRIVE FOR TESTING
004726  005200                  5$:     INC     R0              ;TRY NEXT DRIVE
004730  106301                          ASLB    R1              ;ANY MORE DRIVES TO CHECK ?
004732  001361                          BNE     3$              ;BR IF YES
004734                          6$:
004734  104401  037122          7$:     TYPE    ,DRIVES         ;'DRIVES(S) TO BE TESTED'
004740  005037  015322                  CLR     $ENDCT          ;DETERMINE PASSES TO MAKE AND
004744  005000                          CLR     R0              ;THE DRIVES TO BE TESTED
004746  013701  001330                  MOV     DRVSEL,R1       ;ANY DRIVES SELECTED?
004752  001017                          BNE     9$              ;YES--BRANCH
004754  104401  037153                  TYPE    ,NONE           ;'NONE'
004760  104401  001231                  TYPE    ,$CRLF          ;CR-LF
004764  005737  000042                  TST     @#42            ;ANY MONITOR PRESENT ?
004770  001002                          BNE     8$              ;BR IF YES
004772  000137  003312                  JMP     START2          ;RETURN TO '^C' INPUT
004776  005300                  8$:     DEC     R0              ;THESE TWO LOOPS ARE ADDED TO
005000  001376                          BNE     .-2             ;WAIT FOR TTY
005002  005300                          DEC     R0
005004  001376                          BNE     .-2
005006  000137  015356                  JMP     $GET42          ;RETURN CONTROL TO MONITOR

005012  006201                  9$:     ASR     R1              ;REPORT THE DRIVES TO BE TESTED
005014  103011                          BCC     10$
005016  005237  015322                  INC     $ENDCT          ;GIVE THIS DRIVE A PASS
005022  010046                          MOV     R0,-(SP)        ;;SAVE R0 FOR TYPEOUT
005024  104403                          TYPO,                   ;;GO TYPE--OCTAL ASCII
005026     001                          .BYTE   1               ;;TYPE 1 DIGIT(S)
```

GET UNIT STATUS

```
            005027    000                          .BYTE   0               ;;SUPPRES  'EADING ZEROS
     63     005030    005701                       TST     R1              ;MORE DRIVES?
     64     005032    001404                       BEQ     11$             ;NO--BRANCH
     65     005034    104401  037160               TYPE    ,COMMA          ;','
     66     005040    005200                 10$:  INC     R0              ;FORM DRIVE NUMBER
     67     005042    000763                       BR      9$
     68
     69     005044    104401  001231         11$:  TYPE    ,$CRLF          ;CR-LF
     70     005050    013737  015322  015314        MOV     $ENDCT,$EOPCT
     71     005056    005737  001342               TST     CLK$TA          ;IS KW11-P AVAILABLE ?
     72     005062    003006                       BGT     12$             ;BR IF YES
     73     005064    032737  017000  001332       BIT     #17000,TSTNMS   ;ANY TIMING TESTS TO BE PERFORMED ?
     74     005072    001402                       BEQ     12$             ;BR IF NO
     75     005074    104401  037166               TYPE    ,NOCLOK         ;TYPE NO KW11-P CLOCK MESSAGE
     76     005100    000414                 12$:  BR      RSTRT1
```

PROGRAM RESTARTS HERE

```
                                        .SBTTL  PROGRAM RESTARTS HERE

                                        .ENABL  LSB

005102  005737  001330         RSTART: TST     DRVSEL          ;ANY DRIVES SELECTED ?
005106  001022                         BNE     3$              ;BR IF YES
005110  005737  000042                 TST     @#42            ;ANY MONITOR PRESENT ?
005114  001402                         BEQ     1$              ;BR IF NO
005116  000137  015356                 JMP     $GET42          ;RETURN CONTROL TO MONITOR
005122  104401  037351         1$:     TYPE    ,NODRVS         ;TYPE 'NO DRIVES TO TEST'
005126  000137  003312                 JMP     START2          ;NO--GET DRIVE ENTRY & RESTART AT BEGINNING

005132  005037  001352         RSTRT1: CLR     CHKDRV          ;START WITH DRIVE 0 AGAIN
005136  012737  000001  001354         MOV     #1,DRVMSK
005144  033737  001354  001330 2$:     BIT     DRVMSK,DRVSEL   ;IS THIS DRIVE SELECTED?
005152  001006                         BNE     4$              ;YES--GO CHECK IF DRIVE IS READY FOR TESTING
005154  005237  001352         3$:     INC     CHKDRV          ;MOVE TO NEXT DRIVE NUMBER
005160  106337  001354                 ASLB    DRVMSK          ;DONE TESTING ALL DRIVES ?
005164  103762                         BCS     RSTRT1          ;BR IF YES
005166  000766                         BR      2$              ;NO--CHECK DRIVE SELECT

005170  013702  001352         4$:     MOV     CHKDRV,R2       ;PICKUP THE DRIVE NUMBER
005174  105762  031242                 TSTB    DRVSTA(R2)      ;IS DESIRED DRIVE ON-LINE?
005200  003007                         BGT     5$              ;YES, BRANCH
005202  104011                         EMT     11
005204  043737  001354  001330         BIC     DRVMSK,DRVSEL   ;DESELECT DRIVE FROM TEST
005212  005337  015314                 DEC     $EOPCT          ;ADJUST 'EOP' COUNT
005216  000731                         BR      RSTART          ;RETURN

005220  004737  023146         5$:     JSR     PC,CNTCLR       ;GO CLEAR MASSBUS CONTROLLER
005224  005037  177776                 CLR     PS              ;ENSURE THE PRIORITY = 0
005230  010237  036506                 MOV     R2,DPB.A        ;SET THE DRIVE NUMBER INTO THE DPB'S
005234  010237  036530                 MOV     R2,DPB.B
005240  010237  036552                 MOV     R2,DPB.C
005244  010237  036574                 MOV     R2,DTADPB
005250  004737  024032                 JSR     PC,LDCMD        ;LOAD COMMAND INTO DPB.B AND DPB.C
005254  012737  015142  001350         MOV     #$EOP,BYPASS    ;IF ERROR GO TO END OF PROGRAM
005262  112737  000020  036507         MOVB    #20,DPB.A+1     ;ASSUME 16 BIT FORMAT
005270  032737  000001  001314         BIT     #BIT00,C.SWR    ;16 BIT FORMAT REQUESTED ?
005276  001402                         BEQ     6$              ;BR IF YES
005300  105037  036507                 CLRB    DPB.A+1         ;CLEAR THE 'FMT16' BIT
005304  112737  000143  036510 6$:     MOVB    #SETFORM,DPB.A+2        ;SET THE FORMAT BIT PER DPB.A+1
005312  004037  024076                 JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
005316  112737  000107  036510         MOVB    #^CRCAL,DPB.A+2 ;RECAL=COMMAND
005324  004037  024076                 JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
005330  104401  001231                 TYPE    ,$CRLF          ;CR-LF
005334  104401  037273                 TYPE    ,MSDRIV         ;TYPE 'DRIVE'
005340  010246                         MOV     R2,-(SP)        ;;SAVE R2 FOR TYPEOUT
005342  104403                         TYPOS                   ;;GO TYPE--OCTAL ASCII
005344     002                         .BYTE   2               ;;TYPE 2 DIGIT(S)
005345     000                         .BYTE   0               ;;SUPPRESS LEADING ZEROS
005346  104401  037160                 TYPE    ,CGMMA          ;TYPE ', '
005352  104401  037253                 TYPE    ,SERIAL         ;TYPE 'MBA SN# '
005356  012700  000004                 MOV     #4,R0           ;FOUR DIGITS TO TYPE
005362  013701  036670                 MOV     RM.REG+30,R1    ;SERIAL NUMBER
005366  005002                 7$:     CLR     R2              ;ZERO
005370  006101                         ROL     R1              ;PUT THE NEXT DIGIT
```

PROGRAM RESTARTS HERE

```
     55 005372  006102                          ROL     R2              ;INTO R2
     56 005374  006101                          ROL     R1
     57 005376  006102                          ROL     R2
     58 005400  006101                          ROL     R1
     59 005402  006102                          ROL     R2
     60 005404  006101                          ROL     R1
     61 005406  006102                          ROL     R2
     62 005410  062702  000060                  ADD     #'0,R2          ;MAKE IT ASCII
     63 005414  010227                          MOV     R2,(PC)+        ;SAVE IT
     64 005416  000000                  8$:     .WORD   0
     65 005420  104401  005416                  TYPE    ,8$             ;TYPE
     66 005424  005300                          DEC     R0              ;ALL DIGITS TYPED?
     67 005426  003357                          BGT     7$              ;NO -- BRANCH
     68 005430  104401  001231                  TYPE    ,$CRLF          ;CR-LF
     69 005434  113737  001464  001131          MOVB    ERR.CT,$ERMAX   ;SETUP MAX ERROR COUNT
     70
     71                             .DSABL  LSB
```

PROGRAM RESTARTS HERE

```
                        ;;////////////////////////////////////////////////////////////////
                        ;*IN THE DESCRIPTIONS OF THE BELOW TESTS THE VARIABLES USED
                        ;*AND THEIR DEFAULT VALUES (UNLESS SPECIFIED OTHERWISE) ARE:
                        ;*
                        ;*MNEMONIC      VALUE           VARIABLE
                        ;*--------      -----           --------
                        ;*
                        ;*R             1               ITERATIONS (REPEATS)
                        ;*FC            0               FIRST CYLINDER ADDRESS
                        ;*LC            560.            LAST CYLINDER ADDRESS
                        ;*IC            1               INCREMENT VALUE
                        ;*NC OF NC1     FC+IC           NEW OR MODIFIED CYLINDER
                        ;*                              ADDRRESS
                        ;*NC2           LC-IC           NEW OR MODIFIED CYLINDER
                        ;*                              ADDRESS
                        ;*
                        ;*FT            0               FIRST TRACK ADDRESS
                        ;*LT            13.             LAST TRACK ADDRESS
                        ;*IT            1               INCREMENT VALUE
                        ;*NT            FT+IT           NEW OR MODIFIED TRACK ADDRESS
                        ;*
                        ;*FS            0               FIRST SECTOR ADDRESS
                        ;*LS            30.             LAST SECTOR ADDRESS
                        ;*
                        ;;////////////////////////////////////////////////////////////////

                        .SBTTL  SEEK TESTS

                        ;;////////////////////////////////////////////////////////////////
                        ;*THE SEEK TESTS WILL BE EXCUTED USING IMPLIED SEEKS. THESE
                        ;*IMPLIED SEEKS WILL BE PERFORMED BY "READ HEADER AND
                        ;*DATA" COMMANDS TO TRACK "FT" SECTOR "FS" OF THE DESIRED CYLINDER.
                        ;*THE WORD COUNT WILL BE SET SUCH THAT ONLY THE CYLINDER AND
                        ;*TRACK/SECTOR WORDS OF THE HEADER ARE READ.
                        ;;////////////////////////////////////////////////////////////////


                        ;;****************************************************************
                        ;*TEST 0        RECAL/RANDOM SEEK TEST
                        ;*THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A RECALIBRATE COMMAND
                        ;*AND THEN SEEK TO A  RANDOM  CYLINDER BETWEEN "FC" AND "LC".  AT
                        ;*THE COMPLETION OF BOTH COMMANDS,  STATUS INDICATORS ARE CHECKED
                        ;*TO ENSURE THAT NO ERRORS OCCURRED.
                        ;;****************************************************************
005442                  TST0:
005442  000240                  NOP
005444  033737  001560  001332  BIT     BITS+<0*2>,TSTNMS       ;DO THIS TEST?
005452  001002                  BNE     .+6                     ;BR IF YES
005454  000137  005704          JMP     TST1                    ;NO--JUMP TO TEST1

005460  012737  000000  001116  MOV     #0,$TSTNM               ;SET TEST #0 AND CLEAR ($ERFLG)
005466  004737  023734          JSR     PC,LODPRM               ;LOAD THE PARMETERS FOR THE TEST
005472  012737  005612  001124  MOV     #TST0,$LPERR            ;SETUP THE LOOP ON ERROR ADDRESS
005500  013737  002322  001220  MOV     RPT,$TIMES              ;GET THE ITERATION COUNT
005506  112737  000031  001131  MOVB    #25.,$ERMAX             ;MAX ERRORS ALLOWED FOR TEST
005514  012737  000000  001240  MOV     #0,$TESTN               ;;SET TEST NUMBER IN APT MAIL BOX
```

```
005522  032777  010000  173424          BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
005530  001406                          BEQ     .+16            ;BR IF YES
005532  104401  037265                  TYPE    ,MSGTST         ;TYPE 'TEST'
005536  013746  001240                  MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
005542  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
005544     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
005545     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS

 46 005546  112737  000107  036510      MOVB    #RECAL,DPB.A+2  ;RECAL=COMMAND
 47 005554  113737  002340  036540      MOVB    FS,DPB.B+10     ;FS
 48 005562  113737  002332  036541      MOVB    FT,DPB.B+11     ;FT
 49 005570  013737  002326  036542      MOV     LC,DPB.B+12     ;LC
 70 005576  012737  005702  001350      MOV     #EXIT0,BYPASS   ;GO TO EXIT0 ON ERROR
    005604  012737  005612  001122      MOV     #TEST0,$LPADR   ;SETUP LOOP ADDRESS
    005612                          TEST0:
    005612  012706  001100              MOV     #STACK,SP       ;SET UP STACK POINTER
    005616  004037  024076              JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
    005622  013737  002324  036542      MOV     FC,DPB.B+12     ;INITIAL CYLINDER ADDRESS
    005630  023737  002324  002326      CMP     FC,LC           ;CYLINDER LIMITS THE SAME ?
    005636  001417                      BEQ     1$              ;BR IF THEY ARE
    005640  004737  022502              JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
    005644  013746  022600              MOV     $HINUM,-(SP)    ;USE THE HIGH RANDOM NUMBER
    005650  005046                      CLR     -(SP)           ;UPPER DIVIDEND
    005652  013746  002326              MOV     LC,-(SP)        ;FORM THE DIVISOR
    005656  005216                      INC     (SP)            ;INCREMENT
    005660  163716  002324              SUB     FC,(SP)         ;SUBTRACT THE LOWER LIMIT
    005664  004737  022604              JSR     PC,$DIV         ;DIVIDE
    005670  062637  036542              ADD     (SP)+,DPB.B+12  ;ADD THE REMAINDER TO THE INITIAL CYLINDER
    005674  005726                      TST     (SP)+           ;DISCARD THE QUOTENT
    005676                          1$:
    005676  004037  024244              JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
    005702  000004                  EXIT0:  SCOPE               ;CALL SCOPE ROUTINE
 71
 78
 79                                 ;;*****************************************************************
                                    ;*TEST 1        SEEK/SEEK TEST
                                    ;*THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A FORWARD SEEK
                                    ;*CYCLE TO "LC", "LT", "LS" FOLLOWED BY A REVERSE SEEK CYCLE TO
                                    ;*"FC", "FT", "FS".  AT THE COMPLETION OF EACH SEEK, THE PROPER
                                    ;*INDICATORS ARE EXAMINED TO ENSURE PROPER OPERATION.
                                    ;;*****************************************************************
    005704                          TST1:
    005704  000240                          NOP
    005706  033737  001562  001332          BIT     BITS+<1*2>,TSTNMS       ;DO THIS TEST?
    005714  001002                          BNE     .+6             ;BR IF YES
    005716  000137  006112                  JMP     TST2            ;NO--JUMP TO TEST2

    005722  012737  000001  001116          MOV     #1,$TSTNM       ;SET TEST #1 AND CLEAR ($ERFLG)
    005730  004737  023734                  JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
    005734  012737  006074  001124          MOV     #TEST1,$LPERR   ;SETUP THE LOOP ON ERROR ADDRESS
    005742  013737  002322  001220          MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
    005750  112737  000031  001131          MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
    005756  012737  000001  001240          MOV     #1,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
 80
    005764  032777  010000  173162          BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
    005772  001406                          BEQ     .+16            ;BR IF YES
    005774  104401  037265                  TYPE    ,MSGTST         ;TYPE 'TEST'
```

```
006000  013746  001240                  MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
006004  104403                           TYPOS                   ;;GO TYPE--OCTAL ASCII
006006     002                           .BYTE   2               ;;TYPE 2 DIGIT(S)
006007     000                           .BYTE   0               ;;SUPPRESS LEADING ZEROS

 81 006010  005037  001466               CLR     BASFLG          ;CLEAR BAD SECTOR ENCOUNTER FOR THE DRIVE
 82 006014  113737  002340  036540       MOVB    FS,DPB.B+10     ;FS
 83 006022  113737  002342  036562       MOVB    LS,DPB.C+10     ;LS
 84 006030  113737  002332  036541       MOVB    FT,DPB.B+11     ;FT
 85 006036  113737  002334  036563       MOVB    LT,DPB.C+11     ;LT
 86 006044  013737  002324  036542       MOV     FC,DPB.B+12     ;FC
 87 006052  013737  002326  036564       MOV     LC,DPB.C+12     ;LC
 92 006060  012737  006110  001350       MOV     #EXIT1,BYPASS   ;GO TO EXIT1 ON ERROR
    006066  012737  006074  001122       MOV     #TEST1,$LPADR   ;SETUP LOOP ADDRESS
    006074                       TEST1:
 93 006074  012706  001100               MOV     #STACK,SP       ;SET THE STACK POINTER
 94 006100  004037  024466               JSR     R0,CALL.C       ;GO EXECUTE THE COMMAND
 95 006104  004037  024244               JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
 96 006110  000004               EXIT1:  SCOPE                   ;CALL SCOPE ROUTINE
 97
108
109                                     ;;*****************************************************************
                                        ;*TEST 2        INCREMENT/SEEK TEST
                                        ;*THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
                                        ;*CYLINDER ADDRESS FROM "FC" TO "LC" BY THE INCREMENT "IC".
                                        ;*WHEN THE RESULTANT CYLINDER ADDRESS (NC) EXCEEDS
                                        ;*"LC" REVERSE SEEK CYCLES ARE INITIATED; STARTING
                                        ;*AT THE LAST LEGAL "NC" AND DECREMENTING BY "IC"
                                        ;*UNTIL "NC" IS LESS THAN "FC". AT THE COMPLETION OF EACH
                                        ;*SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
                                        ;*ENSURE PROPER OPERATION.
                                        ;;*****************************************************************
006112                          TST2:
006112  000240                          NOP
006114  033737  001564  001332          BIT     BITS+<2*2>,TSTNMS       ;DO THIS TEST?
006122  001002                          BNE     .+6             ;BR IF YES
006124  000137  006354                  JMP     TST3            ;NO--JUMP TO TEST3

006130  012737  000002  001116          MOV     #2,$TSTNM       ;SET TEST #2 AND CLEAR ($ERFLG)
006136  004737  023734                  JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
006142  012737  006246  001124          MOV     #TEST2,$LPERR   ;SETUP THE LOOP ON ERROR ADDRESS
006150  013737  002322  001220          MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
006156  112737  000031  001131          MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
006164  012737  000002  001240          MOV     #2,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
110
006172  032777  010000  172754          BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
006200  001406                          BEQ     .+16            ;BR IF YES
006202  104401  037265                  TYPE    ,MSGTST         ;TYPE 'TEST'
006206  013746  001240                  MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
006212  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
006214     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
006215     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS

111 006216  012737  006224  001122          MOV     #1$,$LPADR      ;SETUP LOOP ADDRESS
112 006224  113737  002340  036540  1$:     MOVB    FS,DPB.B+10     ;FS
113 006232  113737  002332  036541          MOVB    FT,DPB.B+11     ;FT
117 006240  012737  006352  001350          MOV     #EXIT2,BYPASS   ;GO TO EXIT2 ON ERROR
```

```
                006246                                  TEST2:
    118 006246  013737  002324  036542                          MOV     FC,DPB.B+12     ;FC
    119 006254  012737  006254  001124                          MOV     #.,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
        006262  012706  001100                              MOV     #STACK,SP       ;LOAD THE STACK POINTER
    120 006266                                  INCSK:
    121 006266  004037  024244                          JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
    122 006272  063737  002330  036542                  ADD     IC,DPB.B+12     ;MOVE TO NEXT CYLINDER
    123 006300  023737  002326  036542                  CMP     LC,DPB.B+12     ;OUT OF CYLINDERS?
    124 006306  002367                                  BGE     INCSK           ;NO--BRANCH
    125 006310  013737  002326  036542                  MOV     LC,DPB.B+12
    126 006316  012737  006316  001124                  MOV     #.,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
        006324  012706  001100                              MOV     #STACK,SP       ;LOAD THE STACK POINTER
    127 006330                                  DECSK:
    128 006330  004037  024244                          JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
    129 006334  163737  002330  036542                  SUB     IC,DPB.B+12
    130 006342  023737  002324  036542                  CMP     FC,DPB.B+12
    131 006350  003767                                  BLE     DECSK
    132 006352  000004                          EXIT2:  SCOPE                   ;CALL SCOPE ROUTINE
    133
    140
    141                                         ;;*****************************************************************
                                                ;*TEST 3         STEPPING SEEK TEST
                                                ;*THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0, 1, 2, 4,
                                                ;*8, 16, 32, 64, 128, 256 AND 512. AT THE COMPLETION OF EACH
                                                ;*SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO ENSURE
                                                ;*PROPER OPERATION.
                                                ;;*****************************************************************
        006354                                  TST3:
        006354  000240                                  NOP
        006356  033737  001566  001332                  BIT     BITS+<3*2>,TSTNMS       ;DO THIS TEST?
        006364  001002                                  BNE     .+6             ;BR IF YES
        006366  000137  006574                          JMP     TST4            ;NO--JUMP TO TEST4

        006372  012737  000003  001116                  MOV     #3,$TSTNM       ;SET TEST #3 AND CLEAR ($ERFLG)
        006400  004737  023734                          JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
        006404  012737  006510  001124                  MOV     #TEST3,$LPERR   ;SETUP THE LOOP ON ERROR ADDRESS
        006412  013737  002322  001220                  MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
        006420  112737  000031  001131                  MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
        006426  012737  000003  001240                  MOV     #3,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
    142
        006434  032777  010000  172512                  BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
        006442  001406                                  BEQ     .+16            ;BR IF YES
        006444  104401  037265                          TYPE    ,MSGTST         ;TYPE 'TEST'
        006450  013746  001240                          MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
        006454  104403                                  TYPOS                   ;;GO TYPE--OCTAL ASCII
        006456     002                                  .BYTE   2               ;;TYPE 2 DIGIT(S)
        006457     000                                  .BYTE   0               ;;SUPPRESS LEADING ZEROS

    143 006460  012737  006466  001122                  MOV     #1$,$LPADR      ;SETUP TEST LOOP ADDRESS
    144 006466  113737  002340  036540          1$:     MOVB    FS,DPB.B+10     ;FS
    145 006474  113737  002332  036541                  MOVB    FT,DPB.B+11     ;FT
    149 006502  012737  006572  001350                  MOV     #EXIT3,BYPASS   ;GO TO BYPASS ON ERROR
        006510                                  TEST3:
    150 006510  013737  002324  036542                  MOV     FC,DPB.B+12     ;FC
    151 006516  012737  006516  001124                  MOV     #.,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
        006524  012706  001100                          MOV     #STACK,SP       ;LOAD THE STACK POINTER
    152 006530  004037  024244                          JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
```

```
153 006534  013701  002330                  MOV     IC,R1               ;CYLINDER 1
154 006540  012737  006540  001124          MOV     #.,$LPERR           ;SETUP THE ERROR LOOP ADDRESS
    006546  012706  001100                  MOV     #STACK,SP           ;LOAD THE STACK POINTER
155 006552  010137  036542          1$:     MOV     R1,DPB.B+12         ;DESIRED CYLINDER
156 006556  004037  024244                  JSR     R0,CALL.B           ;GO EXECUTE THE COMMAND
157 006562  006301                          ASL     R1                  ;MOVE TO NEXT CYLINDER
158 006564  020137  002326                  CMP     R1,LC               ;DONE?
159 006570  003770                          BLE     1$                  ;NO--LOOP
160 006572  000004                  EXIT3:  SCOPE                       ;CALL SCOPE ROUTINE
161
169
170                                         ;;*****************************************************************
                                            ;*TEST 4        OSCILLATING SEEK TEST
                                            ;*THIS TEST WILL COMMAND SEEK CYCLES FROM "FC" TO "NC" AND BACK
                                            ;*TO "FC".  "NC" STARTS AT "FC" AND INCREMENTS BY "IC" UP TO CYLINDER
                                            ;*"LC", THEN IS DECREMENTED BY "IC" BACK TO CYLINDER "FC".  AT THE
                                            ;*COMPLETION OF EVERY SEEK COMMAND THE PROPER INDICATORS ARE
                                            ;*EXAMINED TO ENSURE PROPER OPERATION.
                                            ;;*****************************************************************
    006574                                  TST4:
    006574  000240                          NOP
    006576  033737  001570  001332          BIT     BITS+<4*2>,TSTNMS       ;DO THIS TEST?
    006604  001002                          BNE     .+6                 ;BR IF YES
    006606  000137  007206                  JMP     TST5                ;NO--JUMP TO TEST5

    006612  012737  000004  001116          MOV     #4,$TSTNM           ;SET TEST #4 AND CLEAR ($ERFLG)
    006620  004737  023734                  JSR     PC,LODPRM           ;LOAD THE PARMETERS FOR THE TEST
    006624  012737  006744  001124          MOV     #TEST4,$LPERR       ;SETUP THE LOOP ON ERROR ADDRESS
    006632  013737  002322  001220          MOV     RPT,$TIMES          ;GET THE ITERATION COUNT
    006640  112737  000031  001131          MOVB    #25.,$ERMAX         ;MAX ERRORS ALLOWED FOR TEST
    006646  012737  000004  001240          MOV     #4,$TESTN           ;;SET TEST NUMBER IN APT MAIL BOX
171
    006654  032777  010000  172272          BIT     #SW12,@SWR          ;INHIBIT TYPING TEST NUMBER ?
    006662  001406                          BEQ     .+16                ;BR IF YES
    006664  104401  037265                  TYPE    ,MSGTST             ;TYPE 'TEST'
    006670  013746  001240                  MOV     $TESTN,-(SP)        ;;SAVE $TESTN FOR TYPEOUT
    006674  104403                          TYPOS                       ;;GO TYPE--OCTAL ASCII
    006676     002                          .BYTE   2                   ;;TYPE 2 DIGIT(S)
    006677     000                          .BYTE   0                   ;;SUPPRESS LEADING ZEROS

172 006700  012737  006706  001122          MOV     #1$,$LPADR          ;SETUP LOOP ADDRESS
173 006706  113737  002340  036540  1$:     MOVB    FS,DPB.B+10         ;FS
174 006714  113737  002332  036541          MOVB    FT,DPB.B+11         ;FT
182 006722  012737  007204  001350          MOV     #EXIT4,BYPASS       ;GO TO EXIT4 ON ERROR
    006730  005002                          CLR     R2                  ;CLEAR STALL SWITCH (NO STALL)
    006732  032737  010000  001314          BIT     #SW12,C.SWR         ;STALL REQUIRED?
    006740  001401                          BEQ     TEST4               ;NO--BRANCH
    006742  005102                          COM     R2                  ;YES--SET SWITCH
    006744                          TEST4:
183 006744  013701  002324                  MOV     FC,R1               ;SET NC TO FC
184 006750  005037  001436                  CLR     STALL0  ;START AT ZERO IF STALLS REQUIRED
185 006754  012737  006754  001124          MOV     #.,$LPERR           ;SETUP THE ERROR LOOP ADDRESS
    006762  012706  001100                  MOV     #STACK,SP           ;LOAD THE STACK POINTER
186 006766  010137  036542          1$:     MOV     R1,DPB.B+12         ;NC
187 006772  004037  024244                  JSR     R0,CALL.B           ;GO EXECUTE THE COMMAND
188 006776  005702                          TST     R2                  ;STALL?
189 007000  001403                          BEQ     2$                  ;NO--BRANCH
```

```
                                                JSR     R0,STALL        ;YES-GO TO STALL ROUTINE
                                                .WORD   STALL0          ;TIME POINTER
007010  013737  002324  036542  2$:           MOV     FC,DPB.B+12     ;FC
007016  004037  024244                        JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
007022  005702                                TST     R2              ;STALL?
007024  001413                                BEQ     3$              ;NO--BRANCH
007026  004037  025220                        JSR     R0,STALL        ;YES--GO TO STALL ROUTINE
007032  001436                                .WORD   STALL0          ;TIME POINTER
007034  005237  001436                        INC     STALL0  ;UPDATE THE TIME
007040  023737  001462  001436                CMP     MXSTAL,STALL0   ;TIME TO BIG?
007046  003347                                BGT     1$              ;NO--BRANCH
007050  005037  001436                        CLR     STALL0  ;YES--START OVER AT ZERO
007054  063701  002330            3$:         ADD     IC,R1           ;MOVE TO NEXT CYLINDER
007060  020137  002326                        CMP     R1,LC           ;LAST CYLINDER COMPLETED?
007064  003740                                BLE     1$              ;NO--BRANCH
007066  013701  002326                        MOV     LC,R1           ;SET NC TO LC
007072  012737  007072  001124                MOV     #.,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
007100  012706  001100                        MOV     #STACK,SP       ;LOAD THE STACK POINTER
007104  010137  036542            4$:         MOV     R1,DPB.B+12     ;NC
007110  004037  024244                        JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
007114  005702                                TST     R2              ;STALL?
007116  001403                                BEQ     5$              ;NO--BRANCH
007120  004037  025220                        JSR     R0,STALL        ;YES--GO TO STALL ROUTINE
007124  001436                                .WORD   STALL0          ;TIME POINTER
007126  013737  002326  036542  5$:           MOV     LC,DPB.B+12     ;LC
007134  004037  024244                        JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
007140  005702                                TST     R2              ;STALL?
007142  001413                                BEQ     6$              ;NO--BRANCH
007144  004037  025220                        JSR     R0,STALL        ;YES--GO TO STALL ROUTINE
007150  001436                                .WORD   STALL0          ;TIME POINTER
007152  005237  001436                        INC     STALL0  ;UPDATE STALL TIME
007156  023737  001462  001436                CMP     MXSTAL,STALL0   ;TIME TOO BIG?
007164  003347                                BGT     4$              ;NO--BRANCH
007166  005037  001436                        CLR     STALL0  ;YES--SET STALL TIME BACK TO ZERO
007172  163701  002330            6$:         SUB     IC,R1           ;NEXT CYLINDER
007176  020137  002324                        CMP     R1,FC           ;DONE?
007202  002340                                BGE     4$              ;NO--BRANCH
007204  000004                    EXIT4:      SCOPE                   ;CALL SCOPE ROUTINE


                                            ;:*****************************************************************
                                            ;*TEST 5        CONVERGING/DIVERGING SEEK TEST
                                            ;*THIS TEST WILL CAUSE THE DRIVE TO EXECUTE FORWARD AND REVERSE
                                            ;*SEEKS FROM "NC1" AND "NC2" RESPECTIVELY. "NC1" WILL BE INCREMENTED
                                            ;*BY "IC" AND "NC2" WILL BE DECREMENTED BY "IC" UNTIL "NC1" IS
                                            ;*GREATER THAN THE INITIAL VALUE OF "NC2" AND "NC2" IS
                                            ;*LESS THAN THE INITIAL VALUE OF "NC1".  AT THE COMPLETION OF
                                            ;*EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
                                            ;*ENSURE PROPER OPERATION.  "NC1" AND "NC2" DEFAULT TO
                                            ;*"FC" AND "LC" RESPECTIVELY.
                                            ;:*****************************************************************
007206                                      TST5:
007206  000240                                    NOP
007210  033737  001572  001332                    BIT     BITS+<5*2>,TSTNMS       ;DO THIS TEST?
007216  001002                                    BNE     .+6             ;BR IF YES
007220  000137  007432                            JMP     TST6            ;NO--JUMP TO TEST6
```

```
                007224  012737  000005  001116          MOV     #5,$TSTNM       ;SET TEST #5 AND CLEAR ($ERFLG)
                007232  004737  023734                  JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
                007236  012737  007342  001124          MOV     #TEST5,$LPERR   ;SETUP THE LOOP ON ERROR ADDRESS
                007244  013737  002322  001220          MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
                007252  112737  000031  001131          MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
                007260  012737  000005  001240          MOV     #5,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
            240
                007266  032777  010000  171660          BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
                007274  001406                          BEQ     .+16            ;BR IF YES
                007276  104401  037265                  TYPE    ,MSGTST         ;TYPE 'TEST'
                007302  013746  001240                  MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
                007306  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
                007310     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
                007311     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS

            241 007312  012737  007320  001122          MOV     #1$,$LPADR      ;SETUP LOOP ADDRESS
            242 007320  113737  002340  036540  1$:     MOVB    FS,DPB.B+10     ;FS
            243 007326  113737  002332  036541          MOVB    FT,DPB.B+11     ;FT
            247 007334  012737  007430  001350          MOV     #EXIT5,BYPASS   ;GO TO EXIT5 ON ERROR
                007342                          TEST5:
            248 007342  013701  002324                  MOV     FC,R1           ;START NC1 AT FC
            249 007346  013702  002326                  MOV     LC,R2           ;START NC2 AT LC
            250 007352  012737  007352  001124          MOV     #.,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
                007360  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
            251 007364  010137  036542          1$:     MOV     R1,DPB.B+12     ;NC1
            252 007370  004037  024244                  JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
            253 007374  010237  036542                  MOV     R2,DPB.B+12     ;NC2
            254 007400  004037  024244                  JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
            255 007404  063701  002330                  ADD     IC,R1           ;NEXT NC1
            256 007410  163702  002330                  SUB     IC,R2           ;NEXT NC2
            257 007414  020137  002326                  CMP     R1,LC           ;DONE?
            260 007420  003003                          BGT     EXIT5           ;YES--BRANCH
            261 007422  020237  002324                  CMP     R2,FC   ;?
            262 007426  002356                          BGE     1$              ;NO--BRANCH
            263 007430  000004                  EXIT5:  SCOPE                   ;CALL SCOPE ROUTINE
            264
            273
            274                                 ;;*****************************************************************
                                                ;*TEST 6        SERVO ADDRESSING LOGIC NOISE GENERATOR
                                                ;*IN THIS TEST A SEEK IS DONE TO CYL "NC" THEN A SEEK TO
                                                ;*NC+4 THEN NC+1 THEN NC+3 THEN NC+2 THEN NC+5.  NOW "NC" IS UPDATED
                                                ;*BY "IC" AND THE ABOVE SEQUENCE IS REPEATED UNITL "LC" IS
                                                ;*EXCEEDED BY ANY OF THE ABOVE VALUES.  THE INITIAL VALUE OF "NC"
                                                ;*IS "FC".  AT THE COMPLETION OF EACH SEEK COMMAND THE
                                                ;*PROPER INDICATORS ARE EXAMINED TO ENSURE PROPER OPERATION.
                                                ;;*****************************************************************
                007432                          TST6:
                007432  000240                          NOP
                007434  033737  001574  001332          BIT     BITS+<6*2>,TSTNMS       ;DO THIS TEST?
                007442  001002                          BNE     .+6             ;BR IF YES
                007444  000137  007722                  JMP     TST7            ;NO--JUMP TO TEST7

                007450  012737  000006  001116          MOV     #6,$TSTNM       ;SET TEST #6 AND CLEAR ($ERFLG)
                007456  004737  023734                  JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
                007462  012737  007566  001124          MOV     #TEST6,$LPERR   ;SETUP THE LOOP ON ERROR ADDRESS
                007470  013737  002322  001220          MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
                007476  112737  000031  001131          MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
```

```
                007504  012737  000006  001240          MOV     #6,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
    275
                007512  032777  010000  171434          BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
                007520  001406                          BEQ     .+16            ;BR IF YES
                007522  104401  037265                  TYPE    ,MSGTST         ;TYPE 'TEST'
                007526  013746  001240                  MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
                007532  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
                007534     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
                007535     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS

    276 007536  012737  007544  001122          1$:     MOV     #1$,$LPADR      ;SETUP LOOP ADDRESS
    277 007544  113737  002340  036540          1$:     MOVB    FS,DPB.B+10     ;FS
    278 007552  113737  002332  036541                  MOVB    FT,DPB.B+11     ;FT
    282 007560  012737  007720  001350                  MOV     #EXIT6,BYPASS   ;GO TO EXIT6 ON ERROR
        007566                          TEST6:
    283 007566  013701  002324                          MOV     FC,R1           ;PICKUP 'FC'
    284 007572  013702  002326                          MOV     LC,R2           ;FORM LAST CYLINDER THAT
    285 007576  162702  000005                          SUB     #5,R2           ;IS AVAILABLE FOR TESTING
    286 007602  012737  007602  001124                  MOV     #.,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
        007610  012706  001100                          MOV     #STACK,SP       ;LOAD THE STACK POINTER
    287 007614  020102                          1$:     CMP     R1,R2           ;LAST CYLINDER
    290 007616  003040                                  BGT     EXIT6           ;YES--BRANCH
    291 007620  010137  036542                          MOV     R1,DPB.B+12     ;NC
    292 007624  004037  024244                          JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
    293 007630  062737  000004  036542                  ADD     #4,DPB.B+12     ;NC+4
    294 007636  004037  024244                          JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
    295 007642  162737  000003  036542                  SUB     #3,DPB.B+12     ;NC+1
    296 007650  004037  024244                          JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
    297 007654  062737  000002  036542                  ADD     #2,DPB.B+12     ;NC+3
    298 007662  004037  024244                          JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
    299 007666  162737  000001  036542                  SUB     #1,DPB.B+12     ;NC+2
    300 007674  004037  024244                          JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
    301 007700  062737  000003  036542                  ADD     #3,DPB.B+12     ;NC+5
    302 007706  004037  024244                          JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
    303 007712  063701  002330                          ADD     IC,R1
    304 007716  000736                                  BR      1$
    305 007720  000004                          EXIT6:  SCOPE                   ;CALL SCOPE ROUTINE
    306
    315
    316                                         ;;*****************************************************************
                                                ;*TEST 7         RANDOM SEEK TEST
                                                ;*THIS TEST PERFORMS RANDOM SEEK OPERATIONS BETWEEN CYLINDERS 'FC'
                                                ;*'LC'.  AFTER EACH SEEK, THE POSITION OF THE DRIVE IS VERIFIED BY
                                                ;*READING A SECTOR FROM THE CURRENTLY ADDRESSED CYLINDER AND TRACK.
                                                ;*THE TRACK ADDRESS IS INCREMENTED FOR EACH SEEK SO THAT VERIFICATION
                                                ;*OF POSITIONING OCCURS USING EACH HEAD.  TRACK ADDRESSES ARE INCREMENTED
                                                ;*BETWEEN PARAMTERS 'FT' AND 'LT'.
                                                ;;*****************************************************************
        007722                                  TST7:
        007722  000240                                  NOP
        007724  033737  001576  001332                  BIT     BITS+<7*2>,TSTNMS       ;DO THIS TEST?
        007732  001002                                  BNE     .+6             ;BR IF YES
        007734  000137  010320                          JMP     TST10           ;NO--JUMP TO TEST10

        007740  012737  000007  001116                  MOV     #7,$TSTNM       ;SET TEST #7 AND CLEAR ($ERFLG)
        007746  004737  023734                          JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
        007752  012737  010056  001124                  MOV     #TEST7,$LPERR   ;SETUP THE LOOP ON ERROR ADDRESS
```

```
                007760  013737  002322  001220          MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
                007766  112737  000031  001131          MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
                007774  012737  000007  001240          MOV     #7,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
    317
                010002  032777  010000  171144          BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
                010010  001406                          BEQ     .+16            ;BR IF YES
                010012  104401  037265                  TYPE    ,MSGTST         ;TYPE 'TEST'
                010016  013746  001240                  MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
                010022  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
                010024     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
                010025     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS

    318 010026  113737  002332  036541                  MOVB    FT,DPB.B+11     ;LOAD STARTING TRACK ADDRESS
    319 010034  112737  000105  036510                  MOVB    #SEEK,DPB.A+2   ;SEEK=COMMAND
    324 010042  012737  010316  001350                  MOV     #EXIT7,BYPASS   ;ERROR TERMINATION ADDRESS
        010050  012737  010056  001122                  MOV     #TEST7,$LPADR   ;SETUP THE LOOP ON TEST ADDRESS
        010056  013704  031366          TEST7:  MOV     RMADR,R4        ;UNIBUS ADDRESS OF THE RH/RM
    325 010062  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
    326 010066  013737  002324  036542          MOV     FC,DPB.B+12     ;INITIAL CYLINDER ADDRESS
    327 010074  023737  002324  002326          CMP     FC,LC           ;CYLINDER LIMITS THE SAME ?
    328 010102  001422                          BEQ     1$              ;BR IF THEY ARE
    329 010104  004737  022502                  JSR     PC,$RAND        ;CYCYLE THE RANDOM NUMBER GENERATOR
    330 010110  013746  022600                  MOV     $HINUM,-(SP)    ;USE THE HIGH RANDOM NUMBER
    331 010114  005046                          CLR     -(SP)           ;UPPER DIVIDEND
    332 010116  013746  002326                  MOV     LC,-(SP)        ;FORM THE DIVISOR
    333 010122  005216                          INC     (SP)            ;INCREMENT
    334 010124  163716  002324                  SUB     FC,(SP)         ;SUBTRACT THE LOWER LIMIT
    335 010130  004737  022604                  JSR     PC,$DIV         ;DIVIDE
    336 010134  062637  036542                  ADD     (SP)+,DPB.B+12  ;ADD THE REMAINDER TO THE INITIAL CYLINDER
    337 010140  005726                          TST     (SP)+           ;DISCARD THE QUOTENT
    338 010142  013737  036542  036520          MOV     DPB.B+12,DPB.A+12       ;COPY NEW CYLINDER ADDRESS
    339 010150                          1$:
        010150  012737  010150  001124          MOV     #.,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
        010156  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
    340 010162  004037  024076                  JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
    341 010166  012737  010166  001124          MOV     #.,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
        010174  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
    342 010200  113764  036506  000010          MOVB    DPB.A,RMCS2(R4) ;SELECT THE DRIVE
    343 010206  016446  000020                  MOV     RMLA(R4),-(SP)  ;GET THE LOOK AHEAD REGISTER
    344 010212  006316                          ASL     (SP)            ;ALIGN THE SECTOR ADDRESS
    345 010214  006316                          ASL     (SP)            ;ALIGN THE SECTOR ADDRESS
    346 010216  000316                          SWAB    (SP)            ;PUT ADDRESS IN LOWER BYTE
    347 010220  105766  000001                  TSTB    1(SP)           ;IN THE 1ST 20% OF SECTOR ?
    348 010224  001401                          BEQ     2$              ;BR IF YES
    349 010226  105216                          INCB    (SP)            ;INCREMENT THE SECTOR ADDRESS
    350 010230  105216                  2$:     INCB    (SP)            ;INCREMENT THE SECTOR ADDRESS
    351 010232  112637  036604                  MOVB    (SP)+,DTADPB+10 ;LOAD THE DPB
    352 010236  013746  002426                  MOV     PRMLMT+16,-(SP) ;PUT LAST SECTOR ADDRESS ON THE STACK
    353 010242  005216                          INC     (SP)            ;INCREMENT IT
    354 010244  122637  036604                  CMPB    (SP)+,DTADPB+10 ;NEW SECTOR ADDRESS TOO LARGE ?
    355 010250  103007                          BHIS    4$              ;BR IF NOT
    356 010252  103403                          BLO     3$              ;BR IF ADDRESS IS 2 GREATER
    357 010254  105037  036604                  CLRB    DTADPB+10       ;RESET TO SECTOR ADDRESS 0
    358 010260  000403                          BR      4$              ;CONTINUE
    359 010262  112737  000001  036604  3$:     MOVB    #1,DTADPB+10    ;RESET ADDRESS TO SECTOR 1
    360 010270                          4$:
        010270  004037  024244                  JSR     R0,CALL.B       ;GO EXECUTE THE COMMAND
```

```
 361 010274  105237  036541                 INCB    DPB.B+11        ;INCREMENT THE TRACK ADDRESS
 362 010300  123737  036541  002334         CMPB    DPB.B+11,LT     ;MAXIMUM ?
 365 010306  101403                         BLOS    EXIT7           ;BR IF NOT
 366 010310  113737  002332  036541         MOVB    FT,DPB.B+11     ;RELOAD STARTING TRACK ADDRESS
 367 010316  000004                 EXIT7:  SCOPE                   ;CALL SCOPE ROUTINE
 368
 379
 380                                        ;;*****************************************************************
                                            ;*TEST 10       ALL SEEKS TEST
                                            ;*THIS TEST VERIFIES THAT THE DISK DRIVE CAN SEEK FROM EACH CYLINDER
                                            ;*TO ALL OTHER CYLINDERS.
                                            ;*
                                            ;*BEGINNING WITH CYLINDER 'FC', THE TEST SEEKS TO EACH CYLINDER
                                            ;*BETWEEN 'FC' AND 'LC' FROM CYINDER 'FC'.  THE BEGINNING CYLINDER
                                            ;*ADDRESS IS INCREMENTED AND THE TEST SEEKS BETWEEN THE NEW CYLINDER
                                            ;*ADDRESS AND ALL CYLINDERS BETWEEN 'FC' AND 'LC'.  THE SEQUENCE
                                            ;*CONTINUES UNTIL ALL CYLINDERS HAVE BEEN CHECKED.
                                            ;;*****************************************************************
      010320                        TST10:
      010320  000240                        NOP
      010322  033737  001600  001332         BIT     BITS+<10*2>,TSTNMS      ;DO THIS TEST?
      010330  001002                         BNE     .+6             ;BR IF YES
      010332  000137  010564                 JMP     TST11           ;NO--JUMP TO TEST11

      010336  012737  000010  001116         MOV     #10,$TSTNM      ;SET TEST #10 AND CLEAR ($ERFLG)
      010344  004737  023734                 JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
      010350  012737  010504  001124         MOV     #TEST10,$LPERR  ;SETUP THE LOOP ON ERROR ADDRESS
      010356  013737  002322  001220         MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
      010364  112737  000031  001131         MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
      010372  012737  000010  001240         MOV     #10,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 381
      010400  032777  010000  170546         BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
      010406  001406                         BEQ     .+16            ;BR IF YES
      010410  104401  037265                 TYPE    ,MSGTST         ;TYPE 'TEST'
      010414  013746  001240                 MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
      010420  104403                         TYPOS                   ;;GO TYPE--OCTAL ASCII
      010422     003                         .BYTE   3               ;;TYPE 3 DIGIT(S)
      010423     000                         .BYTE   0               ;;SUPPRESS LEADING ZEROS

 382 010424  012737  010432  001122         MOV     #1$,$LPADR      ;SETUP THE LOOP ADDRESS
 383 010432  113737  002340  036540  1$:    MOVB    FS,DPB.B+10     ;SECTOR ADDRESS
 384 010440  113737  002340  036562         MOVB    FS,DPB.C+10     ;SECTOR ADDRESS
 385 010446  113737  002332  036541         MOVB    FT,DPB.B+11     ;TRACK ADDRESS
 386 010454  113737  002332  036563         MOVB    FT,DPB.C+11     ;TRACK ADDRESS
 387 010462  013737  002324  036542         MOV     FC,DPB.B+12     ;STARTING CYLINDER ADDRESS
 388 010470  013737  002324  036564         MOV     FC,DPB.C+12     ;STARTING CYLINDER ADDRESS
 392 010476  012737  010562  001350         MOV     #EXIT10,BYPASS  ;TEST ABORT EXIT
      010504                        TEST10:
 393 010504  012706  001100                 MOV     #STACK,SP       ;SETUP THE STACK POINTER
 394 010510                         1$:
      010510  004037  024466                 JSR     R0,CALL.C       ;GO EXECUTE THE COMMAND
 395 010514  004037  024244                 JSR     R0,CALL.3       ;GO EXECUTE THE COMMAND
 396 010520  063737  002330  036564         ADD     IC,DPB.C+12     ;INCREMENT THE ENDING CYLINDER ADDRESS
 397 010526  023737  002326  036564         CMP     LC,DPB.C+12     ;CHECK IF EXCEEDING MAXIMUM
 398 010534  002365                         BGE     1$              ;BR IF NOT
 399 010536  013737  002324  036564         MOV     FC,DPB.C+12     ;RESET ENDING CYLINDER ADDRESS
 400 010544  063737  002330  036542         ADD     IC,DPB.B+12     ;INCREMENT THE STARTING ADDRESS
```

```
    401 010552  023737  002326  036542          CMP     LC,DPB.B+12     ;EXCEEDING MAXIMUM ?
    402 010560  002353                          BGE     1$              ;BR IF NOT
    403 010562  000004                  EXIT10: SCOPE                   ;CALL SCOPE ROUTINE
```

```
                        .SBTTL  TIMING TESTS

                        ;;////////////////////////////////////////////////////////////////
                        ;*THE TIMING TESTS WILL ENSURE THAT THOSE FUNCTIONS BEING
                        ;*TIMED ARE WITHIN THE TOLERANCES SPECIFIED IN THE 'RM80
                        ;*ENGINEERING SPECIFICATIONS".
                        ;*THE SEEK TIMING WILL BE PERFORMED USING EXPLICIT SEEK
                        ;*OPERATIONS. AT THE COMPLETION OF EACH OF THE TIMING
                        ;*TESTS THE MINIMUM, MAXIMUM AND AVERAGE TIMES WILL BE
                        ;*TYPED.
                        ;;////////////////////////////////////////////////////////////////


                        ;;****************************************************************
                        ;*TEST 11       ROTATIONAL SPEED TIMING TEST
                        ;*THIS TEST WILL START A SEARCH TO CYLINDER 'FC', TRACK 'FT',
                        ;*SECTOR 'FS'.  AS SOON AS THE INTERRUPT OCCURS, THE GO BIT
                        ;*IS SET AGAIN AND THE OPERATION IS TIMED.  THIS PROCEDURE
                        ;*IS REPEATED 10 TIMES THEN THE AVERAGE TIME IS CALCULATED
                        ;*AND CHECKED TO ENSURE IT IS WITHIN TOLERANCE:
                        ;*      LO LIMIT (16.67MS - 4.3%) = 15.97MS @ 50/60 HZ.
                        ;*      HI LIMIT (16.67MS + 3.5%) = 17.27MS @ 50/60 HZ.
                        ;;****************************************************************
010564                  TST11:
010564  000240                  NOP
010566  033737  001602  001332        BIT     BITS+<11*2>,TSTNMS      ;DO THIS TEST?
010574  001002                  BNE     .+6             ;BR IF YES
010576  000137  011426          JMP     TST12           ;NO--JUMP TO TEST12

010602  012737  000011  001116        MOV     #11,$TSTNM      ;SET TEST #11 AND CLEAR ($ERFLG)
010610  004737  023734          JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
010614  012737  011102  001124        MOV     #TEST11,$LPERR  ;SETUP THE LOOP ON ERROR ADDRESS
010622  013737  002322  001220        MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
010630  112737  000031  001131        MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
010636  012737  000011  001240        MOV     #11,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
24
010644  032777  010000  170302        BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
010652  001406                  BEQ     .+16            ;BR IF YES
010654  104401  037265          TYPE    ,MSGTST         ;TYPE 'TEST'
010660  013746  001240          MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
010664  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
010666     003                  .BYTE   3               ;;TYPE 3 DIGIT(S)
010667     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS

25 010670  005737  001342          TST     CLKSTA          ;KW11-P CLOCK?
26 010674  003002                  BGT     1$              ;YES--START TEST
29 010676  000137  011424          JMP     EXIT11          ;NO--JUMP TO EXIT11
30 010702  012737  010702  001122  1$:     MOV     #.,$LPADR       ;SETUP LOOP ADDRESS
31 010710  004037  025500          JSR     R0,SRCH00       ;DO A MASSBUS INIT & RECAL
32 010714  000402                  BR      2$              ;RETURN HERE IF NO ERROR
54 010716  000137  011424          JMP     EXIT11          ;RETURN HERE IF ERROR
   010722  012737  010722  001122  2$:     MOV     #.,$LPADR       ;ERROR LOOP ADDRESS
   010730  112737  000105  036510        MOVB    #SEEK,DPB.A+2   ;SEEK=COMMAND
   010736  005037  036516          CLR     DPB.A+10        ;USE TRACK 0 & SECTOR 0
   010742  013737  002324  036520        MOV     FC,DPB.A+12     ;STARTING CYLINDER
   010750  012737  011424  001350        MOV     #EXIT11,BYPASS  ;GO TO EXIT11 IF ERROR
   010756  004037  024076          JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
```

```
T11     ROTATIONAL SPEED TIMING TEST

        010762  013764  002324  000034          MOV     FC,RMDC(R4)     ;FC
        010770  013746  002340                  MOV     FS,-(SP)        ;FS
        010774  113766  002332  000001          MOVB    FT,1(SP)        ;FT
        011002  012664  000006                  MOV     (SP)+,RMDA(R4)  ;LOAD FT/FS
        011006  012737  011424  001222          MOV     #EXIT11,$ESCAPE ;;ESCAPE TO EXIT11 ON ERROR
        011014  005005                          CLR     R5              ;COUNT UP

    55                                  ;SETUP PARAMETER TABLE FOR RM80
    56 011016  010046                          MOV     R0,-(SP)        ;SAVE R0
    57 011020  113700  036574                  MOVB    DTADPB,R0       ;DRIVE ADDRESS
    58 011024  032737  000100  001314          BIT     #SW06,C.SWR     ;CHECK CONTROL SWR FOR 60 HZ.
    59 011032  001411                          BEQ     3$              ;BR IF YES
    71 011034  012703  001650                  MOV     #T11B,R3        ;LOAD 50 HZ. TABLE FOR RM80
       011040  012737  001650  011414          MOV     #T11B,TP50
       011046  012737  001716  011422          MOV     #SP11B,TPS50
       011054  000410                          BR      4$              ;EXIT
       011056  012737  001640  011376  3$:     MOV     #T11A,TP60      ;LOAD 60 HZ. TABLE FOR RM80
       011064  012703  001640                  MOV     #T11A,R3
       011070  012737  001710  011404          MOV     #SP11A,TPS60
       011076  012600                  4$:     MOV     (SP)+,R0        ;RESTORE R0
       011100  000240                          NOP                     ;EXIT
       011102                          TEST11:
    72 011102  012706  001100                  MOV     #STACK,SP       ;SETUP STACK
    73 011106  012701  000012                  MOV     #10.,R1         ;TIME 10 SEARCHES
    74 011112  004737  025664                  JSR     PC,STRTMR       ;INITIALIZE THE TIMERS
    75 011116  012777  011304  170404          MOV     #4$,@PKV        ;SETUP VECTOR IN CASE OF OVERFLOW
    76 011124  012777  025662  020236          MOV     #DORTI,@RMVEC   ;SETUP RM80 VECTOR
    77 011132  005077  170400          1$:     CLR     @PKB            ;START COUNTING AT ZERO
    78 011136  012777  000131  170370          MOV     #131,@PKCS      ;INT.EN., COUNT UP AT 100K HZ RATE
    79 011144  012714  000131                  MOV     #SEARCH,(R4)    ;START A SEARCH
    80 011150  000001                          WAIT                    ;WAIT ON INTERRUPT
    81 011152  042777  000101  170354          BIC     #101,@PKCS      ;STOP THE CLOCK
    82 011160  032764  040000  000012          BIT     #ERR,RMDS(R4)   ;ERROR?
    83 011166  001411                          BEQ     2$              ;NO--BRANCH
    84 011170  104412                          SAVREG                  ;SAVE R0-R5
       011172  012702  036574                  MOV     #DTADPB,R2      ;DPB POINTER
       011176  004737  035744                  JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
       011202  004737  023146                  JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
       011206  104413                          RESREG                  ;RESTORE R0-R5
    85 011210  104017                          EMT     17
    86 011212  005077  170320          2$:     CLR     @PKB            ;START THE COUNT AT ZERO
    87 011216  012714  000131                  MOV     #SEARCH,(R4)    ;START A SEARCH
    88 011222  012777  000131  170304          MOV     #131,@PKCS      ;START THE CLOCK
    89 011230  000001                          WAIT                    ;WAIT ON INTERRUPT
    90 011232  042777  000101  170274          BIC     #101,@PKCS      ;STOP THE CLOCK
    91 011240  032764  040000  000012          BIT     #ERR,RMDS(R4)   ;IS 'ERR=1'?
    92 011246  001411                          BEQ     3$              ;NO--BRANCH
    93 011250  104412                          SAVREG                  ;SAVE R0-R5
       011252  012702  036574                  MOV     #DTADPB,R2      ;DPB POINTER
       011256  004737  035744                  JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
       011262  004737  023146                  JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
       011266  104413                          RESREG                  ;RESTORE R0-R5
    94 011270  104017                          EMT     17
    95 011272  004737  025726          3$:     JSR     PC,COUNT        ;UPDATE THE COUNT
    96 011276  005301                          DEC     R1              ;DONE?
    97 011300  003314                          BGT     1$              ;NO--BRANCH
    98 011302  000420                          BR      5$              ;YES--GO TO THE EXIT
```

```
     99 011304  042777  000101  170222  4$:     BIC     #101,@PKCS      ;STOP THE CLOCK
    100 011312  005037  177776                  CLR     PS              ;DROP THE PRIORITY
    101 011316  012600                          MOV     (SP)+,R0        ;PC OF WAIT+2
    102 011320  005726                          TST     (SP)+           ;POP THE PS FROM THE STACK
    103 011322  104412                          SAVREG                  ;SAVE R0-R5
        011324  012702  036574                  MOV     #DTADPB,R2      ;DPB POINTER
        011330  004737  035744                  JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
        011334  004737  023146                  JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
        011340  104413                          RESREG                  ;RESTORE R0-R5
    104 011342  104020                          EMT     20
    105 011344  004737  023146          5$:     JSR     PC,CNTCLR       ;GO CLEAR MASSBUS CONTROLLER
    106
    107 011350  004737  023240                  JSR     PC,ST.CLK       ;INITIALIZE THE CLOCK
    108 011354  012777  033526  020006          MOV     #ISR,@RMVEC     ;RESTORE RH/RM INT. VECTOR
    109 011362  032737  000100  001314          BIT     #SW06,C.SWR     ;60 HZ?
    128 011370  001007                          BNE     EXIT.A          ;NO-BRANCH
        011372  004037  026166                  JSR     R0,TYPTIM       ;TYPE THE TIMING
        011376  001640                  TP60:   T11A                    ;TABLE ADDRESS
        011400  004037  026060                  JSR     R0,SPTYP        ;TYPE THE SPEC
        011404  001710                  TPS60:  SP11A
        011406  000406                          BR      EXIT11          ;EXIT

        011410  004037  026166          EXIT.A: JSR     R0,TYPTIM       ;TYPE THE TIMING
        011414  001650                  TP50:   T11B                    ;TABLE ADDRESS
        011416  004037  026060                  JSR     R0,SPTYP        ;
        011422  001716                  TPS50:  SP11B                   ;
    129 011424  000004                  EXIT11: SCOPE                   ;CALL SCOPE ROUTINE
    130
    139
    140                                 ;;*****************************************************************
                                        ;*TEST 12       ONE CYLINDER SEEK TIMING TEST
                                        ;*THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
                                        ;*CYLINDER BY ONE UNTIL THE INCREMENT IS GREATER THAN THE
                                        ;*CYLINDER 'LC', THEN REVERSE SEEK TO CYLINDER 'FC'.  THE
                                        ;*TIME TO PERFORM EACH SEEK IS CHECKED TO ENSURE IT DOES NOT
                                        ;*EXCEED THE MAXIMUM TIME PERMITTED FOR A ONE CYLINDER SEEK.
                                        ;*MAXIMUM TIME IS 6.0 MS
                                        ;;*****************************************************************
        011426                          TST12:
        011426  000240                          NOP
        011430  033737  001604  001332          BIT     BITS+<12*2>,TSTNMS      ;DO THIS TEST?
        011436  001002                          BNE     .+6             ;BR IF YES
        011440  000137  012130                  JMP     TST13           ;NO--JUMP TO TEST13

        011444  012737  000012  001116          MOV     #12,$TSTNM      ;SET TEST #12 AND CLEAR ($ERFLG)
        011452  004737  023734                  JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
        011456  012737  011636  001124          MOV     #TEST12,$LPERR  ;SETUP THE LOOP ON ERROR ADDRESS
        011464  013737  002322  001220          MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
        011472  112737  000031  001131          MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
        011500  012737  000012  001240          MOV     #12,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
    141
        011506  032777  010000  167440          BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
        011514  001406                          BEQ     .+16            ;BR IF YES
        011516  104401  037265                  TYPE    ,MSGTST         ;TYPE 'TEST'
        011522  013746  001240                  MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
        011526  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
        011530     003                          .BYTE   3               ;;TYPE 3 DIGIT(S)
```

T12 ONE CYLINDER SEEK TIMING TEST

```
011531  000                             .BYTE   0               ;;SUPPRESS LEADING ZEROS

142 011532  005737  001342              TST     CLKSTA          ;KW11-P CLOCK?
143 011536  003002                      BGT     1$              ;YES--START TEST
146 011540  000137  012126              JMP     EXIT12          ;NO--JUMP TO EXIT12
147 011544  012737  011544  001122  1$: MOV     #.,$LPADR       ;SETUP THE LOOP ADDRESS
148 011552  004037  025500              JSR     R0,SRCH00       ;DO A MASSBUS INIT. AND RECAL
149 011556  000402                      BR      2$              ;NO ERROR RETURN
161 011560  000137  012126              JMP     EXIT12          ;ERROR RETURN--SCOPE LOOP CALL
    011564  012737  011564  001122  2$: MOV     #.,$LPADR       ;ERROR LOOP ADDRESS
    011572  112737  000105  036510      MOVB    #SEEK,DPB.A+2   ;SEEK=COMMAND
    011600  005037  036516              CLR     DPB.A+10        ;USE TRACK 0 & SECTOR 0
    011604  013737  002324  036520      MOV     FC,DPB.A+12     ;STARTING CYLINDER
    011612  012737  012126  001350      MOV     #EXIT12,BYPASS  ;GO TO EXIT12 IF ERROR
    011620  004037  024076              JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
    011624  012703  001660              MOV     #T12,R3         ;PARAMETER POINTER
    011630  012737  012126  001222      MOV     #EXIT12,$ESCAPE ;;ESCAPE TO EXIT12 ON ERROR
    011636                          TEST12:
162 011636  012706  001100              MOV     #STACK,SP       ;SETUP STACK
163 011642  013737  002324  036606      MOV     FC,DTADPB+12    ;START WITH BEGINNING CYLINDER
164 011650  005237  036606              INC     DTADPB+12       ;INCREMENT THE BEGINNING CYLINDER
165 011654  005005                      CLR     R5              ;SET THE UP/DOWN SWITCH TO UP
166 011656  004737  025664              JSR     PC,STRTMR       ;INITIALIZE THE TIMERS
167 011662  012777  012034  167640      MOV     #5$,@PKV        ;SETUP INCASE OF OVERFLOW
168 011670  012777  025662  017472      MOV     #DORTI,@RMVEC   ;SET RM80 VECTOR
169 011676  005077  167634          1$: CLR     @PKB            ;START THE COUNTER AT ZERO
170 011702  013764  036606  000034      MOV     DTADPB+12,RMDC(R4)      ;LOAD DESIRED CYLINDER
171 011710  012714  000105              MOV     #SEEK,(R4)      ;START A SEEK
172 011714  012777  000131  167612      MOV     #131,@PKCS      ;START THE CLOCK
173 011722  000001                      WAIT                    ;WAIT ON INTERRUPT
174 011724  042777  000101  167602      BIC     #101,@PKCS      ;STOP THE CLOCK
175 011732  032764  040000  000012      BIT     #ERR,RMDS(R4)   ;ANY DISK ERRORS?
176 011740  001411                      BEQ     2$              ;NO--BRANCH
177 011742  104412                      SAVREG                  ;SAVE R0-R5
    011744  012702  036574              MOV     #DTADPB,R2      ;DPB POINTER
    011750  004737  035744              JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
    011754  004737  023146              JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
    011760  104413                      RESREG                  ;RESTORE R0-R5
178 011762  104017                      EMT     17
179 011764  004737  025726          2$: JSR     PC,COUNT        ;COUNT THIS SEEKS TIME
180 011770  005705                      TST     R5              ;UP OR DOWN?
181 011772  001011                      BNE     4$              ;DOWN--BRANCH
182 011774  005237  036606          3$: INC     DTADPB+12       ;MOVE TO NEXT CYLINDER
183 012000  023737  036606  002326      CMP     DTADPB+12,LC    ;OUT OF CYLINDERS?
184 012006  002733                      BLT     1$              ;NO--GO DO THE NEXT SEEK
185 012010  012705  177777              MOV     #-1,R5          ;SET UP/DOWN SWITCH TO DOWN
186 012014  000730                      BR      1$              ;GO DO THE NEXT SEEK
187 012016  005337  036606          4$: DEC     DTADPB+12       ;MOVE TO NEXT CYLINDER
188 012022  023737  036606  002324      CMP     DTADPB+12,FC    ;OUT OF CYLINDERS?
189 012030  003322                      BGT     1$              ;NO--GO DO THE NEXT SEEK
190 012032  000420                      BR      6$              ;GO TO THE EXIT
191 012034  042777  000101  167472  5$: BIC     #101,@PKCS      ;STOP THE CLOCK
192 012042  005037  177776              CLR     PS              ;DROP THE PRIORITY
193 012046  012600                      MOV     (SP)+,R0        ;PC OF WAIT+2
194 012050  005726                      TST     (SP)+           ;POP THE PS FROM THE STACK
195 012052  104412                      SAVREG                  ;SAVE R0-R5
    012054  012702  036574              MOV     #DTADPB,R2      ;DPB POINTER
```

```
T12     ONE CYLINDER SEEK TIMING TEST

        012060  004737  035744                  JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
        012064  004737  023146                  JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
        012070  104413                          RESREG                  ;RESTORE R0-R5
    196 012072  104020                          EMT     20
    197 012074  004737  023146          6$:     JSR     PC,CNTCLR       ;GO CLEAR MASSBUS CONTROLLER
    198 012100  004737  023240                  JSR     PC,ST.CLK       ;INITIALIZE THE CLOCK
    199 012104  012777  033526  017256          MOV     #ISR,@RMVEC     ;RESTORE RH/RM INT. VECTOR
    204 012112  004037  026166                  JSR     R0,TYPTIM       ;GO TYPE THE TIMES
        012116  001660                          T12                     ;POINTER
        012120  004037  026060                  JSR     R0,SPTYP
        012124  001724                          SP12
    205 012126  000004                  EXIT12: SCOPE                   ;CALL SCOPE ROUTINE
    206
    220
    221                                 ;;*****************************************************************
                                        ;*TEST 13       AVERAGE SEEK TIMING TEST
                                        ;*THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 'FC' TO
                                        ;*CYLINDER 'LC', THEN A REVERSE  SEEK FROM CYLINDER 'LC' TO
                                        ;*CYLINDER 'FC'.  BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE THEY
                                        ;*ARE WITHIN THE TOLERANCE ALLOWED FOR THE AVERAGE SEEK.
                                        ;*THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL OF 256 SEEKS).
                                        ;*MAXIMUM TIME IS 25.0 MS
                                        ;*
                                        ; THERE ARE NO SPECIFICATIONS GIVEN  FOR AN AVERAGE SEEK TIME
                                        ; ON  THIS PARTICULAR  DRIVE.  THEREFORE, THIS TEST SHOULD BE
                                        ; USED FOR REFERENCE ONLY.
                                        ;*
                                        ;;*****************************************************************
        012130                          TST13:
        012130  000240                          NOP
        012132  033737  001606  001332          BIT     BITS+<13*2>,TSTNMS      ;DO THIS TEST?
        012140  001002                          BNE     .+6             ;BR IF YES
        012142  000137  012670                  JMP     TST14           ;NO--JUMP TO TEST14

        012146  012737  000013  001116          MOV     #13,$TSTNM      ;SET TEST #13 AND CLEAR ($ERFLG)
        012154  004737  023734                  JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
        012160  012737  012340  001124          MOV     #TEST13,$LPERR  ;SETUP THE LOOP ON ERROR ADDRESS
        012166  013737  002322  001220          MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
        012174  112737  000031  001131          MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
        012202  012737  000013  001240          MOV     #13,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
    222
        012210  032777  010000  166736          BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
        012216  001406                          BEQ     .+16            ;BR IF YES
        012220  104401  037265                  TYPE    ,MSGTST         ;TYPE 'TEST'
        012224  013746  001240                  MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
        012230  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
        012232     003                          .BYTE   3               ;;TYPE 3 DIGIT(S)
        012233     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS

    223 012234  005737  001342                  TST     CLKSTA          ;KW11-P CLOCK?
    224 012240  003002                          BGT     1$              ;YES--START TEST
    240 012242  000137  012666                  JMP     EXIT13          ;NO--JUMP TO EXIT13
        012246  012737  012246  001122  1$:     MOV     #.,$LPADR       ;SET THE LOOP ADDRESS
        012254  004037  025500                  JSR     R0,SRCH00       ;DO A MASSBUS INIT & RECAL
        012260  000402                          BR      2$              ;RETURN HERE IF NO ERROR
        012262  000137  012666                  JMP     EXIT13          ;RETURN HERE ON ERROR
        012266  012737  012266  001122  2$:     MOV     #.,$LPADR       ;ERROR LOOP ADDRESS
```

```
T13     AVERAGE SEEK TIMING TEST

012274  112737  000105  036510          MOVB    #SEEK,DPB.A+2   ;SEEK=COMMAND
012302  005037  036516                  CLR     DPB.A+10        ;USE TRACK 0 & SECTOR 0
012306  013737  002324  036520          MOV     FC,DPB.A+12     ;STARTING CYLINDER
012314  012737  012666  001350          MOV     #EXIT13,BYPASS  ;GO TO EXIT13 IF ERROR
012322  004037  024076                  JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
012326  012703  001670                  MOV     #T13,R3         ;PARAMETER POINTER
012332  012737  012666  001222          MOV     #EXIT13,$ESCAPE ;;ESCAPE TO EXIT13 ON ERROR
012340                          TEST13:
241 012340  012706  001100              MOV     #STACK,SP       ;SETUP STACK
242 012344  012701  000200              MOV     #128.,R1        ;REPEAT '"FC'-'LC'-'FC"' 128 TIMES
243 012350  004737  025664              JSR     PC,STRTMR       ;INIT. THE COUNTERS
244 012354  012777  012574  167146      MOV     #5$,@PKV        ;SET UP VECTOR IN CASE OF OVERFLOW
245 012362  012777  025662  017000      MOV     #DORTI,@RMVEC   ;SETUP RM80 VECTOR
246 012370  005077  167142          1$: CLR     @PKB            ;START COUNT AT ZERO
247 012374  013764  002326  000034      MOV     LC,RMDC(R4)     ;'MIDDLE' CYLINDER
248 012402  012764  000105  000000      MOV     #SEEK,RMCS1(R4) ;START A SEEK
249 012410  012777  000131  167116      MOV     #131,@PKCS      ;START THE CLOCK
250 012416  000001                      WAIT                    ;WAIT ON INTERRUPT
251 012420  042777  000101  167106      BIC     #101,@PKCS      ;STOP CLOCK
252 012426  032764  040000  000012      BIT     #ERR,RMDS(R4)   ;ERR=1?
253 012434  001411                      BEQ     2$              ;NO--BRANCH
254 012436  104412                      SAVREG                  ;SAVE R0-R5
012440  012702  036574                  MOV     #DTADPB,R2      ;DPB POINTER
012444  004737  035744                  JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
012450  004737  023146                  JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
012454  104413                      RESREG                  ;RESTORE R0-R5
255 012456  104017                      EMT     17
256 012460  005005                  2$: CLR     R5              ;SET UP/DOWN SWITCH TO UP
257 012462  004737  025726              JSR     PC,COUNT        ;UPDATE THE COUNT
258 012466  005077  167044              CLR     @PKB            ;START THE COUNT AT ZERO
259 012472  013764  002324  000034      MOV     FC,RMDC(R4)     ;BEGINNING CYLINDER
260 012500  012764  000105  000000      MOV     #SEEK,RMCS1(R4) ;START A SEEK
261 012506  012777  000131  167020      MOV     #131,@PKCS      ;START THE CLOCK
262 012514  000001                      WAIT                    ;WAIT ON INTERRUPT
263 012516  042777  000101  167010      BIC     #101,@PKCS      ;STOP THE CLOCK
264 012524  032764  040000  000012      BIT     #ERR,RMDS(R4)   ;ERR=1?
265 012532  001411                      BEQ     3$              ;NO--BRANCH
266 012534  104412                      SAVREG                  ;SAVE R0-R5
012536  012702  036574                  MOV     #DTADPB,R2      ;DPB POINTER
012542  004737  035744                  JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
012546  004737  023146                  JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
012552  104413                      RESREG                  ;RESTORE R0-R5
267 012554  104017                      EMT     17
268 012556  012705  177777          3$: MOV     #-1,R5          ;SET UP/DOWN SWITCH TO DOWN
269 012562  004737  025726              JSR     PC,COUNT        ;UPDATE THE COUNT
270 012566  005301                      DEC     R1              ;DONE?
271 012570  003277                      BGT     1$              ;NO--BRANCH
272 012572  000420                      BR      6$              ;YES--EXIT
273 012574  042777  000101  166732  5$: BIC     #101,@PKCS      ;STOP THE CLOCK
274 012602  005037  177776              CLR     PS              ;DROP THE PRIORITY
275 012606  012600                      MOV     (SP)+,R0        ;PC OF WAIT+2
276 012610  005726                      TST     (SP)+           ;POP THE PS FROM THE STACK
277 012612  104412                      SAVREG                  ;SAVE R0-R5
012614  012702  036574                  MOV     #DTADPB,R2      ;DPB POINTER
012620  004737  035744                  JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
012624  004737  023146                  JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
012630  104413                      RESREG                  ;RESTORE R0-R5
```

```
    278 012632  104020                          EMT     20
    279 012634  004737  023146          6$:     JSR     PC,CNTCLR       ;GO CLEAR MASSBUS CONTROLLER
    280 012640  004737  023240                  JSR     PC,ST.CLK       ;INITIALIZE THE CLOCK
    281 012644  012777  033526  016516          MOV     #ISR,@RMVEC     ;RESTORE RH/RM INT. VECTOR
    286 012652  004037  026166                  JSR     R0,TYPTIM       ;GO TYPE THE TIMES
        012656  001670                          T13                     ;POINTER
        012660  004037  026060                  JSR     R0,SPTYP
        012664  001732                          SP13
    287 012666  000004                  EXIT13: SCOPE                   ;CALL SCOPE ROUTINE
    288
    304
    305                                 ;;*****************************************************************
                                        ;*TEST 14       MAXIMUM SEEK TIMING TEST
                                        ;*THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 'FC' TO
                                        ;*CYLINDER 'LC', THEN A REVERSE SEEK FROM CYLINDER 'LC' TO
                                        ;*CYLINDER 'FC'.  BOTH SEEKS ARE TIMED, BUT ONLY THE FORWARD SEEKS
                                        ;*ARE CHECKED TO ENSURE THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE
                                        ;*MAXIMUM SEEK TIME. THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL
                                        ;*OF 256. SEEKS). MAXIMUM (FORWARD) TIME IS 50.0 MS
                                        ;*
                                        ; THERE IS  NO SPECIFICATION GIVEN FOR THE MAXIMUM  REVERSE SEEK TIME
                                        ; ON THIS PARTICULAR DRIVE.  THEREFORE,  ANY REVERSE SEEK TIMES ABOVE
                                        ; THE MAXIMUM TIME OF 50.0 MS WILL NOT BE TYPED IN THE TIMING REPORT.
                                        ; HOWEVER,  THE TIMING REPORT WILL  STILL TYPE THE MINIMUM,  MAXIMUM,
                                        ; AND AVERAGE TIMES FOR THE REVERSE SEEKS.
                                        ;*
                                        ;;*****************************************************************
        012670                          TST14:
        012670  000240                          NOP
        012672  033737  001610  001332          BIT     BITS+<14*2>,TSTNMS      ;DO THIS TEST?
        012700  001002                          BNE     .+6             ;BR IF YES
        012702  000137  013440                  JMP     TST15           ;NO--JUMP TO TEST15

        012706  012737  000014  001116          MOV     #14,$TSTNM      ;SET TEST #14 AND CLEAR ($ERFLG)
        012714  004737  023734                  JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
        012720  012737  013100  001124          MOV     #TEST14,$LPERR  ;SETUP THE LOOP ON ERROR ADDRESS
        012726  013737  002322  001220          MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
        012734  112737  000031  001131          MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
        012742  012737  000014  001240          MOV     #14,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
    306
        012750  032777  010000  166176          BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
        012756  001406                          BEQ     .+16            ;BR IF YES
        012760  104401  037265                  TYPE    ,MSGTST         ;TYPE 'TEST'
        012764  013746  001240                  MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
        012770  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
        012772     003                          .BYTE   3               ;;TYPE 3 DIGIT(S)
        012773     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS

    307 012774  005737  001342                  TST     CLKSTA          ;KW11-P CLOCK
    308 013000  003002                          BGT     1$              ;YES--START TEST
    311 013002  000137  013436                  JMP     EXIT14          ;NO--JUMP TO EXIT14
    312 013006  012737  013006  001122  1$:     MOV     #.,$LPADR       ;SETUP THE LOOP ADDRESS
    313 013014  004037  025500                  JSR     R0,SRCH00       ;DO A MASSBUS INIT & RECAL
    314 013020  000402                          BR      2$              ;RETURN HERE IF NO ERROR
    326 013022  000137  013436                  JMP     EXIT14          ;RETURN HERE ON ERROR
        013026  012737  013026  001122  2$:     MOV     #.,$LPADR       ;ERROR LOOP ADDRESS
        013034  112737  000105  036510          MOVB    #SEEK,DPB.A+2   ;SEEK=COMMAND
```

T14 MAXIMUM SEEK TIMING TEST

```
013042  005037  036516                  CLR     DPB.A+10        ;USE TRACK 0 & SECTOR 0
013046  013737  002324  036520          MOV     FC,DPB.A+12     ;STARTING CYLINDER
013054  012737  013436  001350          MOV     #EXIT14,BYPASS  ;GO TO EXIT14 IF ERROR
013062  004037  024076                  JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
013066  012703  001700                  MOV     #T14,R3         ;PARAMETER POINTER
013072  012737  013436  001222          MOV     #EXIT14,$ESCAPE ;;ESCAPE TO EXIT14 ON ERROR
013100                          TEST14:
327 013100  012706  001100              MOV     #STACK,SP       ;SETUP STACK
328 013104  012701  000200              MOV     #128.,R1        ;REPEAT '"FC'-'LC'-'FC"' 128 TIMES
329 013110  004737  025664              JSR     PC,STRTMR       ;INIT. THE TIMERS
330 013114  012777  013344  166406      MOV     #5$,@PRV        ;SETUP VECTOR IN CASE OF OVERFLOW
331 013122  012777  025662  016240      MOV     #CORTI,@RMVEC   ;SETUP RM80 VECTOR
332 013130  005077  166402          1$: CLR     @PKB            ;START COUNTING FROM ZERO
333 013134  013764  002326  000034      MOV     LC,RMDC(R4)     ;MAXIMUM CYLINDER
334 013142  012764  000105  000000      MOV     #SEEK,RMCS1(R4) ;START A SEEK
335 013150  012777  000131  166356      MOV     #131,@PKCS      ;START THE CLOCK
336 013156  000001                      WAIT                    ;WAIT ON INTERRUPT
337 013160  042777  000101  166346      BIC     #101,@PKCS      ;STOP THE CLOCK
338 013166  032764  040000  000012      BIT     #ERR,RMDS(R4)   ;ERR=1?
339 013174  001411                      BEQ     2$              ;NO--BRANCH
340 013176  104412                      SAVREG                  ;SAVE R0-R5
    013200  012702  036574              MOV     #DTADPB,R2      ;DPB POINTER
    013204  004737  035744              JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
    013210  004737  023146              JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
    013214  104413                      RESREG                  ;RESTORE R0-R5
341 013216  104017                      EMT     17
342 013220  005005                  2$: CLR     R5              ;SET THE UP/DOWN SWITCH TO UP
343 013222  004737  025726              JSR     PC,COUNT        ;UP THE COUNT
344 013226  005077  166304              CLR     @PKB            ;START COUNT AT ZERO
345 013232  013764  002324  000034      MOV     FC,RMDC(R4)     ;BEGINNING CYLINDER
346 013240  012764  000105  000000      MOV     #SEEK,RMCS1(R4) ;START A SEEK
347 013246  012777  000131  166260      MOV     #131,@PKCS      ;START THE CLOCK
348 013254  000001                      WAIT                    ;WAIT ON INTERRUPT
349 013256  042777  000101  166250      BIC     #101,@PKCS      ;STOP THE CLOCK
350 013264  032764  040000  000012      BIT     #ERR,RMDS(R4)   ;'ERR'=1?
351 013272  001411                      BEQ     3$              ;NO--BRANCH
352 013274  104412                      SAVREG                  ;SAVE R0-R5
    013276  012702  036574              MOV     #DTADPB,R2      ;DPB POINTER
    013302  004737  035744              JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
    013306  004737  023146              JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
    013312  104413                      RESREG                  ;RESTORE R0-R5
353 013314  104017                      EMT     17
354 013316  012705  177777          3$: MOV     #-1,R5          ;SET THE UP/DOWN SWITCH TO DOWN
355 013322  004737  025726              JSR     PC,COUNT        ;UPDATE THE COUNT
356 013326  005037  001416              CLR     TIM.DN+2        ;FORGET ABOUT # OF SEEKS BELOW MINIMUM TIME
357 013332  005037  001422              CLR     TIM.DN+6        ;FORGET ABOUT # OF SEEKS ABOVE MAXIMUM TIME
358 013336  005301                      DEC     R1              ;DONE?
359 013340  003273                      BGT     1$              ;NO--BRANCH
360 013342  000420                      BR      6$              ;YES--EXIT
361 013344  042777  000101  166162  5$: BIC     #101,@PKCS      ;STOP THE CLOCK
362 013352  005037  177776              CLR     PS              ;DROP THE PRIORITY
363 013356  012600                      MOV     (SP)+,R0        ;PC OF WAIT+2
364 013360  005726                      TST     (SP)+           ;POP THE PS FROM THE STACK
365 013362  104412                      SAVREG                  ;SAVE R0-R5
    013364  012702  036574              MOV     #DTADPB,R2      ;DPB POINTER
    013370  004737  035744              JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
    013374  004737  023146              JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
```

```
013400  104413                          RESREG                  ;RESTORE R0-R5
366 013402  104020                      EMT     20
367 013404  004737  023146      6$:     JSR     PC,CNTCLR       ;GO CLEAR MASSBUS CONTROLLER
368 013410  004737  023240              JSR     PC,ST.CLK       ;INITIALIZE THE CLOCK
369 013414  012777  033526  015746      MOV     #ISR,@RMVEC     ;RESTORE RH/RM INT. VECTOR
374 013422  004037  026166              JSR     R0,TYPTIM       ;GO TYPE THE TIMES
    013426  001700                      T14                     ;POINTER
    013430  004037  026060              JSR     R0,SPTYP
    013434  001740                      SP14
375 013436  000004              EXIT14: SCOPE                   ;CALL SCOPE ROUTINE
376
387
388                                     ;*****************************************************************
                                        ;*TEST 15       AVERAGE SEEK TIME MEASUREMENT TEST
                                        ;*THIS TEST MEASURES THE AVERAGE SEEK TIME BY DOING ALL UNIQUE SEEKS
                                        ;*BETWEEN 'FC' AND 'LC',  AND  MEASURING THE  SEEK TIMES.  THE TOTAL
                                        ;*SEEK TIME  FOR ALL  SEEKS IS THEN DIVIDED  BY THE  NUMBER OF SEEKS
                                        ;*PERFORMED TO  PROVIDE AN AVERAGE SEEK TIME VALUE FOR  THE RANGE OF
                                        ;*SEEKS TIMED.  MAXIMUM TIME IS 25.0 MS
                                        ;*
                                        ; WARNING - THIS TEST TAKES APPROXIMATELY 3.0 HOURS TO RUN
                                        ;*
                                        ;*****************************************************************
    013440                              TST15:
    013440  000240                      NOP
    013442  033737  001612  001332      BIT     BITS+<15*2>,TSTNMS      ;DO THIS TEST?
    013450  001002                      BNE     .+6             ;BR IF YES
    013452  000137  015142              JMP     $EOP            ;NO--GO TO THE END OF THE PROGRAM

    013456  012737  000015  001116      MOV     #15,$TSTNM      ;SET TEST #15 AND CLEAR ($ERFLG)
    013464  004737  023734              JSR     PC,LODPRM       ;LOAD THE PARMETERS FOR THE TEST
    013470  012737  013644  001124      MOV     #TEST15,$LPERR  ;SETUP THE LOOP ON ERROR ADDRESS
    013476  013737  002322  001220      MOV     RPT,$TIMES      ;GET THE ITERATION COUNT
    013504  112737  000031  001131      MOVB    #25.,$ERMAX     ;MAX ERRORS ALLOWED FOR TEST
    013512  012737  000015  001240      MOV     #15,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
389
    013520  032777  010000  165426      BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
    013526  001406                      BEQ     .+16            ;BR IF YES
    013530  104401  037265              TYPE    ,MSGTST         ;TYPE 'TEST'
    013534  013746  001240              MOV     $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPEOUT
    013540  104403                      TYPOS                   ;;GO TYPE--OCTAL ASCII
    013542     003                      .BYTE   3               ;;TYPE 3 DIGIT(S)
    013543     000                      .BYTE   0               ;;SUPPRESS LEADING ZEROS

390 013544  005737  001342              TST     CLKSTA          ;KW11-P CLOCK?
391 013550  003002                      BGT     1$              ;YES--START TEST
394 013552  000137  015140              JMP     EXIT15          ;NO--JUMP TO EXIT15
395 013556  012737  013556  001122  1$: MOV     #.,$LPADR       ;SETUP THE LOOP ADDRESS
396 013564  004037  025500              JSR     R0,SRCH00       ;DO A MASSBUS INIT. AND RECAL
397 013570  000402                      BR      2$              ;NO ERROR RETURN
408 013572  000137  015140              JMP     EXIT15          ;ERROR RETURN--SCOPE LOOP CALL
    013576  012737  013576  001122  2$: MOV     #.,$LPADR       ;ERROR LOOP ADDRESS
    013604  112737  000105  036510      MOVB    #SEEK,DPB.A+2   ;SEEK=COMMAND
    013612  005037  036516              CLR     DPB.A+10        ;USE TRACK 0 & SECTOR 0
    013616  013737  002324  036520      MOV     FC,DPB.A+12     ;STARTING CYLINDER
    013624  012737  015140  001350      MOV     #EXIT15,BYPASS  ;GO TO EXIT15 IF ERROR
    013632  004037  024076              JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
```

```
                013636  012737  015140  001222          MOV     #EXIT15,$ESCAPE ;;ESCAPE TO EXIT15 ON ERROR
                013644                          TEST15:
    409         013644  012706  001100                  MOV     #STACK,SP       ;SETUP STACK
    410         013650  005037  001472                  CLR     SKTIM           ;CLEAR THE SEEK TIMER
    413         013654  005037  001474                  CLR     SKTIM+2         ;  ''    ''   ''    ''
                013660  005037  001476                  CLR     SKTIM+4         ;  ''    ''   ''    ''
                013664  005037  001500                  CLR     SKTIM+6         ;
    414         013670  005037  001502                  CLR     SKCNT           ;CLEAR THE SEEK COUNTER
    415         013674  005037  001504                  CLR     SKCNT+2         ;  ''    ''   ''    ''
    416         013700  005037  001506                  CLR     SKSIZ           ;SET SEEK LENGTH TO ZERO
    417         013704  005237  001506          1$:     INC     SKSIZ           ;INCREMENT THE SEEK LENGTH
    418         013710  005037  001510                  CLR     CYINC           ;CLEAR THE STARTING INCREMENT
    419         013714  023737  001510  001506  2$:     CMP     CYINC,SKSIZ     ;FINISHED WITH THIS PASS ?
    420         013722  001025                          BNE     3$              ;IF NE, NO
    421
    422                                         ;TYPE CURRENT SEEK SIZE AND NUMBER OF SEEKS, IF SW12=1
    423
    424         013724  032777  010000  165222          BIT     #SW12,@SWR      ;SEE IF SWITCH 12 SET
    425         013732  001764                          BEQ     1$              ;IF EQ, NO
    426         013734  104401  015046                  TYPE    ,16$            ;'SEEK SIZE='
    427         013740  013746  001506                  MOV     SKSIZ,-(SP)     ;MOVE IT TO THE STACK
    428         013744  004737  022212                  JSR     PC,$SB2D        ;CONVERT IT
    429         013750  004737  022442                  JSR     PC,$SUPRS       ;TYPE IT
    430         013754  104401  015062                  TYPE    ,17$            ;'NUMBER OF SEEKS='
    431         013760  012746  001502                  MOV     #SKCNT,-(SP)    ;SEEK COUNT POINTER
    432         013764  004737  022246                  JSR     PC,$DB2D        ;CONVERT TO ASCII
    433         013770  004737  022442                  JSR     PC,$SUPRS       ;TYPE IT
    434         013774  000743                          BR      1$              ;CONTINUE
    435
    436         013776  013737  001510  002346  3$:     MOV     CYINC,NC1       ;SETUP STARTING ADDRESS
    437         014004  063737  002324  002346          ADD     FC,NC1          ;ADD THE STARTING ADDRESS
    438         014012  013737  002346  002350          MOV     NC1,NC2         ;COPY THE STARTING ADDRESS
    439         014020  063737  001506  002350          ADD     SKSIZ,NC2       ;ENDING CYLINDER
    440         014026  023737  002350  002326          CMP     NC2,LC          ;SEEK IF ENDING CYLINDER TOO LARGE
    441         014034  101402                          BLOS    4$              ;IF LOS, NO
    442         014036  000137  014650                  JMP     14$             ;FINISHED WITH THIS PASS
    443
    444         014042  004737  023240          4$:     JSR     PC,ST.CLK       ;RE-INIT THE CLOCK
    445         014046  012777  033526  015314          MOV     #ISR,@RMVEC     ;RESTORE THE RH VECTOR
    446
    447         014054                          5$:
                014054  112737  000105  036510          MOVB    #SEEK,DPB.A+2   ;SEEK=COMMAND
    448         014062  005037  036516                  CLR     DPB.A+10        ;USE TRACK 0 & SECTOR 0
    449         014066  013737  002346  036520          MOV     NC1,DPB.A+12    ;STARTING CYLINDER
    452         014074  012737  015140  001350          MOV     #EXIT15,BYPASS  ;GO TO EXIT15 IF ERROR
    453         014102  004037  024076                  JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
    454
    455         014106  005077  165422          6$:     CLR     @PKCS           ;STOP THE CLOCK
    456         014112  012777  015102  165410          MOV     #18$,@PKV       ;SETUP CLOCK VECTOR IN CASE OF OVERFLOW
    457         014120  012777  025662  015242          MOV     #DORTI,@RMVEC   ;CHANGE RM80 VECTOR
    458
    459         014126  005077  165404          7$:     CLR     @PKB            ;START COUNT AT ZERO
    460         014132  013764  002350  000034          MOV     NC2,RMDC(R4)    ;LOAD CYLINDER ADDRESS
    461         014140  012764  000105  000000          MOV     #SEEK,RMCS1(R4) ;START A SEEK
    462         014146  012777  000131  165360          MOV     #131,@PKCS      ;START THE CLOCK
    463         014154  000001                          WAIT                    ;WAIT ON INTERRUPT
    464         014156  042777  000101  165350          BIC     #101,@PKCS      ;STOP CLOCK
```

```
    465 014164  032764  040000  000012          BIT     #BIT14,RMDS(R4) ;ERR=1?
    466 014172  001425                          BEQ     8$              ;NO--BRANCH
    467 014174  104412                          SAVREG                  ;SAVE R0-R5
        014176  012702  036574                  MOV     #DTADPB,R2      ;DPB POINTER
        014202  004737  035744                  JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
        014206  004737  023146                  JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
        014212  104413                          RESREG                  ;RESTORE R0-R5
    468 014214  104017                          EMT     17              ;REPORT THE ERROR
    469 014216  032737  040000  036702          BIT     #SKI,RM.REG+RMER2  ;SEE IF SKI SET
    470 014224  001410                          BEQ     8$              ;IF EQ, NO
    471 014226  004737  023240                  JSR     PC,ST.CLK       ;RE-INIT THE CLOCK
    472 014232  012777  033526  015130          MOV     #ISR,@RMVEC     ;RESTORE THE RH VECTOR
    473 014240  004037  024710                  JSR     R0,CALL.R       ;ISSUE A RECAL
    474 014244  000703                          BR      5$              ;TRY AGAIN
    475
    476 014246  067737  165266  001472  8$:     ADD     @PKC,SKTIM      ;ADD THE COUNTER VALUE
    479 014254  005537  001474                  ADC     SKTIM+2         ;ADD ANY CARRY
        014260  005537  001476                  ADC     SKTIM+4         ;ADD ANY CARRY
        014264  005537  001500                  ADC     SKTIM+6         ;ADD ANY CARRY
    480 014270  062737  000001  001502          ADD     #1,SKCNT        ;COUNT THE SEEK
    481 014276  005537  001504                  ADC     SKCNT+2         ;   "     "   "
    482
    483 014302  013737  002350  002346          MOV     NC2,NC1         ;GET NEW STARTING CYLINDER
    484 014310  063737  001506  002350          ADD     SKSIZ,NC2       ;NEW ENDING CYLINDER
    485 014316  023737  002350  002326          CMP     NC2,LC          ;FINISHED IN THIS DIRECTION ?
    486 014324  101700                          BLOS    7$              ;IF LOS, NO
    487 014326  013737  002326  002350          MOV     LC,NC2          ;SETUP FOR REVERSE SEEK
    488 014334  163737  001510  002350          SUB     CYINC,NC2       ;ADJUST ADDRESS
    489 014342  013737  002350  002346          MOV     NC2,NC1         ;COPY ADDRESS
    490 014350  163737  001506  002346          SUB     SKSIZ,NC1       ;NEW ENDING ADDRESS
    491
    492 014356  004737  023240          9$:     JSR     PC,ST.CLK       ;RE-INIT THE CLOCK
    493 014362  012777  033526  015000          MOV     #ISR,@RMVEC     ;RESTORE THE RH VECTOR
    494
    495 014370                          10$:
        014370  112737  000105  036510          MOVB    #SEEK,DPB.A+2   ;SEEK=COMMAND
    496 014376  005037  036516                  CLR     DPB.A+10        ;USE TRACK 0 & SECTOR 0
    497 014402  013737  002350  036520          MOV     NC2,DPB.A+12    ;STARTING CYLINDER
    500 014410  012737  015140  001350          MOV     #EXIT15,BYPASS  ;GO TO EXIT15 IF ERROR
    501 014416  004037  024076                  JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
    502
    503 014422  005077  165106          11$:    CLR     @PKCS           ;STOP THE CLOCK
    504 014426  012777  015102  165074          MOV     #18$,@PKV       ;SETUP CLOCK VECTOR IN CASE OF OVERFLOW
    505 014434  012777  025662  014726          MOV     #DORTI,@RMVEC   ;CHANGE RM80 VECTOR
    506
    507 014442  005077  165070          12$:    CLR     @PKB            ;START COUNT AT ZERO
    508 014446  013764  002346  000034          MOV     NC1,RMDC(R4)    ;LOAD CYLINDER ADDRESS
    509 014454  012764  000105  000000          MOV     #SEEK,RMCS1(R4) ;START A SEEK
    510 014462  012777  000131  165044          MOV     #131,@PKCS      ;START THE CLOCK
    511 014470  000001                          WAIT                    ;WAIT ON INTERRUPT
    512 014472  042777  000101  165034          BIC     #101,@PKCS      ;STOP CLOCK
    513 014500  032764  040000  000012          BIT     #BIT14,RMDS(R4) ;ERR=1?
    514 014506  001425                          BEQ     13$             ;NO--BRANCH
    515 014510  104412                          SAVREG                  ;SAVE R0-R5
        014512  012702  036574                  MOV     #DTADPB,R2      ;DPB POINTER
        014516  004737  035744                  JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
        014522  004737  023146                  JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
```

```
                014526  104413                          RESREG                  ;RESTORE R0-R5
    516         014530  104017                          EMT     17              ;REPORT THE ERROR
    517         014532  032737  040000  036702          BIT     #SKI,RM.REG+RMER2  ;SEE IF SKI SET
    518         014540  001410                          BEQ     13$             ;IF EQ, NO
    519         014542  004737  023240                  JSR     PC,ST.CLK       ;RE-INIT THE CLOCK
    520         014546  012777  033526  014614          MOV     #ISR,@RMVEC     ;RESTORE THE RH VECTOR
    521         014554  004037  024710                  JSR     R0,CALL.R       ;ISSUE A RECAL
    522         014560  000703                          BR      10$             ;TRY AGAIN
    523
    524         014562  067737  164752  001472  13$:    ADD     @PKC,SKTIM      ;ADD THE COUNTER VALUE
    527         014570  005537  001474                  ADC     SKTIM+2         ;ADD ANY CARRY
                014574  005537  001476                  ADC     SKTIM+4         ;ADD ANY CARRY
                014600  005537  001500                  ADC     SKTIM+6         ;ADD ANY CARRY
    528         014604  062737  000001  001502          ADD     #1,SKCNT        ;COUNT THE SEEK
    529         014612  005537  001504                  ADC     SKCNT+2         ;  "     "  "
    530
    531         014616  013737  002346  002350          MOV     NC1,NC2         ;COPY ENDING ADDRESS
    532         014624  163737  001506  002346          SUB     SKSIZ,NC1       ;NEW ENDING ADDRESS
    533         014632  005737  002346                  TST     NC1             ;SEE IF THE NEW CYL IS NEGATIVE
    534         014636  100404                          BMI     14$             ;IF MI, YES - FINISHED
    535         014640  023737  002346  002324          CMP     NC1,FC          ;FINISHED ?
    536         014646  103275                          BHIS    12$             ;IF HIS, NO
    537         014650  013746  002326          14$:    MOV     LC,-(SP)        ;PUT THE ENDING CYLINDER ON THE STACK
    538         014654  163716  002324                  SUB     FC,(SP)         ;FIND THE DIFFERENCE
    539         014660  023726  001506                  CMP     SKSIZ,(SP)+     ;SEE IF TEST FINISHED
    540         014664  103004                          BHIS    15$             ;IF HIS, YES
    541         014666  005237  001510                  INC     CYINC           ;INCREMENT THE CYLINDER OFFSET
    542         014672  000137  013714                  JMP     2$              ;CONTINUE
    543
    544         014676  004737  023146          15$:    JSR     PC,CNTCLR       ;GO CLEAR MASSBUS CONTROLLER
    545         014702  004737  023240                  JSR     PC,ST.CLK       ;RE-INIT THE CLOCK
    546         014706  012777  033526  014454          MOV     #ISR,@RMVEC     ;RESTORE THE RH VECTOR
    547
    548         014714  104412                          SAVREG                  ;SAVE R0-R5
    583         014716  104401  037634                  TYPE    ,MEASUR         ;TYPE 'AVERAGE SEEK TIME MEASUREMENT'
                014722  013703  001472                  MOV     SKTIM,R3        ;LOAD THE DIVIDEND
                014726  013702  001474                  MOV     SKTIM+2,R2      ;  "     "  "
                014732  013701  001476                  MOV     SKTIM+4,R1      ;  "     "  "
                014736  013700  001500                  MOV     SKTIM+6,R0      ;  "     "  "
                014742  013705  001502                  MOV     SKCNT,R5        ;LOAD THE DIVISOR
                014746  013704  001504                  MOV     SKCNT+2,R4      ;  "     "  "
                014752  004737  022652                  JSR     PC,M.DPID       ;FIND THE AVERAGE
                014756  010337  001472                  MOV     R3,SKTIM        ;LOAD THE QUOTENT FOR DISPLAY
                014762  010237  001474                  MOV     R2,SKTIM+2      ;  "     "  "     "  "
                014766  104401  001231                  TYPE    ,$CRLF          ;CR-LF
                014772  104401  037710                  TYPE    ,MSGAVG         ;"AVG="
                014776  012746  001472                  MOV     #SKTIM,-(SP)    ;AVERAGE SEEK TIME POINTER
                015002  004737  022246                  JSR     PC,$DB2D        ;CONVERT TO ASCII
                015006  004737  022442                  JSR     PC,$SUPRS       ;TYPE IT
                015012  104401  037716                  TYPE    ,MSGUS          ;TYPE "0 US"
                015016  104401  037160                  TYPE    ,COMMA          ;TYPE ", "
                015022  012746  001502                  MOV     #SKCNT,-(SP)    ;SEEK COUNT POINTER
                015026  004737  022246                  JSR     PC,$DB2D        ;CONVERT TO ASCII
                015032  004737  022442                  JSR     PC,$SUPRS       ;TYPE IT
                015036  104401  040021                  TYPE    ,MSGSEK         ;TYPE " SEEKS TIMED"
                015042  104413                          RESREG                  ;RESTORE R0-R5
                015044  000435                          BR      EXIT15          ;BR TO EXIT15
```

```
    015046     200     123     105  16$:    .ASCIZ  <CRLF>/SEEK SIZE=/
    015062     054     040     116  17$:    .ASCIZ  /, NO. OF SEEKS=/
                                    .EVEN

    015102  005077  164426          18$:    CLR     @PKCS           ;STOP THE CLOCK
    015106  005037  177776                  CLR     PS              ;DROP THE PRIORITY
    015112  012600                          MOV     (SP)+,R0        ;PC OF WAIT+2
    015114  005726                          TST     (SP)+           ;POP THE PS FROM THE STACK
    015116  104412                          SAVREG                  ;SAVE R0-R5
    015120  012702  036574                  MOV     #DTADPB,R2      ;DPB POINTER
    015124  004737  035744                  JSR     PC,SVRH70       ;SAVE ALL THE RH/RM REGISTERS
    015130  004737  023146                  JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
    015134  104413                          RESREG                  ;RESTORE R0-R5
    015136  104020                          EMT     20              ;CLOCK OVERFLOWED
    015140  000004                  EXIT15: SCOPE                   ;CALL SCOPE ROUTINE
```

END OF PASS ROUTINE

```
     1                          .SBTTL  END OF PASS ROUTINE

                                ;;*****************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*INDICATE END-OF-PROGRAM AFTER 8. PASSES THRU THE PROGRAM
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO RTURN

     015142                     $EOP:
     015142  104401  015150             TYPE    ,65$            ;;TYPE ASCIZ STRING
     015146  000407                     BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <CRLF><LF>/END OF PASS/
     015166                     64$:
     015166  104401  015174             TYPE    ,67$            ;;TYPE ASCIZ STRING
     015172  000405                     BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  / ON DRIVE/
     015206                     66$:
     015206  013746  001352             MOV     CHKDRV,-(SP)    ;;SAVE CHKDRV FOR TYPEOUT
     015212  104403                     TYPOS                   ;;GO TYPE--OCTAL ASCII
     015214     002                     .BYTE   2               ;;TYPE 2 DIGIT(S)
     015215     000                     .BYTE   0               ;;SUPPRESS LEADING ZEROS
     015216  005737  001126             TST     $ERTTL          ;ANY ERRORS DETECTED ?
     015222  001420                     BEQ     1$              ;BR IF NO
     015224  104401  015232             TYPE    ,69$            ;;TYPE ASCIZ STRING
     015230  000412                     BR      68$             ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  /   ERRORS DETECTED=/
     015256                     68$:
     015256  013746  001126             MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
     015262  104402                     TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
     015264  005037  001126     1$:     CLR     $ERTTL          ;CLEAR ERROR TOTAL
     015270  005037  001116             CLR     $TSTNM          ;;ZERO THE TEST NUMBER
     015274  005037  001220             CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
     015300  005237  001242             INC     $PASS           ;;INCREMENT THE PASS NUMBER
     015304  042737  100000  001242     BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
     015312  005327                     DEC     (PC)+           ;;LOOP?
     015314  000010             $EOPCT: .WORD   8.
     015316  003027                     BGT     $DOAGN          ;;YES
     015320  012737                     MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
     015322  000010             $ENDCT: .WORD   8.
     015324  015314                     $EOPCT
     015326  104401  015334             TYPE    ,65$            ;;TYPE ASCIZ STRING
     015332  000407                     BR      64$             ;;GET OVER THE ASCIZ
                                ;;6?$:  .ASCIZ  <CRLF>/END OF TEST/<CRLF>
     015352                     64$:
     015352  104401  015402             TYPE    ,$ENULL         ;TYPE NULL CHARACTER
     015356  013700  000042     $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
     015362  001405                     BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
     015364  000005                     RESET                   ;;CLEAR THE WORLD
     015366  004710             $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
     015370  000240                     NOP                     ;;SAVE ROOM
     015372  000240                     NOP                     ;;FOR
     015374  000240                     NOP                     ;;ACT11
     015376                     $DOAGN:
     015376  000137                     JMP     @(PC)+          ;;RETURN
     015400  015406             $RTNAD: .WORD   RTURN
     015402     377     377     000     $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
                                        .EVEN
```

END OF PASS ROUTINE

```
      2
      3 015406  012706  001100          RTURN:  MOV     #STACK,SP       ;RESTORE STACK
      4 015412  004737  020134                  JSR     PC,$TKINT       ;MAKE SURE KEYBOARD INTERRUPT AND
      5 015416  004737  023240                  JSR     PC,ST.CLK       ;INITIALIZE THE CLOCK
      6 015422  000137  005102                  JMP     RSTART          ;RETURN TO RESTART
```

```
     1                                  .SBTTL  APT COMMUNICATIONS ROUTINE

                                        ;;*****************************************************************
015426  112737  000001  015672  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
015434  112737  000001  015670  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
015442  000403                          BR      $ATYC
015444  112737  000001  015672  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
015452                          $ATYC:
015452  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
015454  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
015456  105737  015670                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
015462  001450                          BEQ     5$              ;;IF NOT:  BR
015464  122737  000001  001254          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
015472  001031                          BNE     3$              ;;IF NOT:  BR
015474  132737  000100  001255          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
015502  001425                          BEQ     3$              ;;IF NOT:  BR
015504  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
015510  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
015516  005737  001234          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
015522  001375                          BNE     1$              ;;IF NOT:  WAIT
015524  010037  001250                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
015530  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
015532  001376                          BNE     2$
015534  163700  001250                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
015540  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
015542  010037  001252                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
015546  012737  000004  001234          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
015554  000413                          BR      5$
015556  017637  000004  015602  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
015564  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
015572  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
015576  004737  017116                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
015602  000000                  4$:     .WORD   0
015604                          5$:
015604  105737  015672          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
015610  001416                          BEQ     12$             ;;IF NOT:  BR
015612  005737  001254                  TST     $ENV            ;;RUNNING UNDER APT?
015616  001413                          BEQ     12$             ;;IF NOT:  BR
015620  005737  001234          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
015624  001375                          BNE     11$             ;;IF NOT:  WAIT
015626  017637  000004  001236          MOV     @4(SP),$FATAL   ;;GET ERROR #
015634  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
015642  005237  001234                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
015646  105037  015672          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
015652  105037  015671                  CLRB    $LFLG           ;;CLEAR LOG FLAG
015656  105037  015670                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
015662  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
015664  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
015666  000207                          RTS     PC              ;;RETURN
015670     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
015671     000          $LFLG:  .BYTE   0               ;;LOG FLAG
015672     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
                                .EVEN
            000200      APTSIZE = 200
            000001      APTENV  = 001
            000100      APTSPOOL= 100
            000040      APTCSUP = 040
```

```
                                .SBTTL  ERROR HANDLER ROUTINE

                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                ;*AND GO TO TYPERR ON ERROR
                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;*SW15=1         HALT ON ERROR
                                ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                ;*SW10=1         BELL ON ERROR
                                ;*SW09=1         LOOP ON ERROR
                                ;*CALL
                                ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

015674  105037  016334         $ERROR: CLRB    IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
015700  104407                         CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
015702  032777  000400  163244         BIT     #SW08,@SWR      ;SEND ERROR MESSAGE TO TTY?
015710  001411                         BEQ     7$              ;YES--BRANCH
015712  005737  001326                 TST     LPTAVL          ;IS THERE A LINE PRINTER AVAILABLE?
015716  001406                         BEQ     7$              ;NO--BRANCH
015720  013737  001554  001164         MOV     LPS,$TPS        ;YES--SETUP STATUS
015726  013737  001556  001166         MOV     LPB,$TPB        ;AND BUFFER REG.'S FOR LINE PRINTER
015734  105237  001117         7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
015740  001775                         BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
015742  013777  001116  163206         MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
015750  032777  002000  163176         BIT     #BIT10,@SWR     ;;BELL ON ERROR?
015756  001402                         BEQ     1$              ;;NO - SKIP
015760  104401  001224                 TYPE    ,$BELL          ;;RING BELL
015764  005237  001126         1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
015770  011637  001132                 MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
015774  162737  000002  001132         SUB     #2,$ERRPC
016002  117737  163124  001130         MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
016010  032777  001000  163136         BIT     #BIT09,@SWR     ;;SEE IF LOOP ON ERROR IS SET
016016  001060                         BNE     1004$           ;;BRANCH AROUND ROUTINE IF SO
016020  122737  000177  001130         CMPB    #177,$ITEMB     ;;SEE IF THIS IS THE POWER FAIL CALL
016026  001454                         BEQ     1004$           ;;BRANCH AROUND ROUTINE IF IT IS
016030  105737  016334                 TSTB    IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
016034  001047                         BNE     1003$           ;;BRANCH IF SO
016036  022737  177777  016332         CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
016044  001445                         BEQ     1004$           ;;BRANCH IF SO
016046  013746  000004                 MOV     ERRVEC,-(SP)    ;;SAVE CONTENTS OF ERROR VECTOR
016052  012737  016070  000004         MOV     #1000$,ERRVEC   ;;SETUP 'TRAP' RETURN ADDRESS
016060  013737  177766  016332         MOV     177766,CPSAVE   ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
016066  000406                         BR      1001$
016070  012737  177777  016332 1000$:  MOV     #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
016076  012716  016104                 MOV     #1001$,(SP)     ;;SETUP RETURN ADDRESS
016102  000002                         RTI
016104  012637  000004         1001$:  MOV     (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR

016110  022737  177777  016332 1002$:  CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
016116  001420                         BEQ     1004$           ;;BRANCH IF SO
016120  032737  000001  016332         BIT     #BIT00,CPSAVE   ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
016126  001414                         BEQ     1004$           ;;BRANCH IF OK
016130  042737  000001  177766         BIC     #BIT00,177766   ;;CLEAR THE BIT FOUND SET
016136  113737  001130  016334         MOVB    $ITEMB,IBSAVE   ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
016144  112737  000177  001130         MOVB    #177,$ITEMB     ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
016152  000402                         BR      1004$           ;;BRANCH OVER IBSAVE CLEARING
```

```
016154  105037  016334          1003$:  CLRB    IBSAVE          ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
016160                          1004$:
016160  032777  020000  162766          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
016166  001004                          BNE     20$             ;;SKIP TYPEOUTS
016170  004737  016336                  JSR     PC,TYPERR       ;;GO TO USER ERROR ROUTINE
016174  104401  001231                  TYPE    ,$CRLF
016200                          20$:
016200  122737  000001  001254          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
016206  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
016210  113737  001130  016222          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
016216  004737  015444                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
016222     000          21$:    .BYTE   0
016223     000                  .BYTE   0
016224  000777          22$:    BR      22$             ;;APT ERROR LOOP
016226  105737  016334          2$:     TSTB    IBSAVE          ;;SEE IF IBSAVE IS LOADED
016232  001005                          BNE     3$              ;;BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
016234  005777  162714                  TST     @SWR            ;;HALT ON ERROR
016240  100002                          BPL     3$              ;;SKIP IF CONTINUE
016242  000000                          HALT                    ;;HALT ON ERROR!
016244  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
016246                          3$:
016246  105737  016334                  TSTB    IBSAVE          ;;SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
016252  001230                          BNE     7$              ;;BRANCH BACK TO CALL ORIGINAL ERROR
016254  032777  001000  162672          BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
016262  001402                          BEQ     4$              ;;BR IF NO
016264  013716  001124                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
016270  005737  001222          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
016274  001402                          BEQ     5$              ;;BR IF NONE
016276  013716  001222                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
016302                          5$:
016302  022737  015366  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
016310  001001                          BNE     6$              ;;BRANCH IF NO
016312  000000                          HALT                    ;;YES
016314                          6$:
016314  013737  001550  001164          MOV     TPS,$TPS        ;SET STATUS AND BUFFER REG.'S
016322  013737  001552  001166          MOV     TPB,$TPB        ;FOR TTY
016330  000002                          RTI                     ;RETURN FROM ERROR CALL
016332  000000          CPSAVE: .WORD   0               ;;LOCATION TO SAVE CPU ERROR REG CONTENTS
016334  000000          IBSAVE: .WORD   0               ;;LOCATION TO SAVE ITEM BYTE
```

```
                                        .SBTTL  TYPERR - TYPE ERROR ROUTINE
                                
                                        ;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
                                        ;WHICH ERROR IS TO BE REPORTED, IT THEN OBTAINS, FROM THE "ERROR
                                        ;TABLE" ($ERRTB), AND REPORTS THE APPROPIATE INFORMATION
                                        ;CONCERNING THE ERROR.
                                        ;CALL
                                        ;       JSR     PC,TYPERR
                                        ;       RETURN
                                
016336  113737  001116  001212  TYPERR: MOVB    $TSTNM,$TMPO    ;SAVE THE TEST NUMBER
016344  104412                          SAVREG                  ;SAVE R0 - R5
016346  162700  000004                  SUB     #4,R0           ;FORM TEST PC
016352  010037  001176                  MOV     R0,$REG0        ;COPY R0-R5 IN $REG0-$REG5
016356  010137  001200                  MOV     R1,$REG1
016362  010237  001202                  MOV     R2,$REG2
016366  010337  001204                  MOV     R3,$REG3
016372  010437  001206                  MOV     R4,$REG4
016376  010537  001210                  MOV     R5,$REG5
016402  113700  001130                  MOVB    $ITEMB,R0       ;PICKUP ERROR ITEM NUMBER
016406  122700  000177                  CMPB    #177,R0         ;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
016412  001007                          BNE     1$              ;BRANCH IF NOT
016414  005001                          CLR     R1
016416  013737  001240  017114          MOV     $TESTN,PFTSTN   ;GET TEST NUMBER
016424  012700  016754                  MOV     #PFECH,R0       ;MOVE POWER FAIL ERROR CALL TABLE TO R0
016430  000412                          BR      3$
016432  010001                  1$:     MOV     R0,R1           ;AND COPY IT INTO R1
016434  005300                          DEC     R0              ;FORM INDEX FOR ERROR TABLE
016436  106300                          ASLB    R0
016440  106300                          ASLB    R0
016442  106300                          ASLB    R0
016444  103002                          BCC     2$              ;IS ERROR > 37?
016446  062700  000240                  ADD     #ITEM41-$ERRTB,R0       ;YES--FORM OFFSET
016452  062700  002000          2$:     ADD     #$ERRTB,R0      ;FORM ADDRESS
016456  012037  016476          3$:     MOV     (R0)+,4$        ;GET ERROR MESSAGE (EM) POINTER
016462  001451                          BEQ     9$              ;BRANCH IF THERE ISN'T ONE
016464  004737  023030                  JSR     PC,INCEC        ;INCREMENT ERROR COUNT
016470  104401  001231                  TYPE    ,$CRLF          ;"CARRIAGE RETURN - LINE FEED
016474  104401                          TYPE
016476  000000                  4$:     .WORD   0               ;"EM" POINTER GOES HERE
016500  162701  000041                  SUB     #41,R1          ;SPECIAL ERROR ITEM NUMBER?
016504  100440                          BMI     9$              ;NO--BRANCH
016506  013701  001356                  MOV     SVSTAT,R1       ;GET STATUS/ERROR INDICATOR
016512  106301                          ASLB    R1              ;STRIP "DONE" BIT (BIT07)
016514  006301                          ASL     R1              ;STRIP "ERROR" BIT (BIT15)
016516  012702  001746                  MOV     #STATBL,R2      ;1ST ADDRESS ON STATUS MESSAGE POINTERS
016522  005003                          CLR     R3              ;CARRIAGE RETURN-LINE FEED SWITCH
016524  104401  016532                  TYPE    ,65$            ;;TYPE ASCIZ STRING
016530  000402                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  / (/
016536                          64$:
016536  012237  016560          5$:     MOV     (R2)+,7$        ;MESSAGE POINTER
016542  006301                          ASL     R1              ;TYPE THIS MESSAGE?
016544  103013                          BCC     8$              ;NO--BRANCH
016546  005103                          COM     R3              ;YES--TYPE A "CR" & "LF"?
016550  001002                          BNE     6$              ;NO--BRANCH
016552  104401  001231                  TYPE    ,$CRLF          ;YES
```

```
     59 016556  104401                  6$:     TYPE
     60 016560  000000                  7$:     .WORD   0               ;MESSAGE POINTER GOES HERE
     61 016562  005701                          TST     R1              ;MORE TO TYPE?
     62 016564  001403                          BEQ     8$              ;NO--BRANCH
     63 016566  104401  040053                  TYPE    ,BLNKS2         ;TYPE 2 BLANKS
     64 016572  000761                          BR      5$              ;LOOP
     65 016574  001360                  8$:     BNE     5$              ;BRANCH IF NOT FINISHED
     66 016576  104401  016604                  TYPE    ,67$            ;;TYPE ASCIZ STRING
        016602  000401                          BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  /!/
        016606                          66$:
     67 016606  012037  016622          9$:     MOV     (R0)+,10$       ;PICK UP DATA HEADER (DH) POINTER
     68 016612  001404                          BEQ     11$             ;BRANCH IF NONE
     69 016614  104401  001231                  TYPE    ,$CRLF          ;CARRIAGE RETURN-LINE FEED
     70 016620  104401                          TYPE
     71 016622  000000                  10$:    .WORD   0               ;'DH' POINTER GOES HERE
     72 016624  012001                  11$:    MOV     (R0)+,R1        ;PICKUP DATA TABLE (DT) POINTER
     73 016626  001450                          BEQ     20$             ;BRANCH IF NONE
     74 016630  005005                          CLR     R5              ;SET INDENT SWITCH
     75 016632  012000                          MOV     (R0)+,R0        ;DATA FORMAT (DF) POINTER
     76 016634  012002                          MOV     (R0)+,R2        ;NUMBER OF DH'S TO TYPE
     77 016636  001441                          BEQ     19$             ;BRANCH IF DH NUMBER IS 0
     78 016640  005105                          COM     R5              ;NO INDENT
     79 016642  104401  001231                  TYPE    ,$CRLF          ;CARRIAGE RETURN-LINE FEED
     80 016646  112003                  12$:    MOVB    (R0)+,R3        ;NUMBER OF DATA WORDS TO TYPE
     81 016650  112004                          MOVB    (R0)+,R4        ;AND HOW TO TYPE THEM
     82 016652  006004                  13$:    ROR     R4              ;OCTAL OR DECIMAL?
     83 016654  103403                          BCS     14$             ;DECIMAL--BRANCH
     84 016656  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
        016660  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
     85 016662  000402                          BR      15$
     86 016664                          14$:
        016664  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
        016666  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
     87 016670  005303                  15$:    DEC     R3              ;MORE NUMBERS TO TYPE?
     88 016672  001403                          BEQ     16$             ;NO--BRANCH
     89 016674  104401  040053                  TYPE    ,BLNKS2         ;TYPE 2 BLANKS
     90 016700  000764                          BR      13$             ;LOOP
     91 016702  005302                  16$:    DEC     R2              ;MORE DH'S?
     92 016704  003421                          BLE     20$             ;NO--BRANCH
     93 016706  104401  001231                  TYPE    ,$CRLF          ;YES--START A NEW LINE
     94 016712  005105                          COM     R5              ;INDENT?
     95 016714  001002                          BNE     17$             ;NO--BRANCH
     96 016716  104401  040053                  TYPE    ,BLNKS2         ;TYPE 2 BLANKS
     97 016722  012037  016730          17$:    MOV     (R0)+,18$       ;GET NEXT DH
     98 016726  104401                          TYPE                    ;AND TYPE IT
     99 016730  000000                  18$:    .WORD   0               ;DH POINTER GOES HERE
    100 016732  104401  001231                  TYPE    ,$CRLF          ;CARRIAGE RETURN-LINE FEED
    101 016736  005705                          TST     R5              ;INDENT?
    102 016740  001342                          BNE     12$             ;NO--BRANCH
    103 016742  104401  040053          19$:    TYPE    ,BLNKS2         ;TYPE 2 BLANKS
    104 016746  000737                          BR      12$             ;LOOP
    105 016750  104413                  20$:    RESREG                  ;RESTORE R0 - R5
    106 016752  000207                          RTS     PC              ;RETURN
    107 016754  016764  017046  017100  PFECH:  PFECH1,PFECH2,PFECH3,PFECH4 ;WORDS DEFINING TABLES BELOW
    108 016764     120     117     127  PFECH1: .ASCIZ  ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
    109 017046     124     105     123  PFECH2: .ASCIZ  ?TESTNO  ERR PC  CPUERREG?
```

```
                                                .EVEN
017100  017114  001132  016332  PFECH3: .WORD   PFTSTN,$ERRPC,CPSAVE,0
017110  000001                  PFECH4: .WORD   1               ;NUMBER OF DATA HEADERS
017112     003                          .BYTE   3               ;NUMBER OF WORDS IN DATA TABLE
017113     000                          .BYTE   0               ;ALL 3 NUMBERS ARE OCTAL
017114  000000                  PFTSTN: .WORD   0               ;CONTAINS TEST NUMBER FOR PF BIT ERROR
```

```
     1                                  .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
                                        ;*

017116  105737  001173          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
017122  100002                          BPL     1$              ;;BR IF YES
017124  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
017126  000430                          BR      3$              ;;LEAVE
017130  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
017132  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
017136  122737  000001  001254          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
017144  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
017146  132737  000100  001255          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
017154  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
017156  010037  017166                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
017162  004737  015434                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
017166  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
017170  132737  000040  001255  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
017176  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
017200  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
017202  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
017204  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
017206  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
017210  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
017214  000002                          RTI                     ;;RETURN
017216  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
017222  001430                          BEQ     8$
017224  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
017230  001006                          BNE     5$
017232  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
017234  104401                          TYPE                    ;;TYPE A CR AND LF
017236  001231                          $CRLF
017240  105037  017446                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
017244  000755                          BR      2$              ;;GET NEXT CHARACTER
017246  004737  017330          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
017252  123726  001172          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
017256  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
017260  013746  001170                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
017264  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
017270  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
017272  004737  017330                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
017276  105337  017446                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
017302  000770                          BR      7$              ;;LOOP
```

```
                                        ;HORIZONTAL TAB PROCESSOR

017304  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
017310  004737  017330          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
017314  132737  000007  017446          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
017322  001372                          BNE     9$              ;;TAB STOP
017324  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
017326  000724                          BR      2$              ;;GET NEXT CHARACTER
017330                          $TYPEC:
017330  105777  161624                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?
017334  100022                          BPL     10$             ;;BR IF NOT
017336  017746  161620                  MOV     @$TKB,-(SP)     ;;GET CHAR
017342  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS
017346  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF
017352  001012                          BNE     102$            ;;BR IF NOT
017354                          101$:
017354  105777  161600                  TSTB    @$TKS           ;;WAIT FOR CHAR
017360  100375                          BPL     101$
017362  117716  161574                  MOVB    @$TKB,(SP)      ;;GET CHAR
017366  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT
017372  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?
017376  001366                          BNE     101$            ;;BR IF NOT
017400                          102$:
017400  005726                          TST     (SP)+           ;;FIX STACK
017402                          10$:
017402  105777  161556                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
017406  100375                          BPL     10$
017410  116677  000002  161550          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
017416  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
017424  001003                          BNE     1$              ;;BRANCH IF NO
017426  105037  017446                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
017432  000406                          BR      $TYPEX          ;;EXIT
017434  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
017442  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
017444  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
017446  000000          $CHARCNT:.WORD  0                       ;;CHARACTER COUNT STORAGE
017450  000207          $TYPEX: RTS     PC
```

```
                                .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 TYP OUT OF A 16 BIT NUMBER
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOC                   ;;CALL FOR TYPEOUT

017452  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
017456  116637  000001  017675          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
017464  112637  017677                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
017470  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
017474  000406                          BR      $TYPON
017476  112737  000001  017675  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
017504  112737  000006  017677          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
017512  112737  000005  017674  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
017520  010346                          MOV     R3,-(SP)        ;;SAVE R3
017522  010446                          MOV     R4,-(SP)        ;;SAVE R4
017524  010546                          MOV     R5,-(SP)        ;;SAVE R5
017526  113704  017677                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
017532  005404                          NEG     R4
017534  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
017540  110437  017676                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
017544  113704  017675                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
017550  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
017554  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
017556  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
017560  000404                          BR      3$              ;;GO DO MSB
017562  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
017564  006105                          ROL     R5
017566  006105                          ROL     R5
017570  010503                          MOV     R5,R3
017572  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
017574  105337  017676                  DECB    $OMODE          ;;TYPE THIS DIGIT?
017600  100016                          BPL     7$              ;;BR IF NO
017602  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
017606  001002                          BNE     4$              ;;TEST FOR 0
017610  005704                          TST     R4              ;;SUPPRESS THIS 0?
017612  001403                          BEQ     5$              ;;BR IF YES
017614  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
```

BINARY TO OCTAL (ASCII) AND TYPE

```
017616  052703  000060              BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
017622  052703  000040      5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
017626  110337  017672              MOVB    R3,8$           ;;SAVE FOR TYPING
017632  104401  017672              TYPE    ,8$             ;;GO TYPE THIS DIGIT
017636  105337  017674      7$:     DECB    $OCNT           ;;COUNT BY 1
017642  003347                      BGT     2$              ;;BR IF MORE TO DO
017644  002402                      BLT     6$              ;;BR IF DONE
017646  005204                      INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
017650  000744                      BR      2$              ;;GO DO THE LAST DIGIT
017652  012605      6$:             MOV     (SP)+,R5        ;;RESTORE R5
017654  012604                      MOV     (SP)+,R4        ;;RESTORE R4
017656  012603                      MOV     (SP)+,R3        ;;RESTORE R3
017660  016666  000002  000004      MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
017666  012616                      MOV     (SP)+,(SP)
017670  000002                      RTI                     ;;RETURN
017672     000      8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
017673     000              .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
017674     000      $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
017675     000      $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
017676  000000      $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
                                .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

017700                          $TYPDS:
017700  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
017702  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
017704  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
017706  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
017710  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
017712  012746  020200                 MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
017716  016605  000020                 MOV     20(SP),R5       ;;GET THE INPUT NUMBER
017722  100004                         BPL     1$              ;;BR IF INPUT IS POS.
017724  005405                         NEG     R5              ;;MAKE THE BINARY NUMBER POS.
017726  112766  000055  000001         MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
017734  005000                 1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
017736  012703  020114                 MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
017742  112723  000040                 MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
017746  005002                 2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
017750  016001  020104                 MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
017754  160105                 3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
017756  002402                         BLT     4$              ;;BR IF DONE
017760  005202                         INC     R2              ;;INCREASE THE BCD DIGIT BY 1
017762  000774                         BR      3$
017764  060105                 4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
017766  005702                         TST     R2              ;;CHECK IF BCD DIGIT=0
017770  001002                         BNE     5$              ;;FALL THROUGH IF 0
017772  105716                         TSTB    (SP)            ;;STILL DOING LEADING 0'S?
017774  100407                         BMI     7$              ;;BR IF YES
017776  106316                 5$:     ASLB    (SP)            ;;MSD?
020000  103003                         BCC     6$              ;;BR IF NO
020002  116663  000001  177777         MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
020010  052702  000060         6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
020014  052702  000040         7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
020020  110223                         MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
020022  005720                         TST     (R0)+           ;;JUST INCREMENTING
020024  020027  000010                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
020030  002746                         BLT     2$              ;;GO DO THE NEXT DIGIT
020032  003002                         BGT     8$              ;;GO TO EXIT
020034  010502                         MOV     R5,R2           ;;GET THE LSD
020036  000764                         BR      6$              ;;GO CHANGE TO ASCII
020040  105726                 8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
020042  100003                         BPL     9$              ;;BR IF NO
020044  116663  177777  177776         MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
020052  105013                 9$:     CLRB    (R3)            ;;SET THE TERMINATOR
020054  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
020056  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
020060  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
020062  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
```

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```
020064  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
020066  104401  020114                   TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
020072  016666  000002  000004           MOV     2(SP),4(SP)     ;;ADJUST THE STACK
020100  012616                          MOV     (SP)+,(SP)
020102  000002                          RTI                     ;;RETURN TO USER
020104  023420                  $DTBL:  10000.
020106  001750                          1000.
020110  000144                          100.
020112  000012                          10.
020114                          $DBLK:  .BLKW   4
```

```
                                .SBTTL  TTY INPUT ROUTINE

                                ;;*******************************************************************
                                .ENABL  LSB
020124  000000                  $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
020126  000000                  $TKQIN: .WORD   0               ;;INPUT POINTER
020130  000000                  $TKQOUT: .WORD  0               ;;OUTPUT POINTER
020132                          $TKQSRT: .BLKB  1               ;;TTY KEYBOARD QUEUE
        020133                  $TKQEND=.
                                .EVEN

                                ;*TK INITIALIZE ROUTINE
                                ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
                                ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
                                ;
                                ;*CALL:
                                ;*      JSR     PC,$TKINT
                                ;*      RETURN
                                ;
020134  005037  020124          $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
020140  012737  020132  020126          MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
020146  013737  020126  020130          MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
020154  012737  020204  000060          MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
020162  012737  000200  000062          MOV     #200,@#TKVEC+2  ;;'BR' LEVEL 4
020170  005777  160766                  TST     @$TKB           ;;CLEAR DONE FLAG
020174  012777  000100  160756          MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
020202  000207                          RTS     PC              ;;RETURN TO CALLER

                                ;*TK SERVICE ROUTINE
                                ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
                                ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
                                ;*IT IN THE QUEUE.
                                ;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
                                ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (SHUT)
                                ;
020204  117746  160752          $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
020210  042716  177600                  BIC     #^C177,(SP)     ;;STRIP THE JUNK
020214  021627  000021                  CMP     (SP),#$XON      ;;IS IT A RANDOM XON?
020220  001002                          BNE     30$             ;;BRANCH IF NO
020222  005726                          TST     (SP)+           ;;CLEAN RANDOM XON OFF STACK
020224  000002                          RTI                     ;;RETURN
020226                          30$:
020226  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL C?
020232  001007                          BNE     1$              ;;BRANCH IF NO
020234  104401  021346                  TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
020240  004737  020134                  JSR     PC,$TKINT       ;;INIT THE KEYBOARD
020244  005726                          TST     (SP)+           ;;CLEAN UP STACK
020246  000137  021410                  JMP     SHUT            ;;CONTROL C RESTART
020252  021627  000007          1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
020256  001004                          BNE     2$              ;;BRANCH IF NO
020260  022737  000176  001154          CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
020266  001500                          BEQ     6$              ;;GO TO SWR CHANGE

020270                          2$:
020270  022737  000001  020124          CMP     #1,$TKCNT       ;;IS THE QUEUE FULL?
020276  001004                          BNE     3$              ;;BRANCH IF NO
020300  104401  001224                  TYPE    ,$BELL          ;;RING THE TTY BELL
```

```
020304  005726                          TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
020306  000451                          BR      5$              ;;EXIT
020310  021627  000023        3$:       CMP     (SP),#23        ;;IS IT A CONTROL-S?
020314  001021                          BNE     32$             ;;BRANCH IF NO
020316  005077  160636                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
020322  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
020324  105777  160630        31$:      TSTB    @$TKS           ;;WAIT FOR A CHAR
020330  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
020332  117746  160624                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
020336  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
020342  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
020346  001366                          BNE     31$             ;;BRANCH IF NO
020350  012777  000100  160602          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
020356  000002                          RTI                     ;;RETURN
020360  005237  020124        32$:      INC     $TKCNT          ;;COUNT THIS CHARACTER
020364  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
020370  002405                          BLT     4$              ;;BRANCH IF YES
020372  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
020376  003002                          BGT     4$              ;;BRANCH IF YES
020400  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
020404  112677  177516        4$:       MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
020410  005237  020126                  INC     $TKQIN          ;;UPDATE THE POINTER
020414  023727  020126  020133          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
020422  001003                          BNE     5$              ;;BRANCH IF NO
020424  012737  020132  020126          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
020432  000002                5$:       RTI                     ;;RETURN

                        ;;**********************************************************************
                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                        ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
                        ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
020434  022737  000176  001154  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
020442  001124                          BNE     15$             ;;EXIT IF NOT
020444  105777  160510                  TSTB    @$TKS           ;;IS A CHAR WAITING?
020450  100121                          BPL     15$             ;;IF NOT, EXIT
020452  117746  160504                  MOVB    @$TKB,-(SP)     ;;YES
020456  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
020462  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
020466  001300                          BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
                                                                ;;AND EXIT

                        ;;**********************************************************************
                        ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
                        ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
                        ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
020470  123727  001150  000001  6$:       CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
020476  001674                          BEQ     2$              ;;BRANCH IF YES
020500  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
020502  004737  020134                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
020506  005077  160446                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
020512  112737  000001  001151          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR

020520  104401  021360                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
020524  104401  021365        $GTSWR:   TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
020530  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
020534  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
020536  104401  021376                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
020542  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
020544  005046                          CLR     -(SP)           ;;THE NEW SWR
020546  105777  160406          7$:     TSTB    @$TKS           ;;CHAR THERE?
020552  100375                          BPL     7$              ;;IF NOT TRY AGAIN

020554  117746  160402                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
020560  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII

020564  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL-C?
020570  001015                          BNE     9$              ;;BRANCH IF NOT
020572  104401  021346                  TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
020576  062706  000006                  ADD     #6,SP           ;;CLEAN UP STACK
020602  123727  001151  000001          CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
020610  001003                          BNE     8$              ;;BRANCH IF NO
020612  012777  000100  160340          MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
020620  000137  021410          8$:     JMP     SHUT            ;;CONTROL-C RESTART


020624  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
020630  001005                          BNE     10$             ;;BRANCH IF NOT
020632  104401  021353                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
020636  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
020642  000737                          BR      19$             ;;LET'S TRY IT AGAIN


020644  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
020650  001022                          BNE     16$             ;;BRANCH IF NO
020652  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
020656  001403                          BEQ     11$             ;;BRANCH IF YES
020660  016677  000002  160266          MOV     2(SP),@SWR      ;;SAVE NEW SWR
020666  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
020672  104401  001231          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
020676  123727  001151  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
020704  001003                          BNE     15$             ;;BRANCH IF NOT
020706  012777  000100  160244          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
020714  000002                  15$:    RTI                     ;;RETURN
020716  004737  017330          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
020722  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
020726  002420                          BLT     18$             ;;BRANCH IF YES
020730  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
020734  003015                          BGT     18$             ;;BRANCH IF YES
020736  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
020742  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
020746  001403                          BEQ     17$             ;;BRANCH IF YES
020750  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
020752  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
020754  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
020756  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
020762  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
020766  000667                          BR      7$              ;;GET THE NEXT ONE
020770  104401  001230          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
020774  000720                          BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB

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

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

020776  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
021000  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE PS
021006  005066  000004                  CLR     4(SP)           ;;GET READY FOR A CHARACTER
021012  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
021014  012746  021022                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
021020  000002                          RTI                     ;;POP NEW PC AND PS
021022                          64$:
021022  005737  020124          1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
021026  001775                          BEQ     1$
021030  005337  020124                  DEC     $TKCNT          ;;DECREMENT THE COUNTER
021034  117766  177070  000004        MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
021042  005237  020130                  INC     $TKQOUT         ;;UPDATE THE POINTER
021046  023727  020130  020133        CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
021054  001003                          BNE     2$              ;;BRANCH IF NO
021056  012737  020132  020130        MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
021064  000002                  2$:     RTI                     ;;RETURN
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

021066  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
021070  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
021072  012703  021322          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
021076  022703  021346          2$:     CMP     #$TTYIN+20.,R3  ;;BUFFER FULL?
021102  101456                          BLOS    4$              ;;BR IF YES
021104  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
021106  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
021110  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
021114  001022                          BNE     5$              ;;BR IF NO
021116  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
021120  001007                          BNE     6$              ;;BR IF NO
021122  112737  000134  021320        MOVB    #'\,9$          ;;TYPE A BACK SLASH
021130  104401  021320                  TYPE    ,9$
021134  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
021140  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
021142  020327  021322                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
021146  103434                          BLO     4$              ;;BR IF YES
021150  111337  021320                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
021154  104401  021320                  TYPE    ,9$             ;;GO TYPE
021160  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
021162  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
021164  001406                          BEQ     7$              ;;BR IF NO
021166  112737  000134  021320        MOVB    #'\,9$          ;;TYPE A BACK SLASH
021174  104401  021320                  TYPE    ,9$
021200  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
021202  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
021206  001003                          BNE     8$              ;;BR IF NO
```

```
021210  104401  021353                  TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
021214  000726                          BR      1$              ;;GO START OVER
021216  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
021222  001011                          BNE     3$              ;;BRANCH IF NO
021224  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
021226  104401  001231                  TYPE    ,$CRLF          ;;TYPE A 'CR' & 'LF'
021232  104401  021322                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
021236  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
021240  104401  001230          4$:     TYPE    ,$QUES          ;;TYPE A '?'
021244  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
021246  111337  021320          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
021252  104401  021320                  TYPE    ,9$
021256  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
021262  001305                          BNE     2$              ;;LOOP IF NOT RETURN
021264  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
021270  104401  001232                  TYPE    ,$LF            ;;TYPE A LINE FEED
021274  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
021276  012603                          MOV     (SP)+,R3        ;;RESTORE R3
021300  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
021302  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
021310  012766  021322  000004          MOV     #$TTYIN,4(SP)
021316  000002                          RTI                     ;;RETURN
021320     000          9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
021321     000                  .BYTE   0                       ;;TERMINATOR
021322                  $TTYIN: .BLKB   20.                     ;;RESERVE 20. BYTES FOR TTY INPUT
021346     136     103     015  $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL 'C'
021353     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
021360     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
021365     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
021376     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
                                .EVEN
2
3 021410  005737  000042          SHUT:   TST     @#42            ;ANY MONITOR PRESENT ?
4 021414  001002                          BNE     1$              ;BR IF YES
5 021416  000137  003312                  JMP     START2          ;GO TO 'START2'
6 021422  005037  001330          1$:     CLR     DRVSEL          ;FUDGE NO DRIVES SELECTED
7 021426  000137  015142                  JMP     $EOP            ;RETURN CONTROL TO MONITOR
```

```
      1                                 .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
                                        ;*SW11=1         INHIBIT ITERATIONS
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*CALL
                                        ;*      SCOPE           ;;SCOPE=IOT

      021432                            $SCOPE:
      021432  104407                            CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
      021434  032777  040000  157512    1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
      021442  001402                            BEQ     9$              ;;NO IF SW14=0
      021444  000137  022014                    JMP     $OVER           ;;JUMP OVER SCOPE ROUTINE
      021450                            9$:
                                        ;#####START OF CODE FOR THF XOR TESTER#####
      021450  000416                    $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
      021452  013746  000004                    MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
      021456  012737  021476  000004            MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
      021464  005737  177060                    TST     @#177060        ;;TIME OUT ON XOR?
      021470  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
      021474  000531                            BR      $SVLAD          ;;GO TO THE NEXT TEST
      021476  022626                    5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
      021500  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
      021504  000474                            BR      7$              ;;LOOP ON THE PRESENT TEST
      021506                            6$:;#####END OF CODE FOR THE XOR TESTER#####
      021506  105737  001117            2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
      021512  001502                            BEQ     3$              ;;BR IF NO
      021514  022737  177777  016332            CMP     #-1,CPSAVE      ;;SEE IF TIMEOUT WAS PREVIOUSLY RECORDED
      021522  001455                            BEQ     2003$           ;;KICK AROUND ROUTINE IF SO
      021524  013746  000004                    MOV     ERRVEC,-(SP)    ;;SAVE CONTENTS OF ERROR VECTOR
      021530  012737  021546  000004            MOV     #2000$,ERRVEC   ;;SETUP 'TRAP' RETURN ADDRESS
      021536  013737  177766  016332            MOV     177766,CPSAVE   ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
      021544  000406                            BR      2001$
      021546  012737  177777  016332  2000$:    MOV     #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
      021554  012716  021562                    MOV     #2001$,(SP)     ;;SETUP RETURN ADDRESS
      021560  000002                            RTI
      021562  012637  000004            2001$:  MOV     (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR

      021566  022737  17777.  016332  2002$:    CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
      021574  001430                            BEQ     2003$           ;;BRANCH IF SO
      021576  032737  000001  016332            BIT     #BIT00,CPSAVE   ;;SEE IF THE POWER MONITOR BIT IS ON
      021604  001424                            BEQ     2003$           ;;BRANCH TO CONTINUE ROUTINE IF CLEAR
      021606  042737  000001  177766            BIC     #BIT00,177766   ;;CLEAR THE BIT FOUND TO BE SET
      021614  013746  001154                    MOV     SWR,-(SP)       ;;SAVE SWR ADDRESS
      021620  017646  000000                    MOV     @(SP),-(SP)     ;;SAVE SWR VALUE
      021624  012737  000176  001154            MOV     #176,SWR        ;;GET SOFTWARE SWR ADDRESS
      021632  011677  157316                    MOV     (SP),@SWR       ;;GET CURRENT SWR VALUE
      021636  042777  001000  157310            BIC     #BIT09,@SWR     ;;DON'T ALLOW LOOP ON ERROR ON THIS ERROR
      021644  104177                            EMT     177             ;;CALL SPECIAL POWER FAIL BIT ERROR CALL
      021646  012676  000000                    MOV     (SP)+,@(SP)     ;;RESTORE SWR TO ORIGINAL VALUE
      021652  012637  001154                    MOV     (SP)+,SWR       ;;RESTORE SWR ADDRES
```

```
021656                                  2003$:
021656  123737  001131  001117                  CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
021664  101015                          BHI     3$              ;;BR IF NO
021666  032777  001000  157260          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
021674  001404                          BEQ     4$              ;;BR IF NO
021676  013737  001124  001122  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
021704  000443                          BR      $OVER
021706  105037  001117          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
021712  005037  001220                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
021716  000412                          BR      1$              ;;ESCAPE TO THE NEXT TEST
021720  032777  004000  157226  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
021726  001006                          BNE     1$              ;;BR IF YES
021730  005237  001120                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
021734  023737  001220  001120          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
021742  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
021744  012737  000001  001120  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
021752  013737  022030  001220          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
021760  105237  001116          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
021764  113737  001116  001240          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
021772  011637  001122                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
021776  011637  001124                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
022002  005037  001222                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
022006  112737  000001  001131          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
022014  013777  001116  157134  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
022022  013716  001122                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
022026  000002                          RTI                     ;;FIXES PS
022030  000001                  $MXCNT: 1                       ;;MAX. NUMBER OF ITERATIONS
```

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

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

022032                          $SAVREG:
022032  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022034  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
022036  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
022040  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
022042  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
022044  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
022046  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
022052  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
022056  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
022062  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
022066  000002                          RTI

                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
022070                          $RESREG:
022070  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
022074  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
022100  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
022104  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
022110  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
022112  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
022114  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
022116  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
022120  012601                          MOV     (SP)+,R1        ;;POP STACK INTO P1
022122  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
022124  000002                          RTI
```

```
                                .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.

022126  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
022130  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
022134  005740                          TST     -(R0)           ;;BACKUP BY 2
022136  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
022140  006300                          ASL     R0              ;;POSITION FOR INDEXING
022142  016000  022162                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
022146  000200                          RTS     R0              ;;GO TO ROUTINE


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

022150  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
022152  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
022160  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
022162  022150                  $TRPAD: .WORD   $TRAP2
022164  017116                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
022166  017476                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
022170  017452                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
022172  017512                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
022174  017700                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

022176  020524                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

022200  020434                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
022202  020776                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
022204  021066                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
022206  022032                          $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
022210  022070                          $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE
```

```
                                .SBTTL  SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE

                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
                                ;*UNSIGNED DECIMAL ASCIZ NUMBER.
                                ;*CALL
                                ;*      MOV     NUMBER,-(SP)    ;;PUT BINARY NUMBER ON THE STACK
                                ;*      JSR     PC,@#$SB2D      ;;CALL
                                ;*      RETURN                  ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK


022212  016637  000002  022242  $SB2D:  MOV     2(SP),1$        ;;SAVE BINARY NUMBER
022220  012746  022242                  MOV     #1$,-(SP)       ;;SET POINTER
022224  004737  022246                  JSR     PC,@#$DB2D      ;;CALL DOUBLE LENGTH CONVERT
022230  062716  000005                  ADD     #5,(SP)         ;;ONLY ALLOW FIVE CHARACTERS
022234  012666  000002                  MOV     (SP)+,2(SP)     ;;PICKUP POINTER
022240  000207                          RTS     PC              ;;RETURN
022242  000000  000000          1$:     .WORD   0,0
```

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

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


022246  104412                  $DB2D:  SAVREG                  ;;SAVE REGISTERS
022250  016602  000002                  MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
022254  012700  022426                  MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
022260  010066  000002                  MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
022264  012201                          MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
022266  012202                          MOV     (R2)+,R2
022270  012737  000012  022344          MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
022276  012704  022356                  MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
022302  012705  022360                  MOV     #$TNPWR+2,R5
022306  005003                  1$:     CLR     R3              ;;CLEAR PARTIAL
022310  161401                  2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
022312  005602                          SBC     R2
022314  161502                          SUB     (R5),R2
022316  002402                          BLT     3$              ;;BR IF TEN POWER TO LARGE
022320  005203                          INC     R3              ;;ADD 1 TO PARTIAL
022322  000772                          BR      2$              ;;LOOP
022324  062401                  3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
022326  005502                          ADC     R2
022330  062402                          ADD     (R4)+,R2
022332  022525                          CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
022334  052703  000060                  BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
022340  110320                          MOVB    R3,(R0)+        ;;SAVE IT
022342  005327                          DEC     (PC)+           ;;DONE?
022344  000000                  4$:     .WORD   0
022346  001357                          BNE     1$              ;;BR IF NO
022350  105020                          CLRB    (R0)+           ;;TERMINATOR
022352  104413                          RESREG                  ;;RESTORE REGISTERS
022354  000207                          RTS     PC              ;;RETURN
022356  145000                  $TNPWR: 145000                  ;;1.0E09
022360  035632                          35632
022362  160400                          160400                  ;;1.0E08
022364  002765                          2765
022366  113200                          113200                  ;;1.0E07
022370  000230                          230
022372  041100                          041100                  ;;1.0E06
022374  000017                          17
022376  103240                          103240                  ;;1.0E05
022400  000001                          1
022402  023420                          23420                   ;;1.0E04
022404  000000                          0
022406  001750                          1750                    ;;1.0E03
022410  000000                          0
022412  000144                          144                     ;;1.0E02
022414  000000                          0
```

```
022416  000012                          12                      ;;1.0E01
022420  000000                          0
022422  000001                          1                       ;;1.0E00
022424  000000                          0
022426                          $DECVL: .BLKB   12.             ;;RESERVE STORAGE FOR ASCII STRING
```

```
                                .SBTTL  TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS

                                ;;*****************************************************************
                                ;*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
                                ;*LEADING NUMBERS.
                                ;*CALL
                                ;*      MOV     #NUMADR,-(SP)   ;;FIRST ADDRESS OF ASCIZ STRING
                                ;*      JSR     PC,@#$SUPRS

022442  010046                  $SUPRS: MOV     R0,-(SP)        ;;SAVE R0
022444  016600  000004                  MOV     4(SP),R0        ;;PICKUP THE POINTER
022450  105710                  1$:     TSTB    (R0)            ;;TERMINATEOR?
022452  001403                          BEQ     2$              ;;BR IF YES
022454  122720  000060                  CMPB    #'0,(R0)+       ;;IS THIS AN ASCII "0" ?
022460  001773                          BEQ     1$              ;;BR IF YES
022462  005300                  2$:     DEC     R0              ;;BACKUP BY "1"
022464  010037  022472                  MOV     R0,3$           ;;SAVE FOR TYPING
022470  104401                          TYPE                    ;;GO TYPE
022472  000000                  3$:     .WORD   0               ;;ASCIZ POINTER GOES HERE
022474  012600                          MOV     (SP)+,R0        ;;RESTORE R0
022476  012616                          MOV     (SP)+,(SP)      ;;RESTORE THE STACK
022500  000207                          RTS     PC              ;;RETURN
```

```
                                .SBTTL  RANDOM NUMBER GENERATOR ROUTINE

                                ;;*****************************************************************
                                ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
                                ;*WITH A RANGE OF 0 TO 2(+33)-1.
                                ;*CALL:
                                ;*      JSR     PC,$RAND        ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;RETURN HERE THE RANDOM
                                ;*                              ;;NUMBER WILL BE IN
                                ;*                              ;;$HINUM,$LONUM

022502                          $RAND:
022502  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
022504  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
022506  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
022510  013700  022602                 MOV     $LONUM,R0       ;;SET R0 WITH LOW
022514  013701  022600                 MOV     $HINUM,R1       ;;SET R1 WITH HIGH
022520  012702  177771                 MOV     #-7,R2          ;;SET SHIFT COUNT
022524  006300                 1$:     ASL     R0              ;;SHIFT R0 LEFT AND
022526  006101                         ROL     R1              ;;ROTATE CARRY INTO R1 AND
022530  005202                         INC     R2              ;;CHECK FOR DONE
022532  001374                         BNE     1$              ;;CONTINUE SHIFT LOOP
022534  063700  022602                 ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
022540  005501                         ADC     R1              ;;PROPOGATE CARRY
022542  063701  022600                 ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
022546  062700  001057                 ADD     #1057,R0        ;;ADD LOW CONSTANT
022552  005501                         ADC     R1              ;;PROPOGATE CARRY
022554  062701  047401                 ADD     #47401,R1       ;;ADD HIGH CONSTANT
022560  010037  022602                 MOV     R0,$LONUM       ;;SAVE R0
022564  010137  022600                 MOV     R1,$HINUM       ;;SAVE R1
022570  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
022572  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
022574  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
022576  000207                         RTS     PC              ;;RETURN
022600  176543                 $HINUM: .WORD   176543
022602  123456                 $LONUM: .WORD   123456
```

```
                                .SBTTL  INTEGER DIVIDE ROUTINE

                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
                                ;*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
                                ;*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
                                ;*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
                                ;*SAME SIGN AS THE DIVIDEND.
                                ;*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
                                ;*
                                ;*      STACK   NO ERROR        OVERFLOW        DIVIDE BY ZERO
                                ;*      -----   --------        --------        -------------
                                ;*      TOP     REMAINDER       ALL ZEROS       ALL ONES
                                ;*      +2      QUOTIENT        ALL ZEROS       ALL ONES
                                ;*
                                ;*
                                ;*NOTE: THIS ROUTINE WILL LINK TO THE DIVISION SUBROUTINE ('M.DPID').

022604  104412                  $DIV:   SAVREG                          ;STORE R0 - R5
022606  016605  000026                  MOV     26(SP),R5               ;DIVISOR
022612  005004                          CLR     R4                      ;OTHER DIVISOR WORD
022614  016602  000030                  MOV     30(SP),R2               ;UPPER DIVIDEND WORD
022620  016603  000032                  MOV     32(SP),R3               ;LOWER DIVIDEND WORD
022624  005000                          CLR     R0                      ;CLEAR OTHER DIVIDEND REGISTERS
022626  005001                          CLR     R1
022630  004737  022652                  JSR     PC,M.DPID               ;GO TO THE DIVIDE ROUTINE
022634  010166  000030                  MOV     R1,30(SP)               ;REMAINDER ON THE STACK
022640  010366  000032                  MOV     R3,32(SP)               ;QUOTIENT ON THE STACK
022644  104413                          RESREG                          ;RESTORE R0 - R5
022646  012616                          MOV     (SP)+,(SP)              ;MOVE RETURN UP THE STACK
022650  000207                          RTS     PC
```

```
                                .SBTTL  DOUBLE PRECISION DIVISION SUBROUTINE
                                ;
                                ;CALL
                                ;       JSR     PC,M.DPID
                                ;
                                ;       DIVIDEND = R0-R1-R2-R3 (R0=MSD)
                                ;       DIVISOR = R4-R5 (R4=MSD)
                                ;
                                ;RETURN
                                ;
                                ;       REMAINDER AFTER DIVISION = R0-R1 (R0=MSD)
                                ;       QUOTIENT AFTER DIVISION = R2-R3 (R2=MSD)
                                ;
022652  012746  000040        M.DPID: MOV     #40,-(SP)       ;COUNTER FOR DIVISION CYCLES
022656  010446                        MOV     R4,-(SP)        ;HIGH ORDER
022660  010546                        MOV     R5,-(SP)        ;LOW ORDER DIVISOR TO THE STACK
022662  005466  000002                NEG     2(SP)           ;FORM NEGATIVE
022666  005416                        ...     @SP             ;VERSION OF THE DIVISOR
022670  005666  000002                SBC     2(SP)
022674  061601                        ADD     @SP,R1
022676  005500                        ADC     R0              ;PERFORM THE INITIAL SUBTRACTION
022700  066600  000002                ADD     2(SP),R0
022704  103445                        BCS     M.DP50          ;IF CARRY THEN OVERFLOW HAS OCCURRED
022706  005046                        CLR     -(SP)           ;THIS IS A LONGER LASTING CARRY BIT
022710  006103                M.DP40: ROL     R3
022712  006102                        ROL     R2
022714  006101                        ROL     R1
022716  006100                        ROL     R0
022720  005716                        TST     @SP             ;TEST "CARRY" INDICATOR
022722  001410                        BEQ     M.DP41          ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
022724  005016                        CLR     @SP             ;CLEAR UP FOR NEXT TIME
022726  066601  000002                ADD     2(SP),R1
022732  005500                        ADC     R0              ;ADD -(DIVISOR)
022734  005516                        ADC     @SP     ;  I    ;SET "CARRY"
022736  066600  000004                ADD     4(SP),R0;<-
022742  000404                        BR      M.DP42
022744  060501                M.DP41: ADD     R5,R1
022746  005500                        ADC     R0              ;ADD +(DIVISOR)
022750  005516                        ADC     @SP     ;  I    ;SET "CARRY"
022752  060400                        ADD     R4,R0   ;<-
022754  005516                M.DP42: ADC     @SP             ;SET "CARRY"
022756  005716                        TST     @SP             ;TEST THE UPDATE INDICATOR
022760  001401                        BEQ     .+4     ;->     ;IF ZERO FORGET IT
022762  005203                        INC     R3      ;  I    ;NO CARRY POSSIBLE HERE
022764  005366  000006                DEC     6(SP)   ;<-     ;DECREMENT COUNTER
022770  003347                        BGT     M.DP40          ;BRANCH IF MORE TO DO
022772  006003                        ROR     R3
022774  103404                        BCS     M.DP44
022776  060501                        ADD     R5,R1
023000  005500                        ADC     R0
023002  060400                        ADD     R4,R0
023004  000241                        CLC
023006  006103                M.DP44: ROL     R3
023010  062706  000010                ADD     #10,SP          ;ADJUST STACK BY 4 WORDS
023014  000242                        CLV
023016  000207                        RTS     PC
023020  062706  000006        M.DP50: ADD     #6,SP
```

```
     58 023024  000262                        SEV
     59 023026  000207                        RTS     PC
```

```
                                        ;THIS ROUTINE IS USED TO INCREMENT THE ERROR COUNT FOR EACH DRIVE BEING
                                        ;TESTED. IF THE TOTAL ERRORS ON ANY DRIVE EXCEEDS THE MAXIMUM ALLOWABLE
                                        ;ERRORS DESIGNATED IN LOCATION 'ERMAX', A MESSAGE WILL BE TYPED AND THE
                                        ;DRIVE IN ERROR WILL BE DROPPED FROM THE TEST.

023030  010046                  INCEC:  MOV     R0,-(SP)        ;SAVE R0
023032  013700  031366                  MOV     RMADR,R0        ;GET RM BASE ADDRESS
023036  016000  000010                  MOV     RMCS2(R0),R0    ;GET CONTENTS OF RMCS2
023042  042700  177770                  BIC     #^C7,R0         ;SAVE UNIT SELECT BITS
023046  136037  031356  001330          BITB    ATABIT(R0),DRVSEL       ;WAS THIS DRIVE SELECTED FOR TEST
023054  001432                          BEQ     1$              ;BR IF NO
023056  105260  001512                  INCB    ERRCN(R0)       ;INCREMENT ERROR COUNT
023062  126037  001512  001316          CMPB    ERRCN(R0),ERMAX ;EXCEEDED ERROR LIMIT ?
023070  103424                          BLO     1$              ;BR IF NO
023072  104401  001231                  TYPE    ,$CRLF          ;CR-LF
023076  104401  037273                  TYPE    ,MSDRIV         ;TYPE 'DRIVE'
023102  010046                          MOV     R0,-(SP)        ;;SAVE R0 FOR TYPEOUT
023104  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
023106     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
023107     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
023110  104401  037302                  TYPE    ,DROP           ;TYPE ' DROPPED'
023114  104401  037160                  TYPE    ,COMMA          ;TYPE ', '
023120  104401  037313                  TYPE    ,EXCEED         ;TYPE 'EXCEEDED MAXIMUM ERROR LIMIT'
023124  146037  031356  001330          BICB    ATABIT(R0),DRVSEL       ;DESELECT DRIVE FROM TEST
023132  005337  015314                  DEC     $EOPCT          ;ADJUST 'EOP' COUNT
023136  000137  015142                  JMP     $EOP            ;RETURN TO $EOP
023142  012600                  1$:     MOV     (SP)+,R0        ;RESTORE R0
023144  000207                          RTS     PC

                                        ;THIS SUBROUTINE CLEARS THE MASSBUS CONTROLLER, MASSBUS ADAPTER,
                                        ;AND DRIVERS, THEN SELECTS THE DRIVE.
                                        ;CALL:
                                        ;       JSR     PC,CNTCLR       ;CALL TO ROUTINE
                                        ;

023146  013704  031366          CNTCLR: MOV     RMADR,R4        ;GET RMCS1 BASE ADDRESS
023152  012764  000040  000010          MOV     #CLR,RMCS2(R4)  ;ISSUE MASSBUS CLEAR AND
023160  042737  177770  001352          BIC     #^C7,CHKDRV     ;SAVE UNIT SELECT BITS
023166  013764  001352  000010          MOV     CHKDRV,RMCS2(R4)        ;SELECT THE DRIVE
023174  000207                          RTS     PC              ;RETURN

                                        ;SET "LPTAVL" TO THE PROPER STATE.
                                        ; LPTAVL = 0 IF NO LINE PRINTER AVAILABLE
                                        ; LPTAVL = 1 IF LINE PRINTER IS AVAILABLE
                                        ;CALL
                                        ;       JSR     PC,LP.AVL
                                        ;       RETURN

023176  005037  001326          LP.AVL: CLR     LPTAVL  ;START WITH NO PRINTER AVAIABLE
023202  012737  023226  000004          MOV     #1$,ERRVEC      ;SETUP THE TIMEOUT VECTOR
023210  005037  000006                  CLR     ERRVEC+2
023214  005777  156334                  TST     @LPS            ;IS THERE A LINE PRINTER?
023220  005237  001326                  INC     LPTAVL  ;YES--SET AVAILABLE SWITCH
023224  000401                          BR      2$
023226  022626                  1$:     CMP     (SP)+,(SP)+     ;NO--POP STACK
023230  012737  000006  000004  2$:     MOV     #ERRVEC+2,ERRVEC        ;RESTORE TIMEOUT VECTOR
023236  000207                          RTS     PC              ;RETURN
```

```
                                        ;THIS ROUTINE WILL DETERMINE IF THERE IS A CLOCK ON THE SYSTEM
                                        ;AND IF THERE IS IT WILL SETUP THE VECTOR AND START THE CLOCK
                                        ;"CLKSTA" WILL INDICATE THE CLOCK TYPE
                                        ; 0= NO CLOCK
                                        ;+1= KW11-P
                                        ;-1= KW11-L
                                        ;THIS ROUTINE WILL ALSO SETUP "TICKMS" (TIME
                                        ;PER CLOCK TICK IN MILLISECONDS) AND "TICKUS"
                                        ;(TIME PER CLOCK TICK IN MICROSECONDS) AS
                                        ;PER SW00.
                                        ;SW00=0 -- 60HZ
                                        ;SW00=1 -- 50HZ
                                        ;CALL
                                        ;       JSR     PC,ST.CLK
                                        ;       RETURN

023240  010146                 ST.CLK: MOV     R1,-(SP)        ;SAVE R1
023242  012701  000006                 MOV     #ERRVEC+2,R1    ;SAVE AND SETUP TIMEOUT VECTOR
023246  011146                         MOV     (R1),-(SP)
023250  005011                         CLR     (R1)            ;LEVEL 0
023252  014146                         MOV     -(R1),-(SP)
023254  012711  023304                 MOV     #1$,(R1)        ;GO TO 1$ ON TIMEOUT
023260  005037  001342                 CLR     CLKSTA          ;SET CLOCK STATUS TO NO CLOCK
023264  005777  156244                 TST     @PKCS           ;IS THERE A KW11-P?
023270  012737  000001  001342         MOV     #1,CLKSTA       ;YES--SET STATUS TO KW11-P
023276  004737  023406                 JSR     PC,ST.PCLK      ;START THE KW11-P
023302  000414                         BR      3$              ;GO TO EXIT
023304  022626                 1$:     CMP     (SP)+,(SP)+     ;CLEAN UP THE STACK
023306  012711  023332                 MOV     #2$,(R1)        ;IF TIMEOUT GO TO 2$
023312  005777  156230                 TST     @LKS            ;IS THERE A KW11-L?
023316  012737  177777  001342         MOV     #-1,CLKSTA      ;YES-- SET STATUS TO KW11-L
023324  004737  023450                 JSR     PC,ST.LCLK      ;START THE KW11-L
023330  000401                         BR      3$              ;EXIT
023332  022626                 2$:     CMP     (SP)+,(SP)+     ;CLEAN UP THE STACK
023334  012621                 3$:     MOV     (SP)+,(R1)+     ;RESTORE THE TIMEOUT VECTOR
023336  012621                         MOV     (SP)+,(R1)+
023340  012601                         MOV     (SP)+,R1        ;RESTORE R1
023342  032737  000100  001314         BIT     #SW06,C.SWR     ;50HZ OR 60HZ?
023350  001407                         BEQ     4$              ;BRANCH IF 60
023352  012737  000020  001344         MOV     #20,TICKMS      ;SETUP TIME PER
023360  012737  047040  001346         MOV     #20000.,TICKUS  ;TICK FOR 50HZ
023366  000406                         BR      5$
023370  012737  000016  001344 4$:     MOV     #16,TICKMS      ;SETUP TIME PER
023376  012737  040432  001346         MOV     #16666.,TICKUS  ;TICK FOR 60HZ
023404  000207                 5$:     RTS     PC              ;RETURN

023406                          ST.PCLK:
023406  032737  000040  001314         BIT     #SW05,C.SWR     ;ALLOW SOFTWARE TIMEOUTS?
023414  001014                         BNE     1$              ;NO--BRANCH
023416  012777  023504  156104         MOV     #SRVCLK,@PKV    ;SETUP THE KW11-P VECTOR
023424  012777  000300  156100         MOV     #300,@PKV+2
023432  012777  000001  156076         MOV     #1,@PKB         ;COUNT ONE TICK
023440  012777  000115  156066         MOV     #115,@PKCS      ;"INT.EN.",COUNT DOWN", "MODE 1 (REPEAT)",
                                                                ;"LINE FREQ", AND "RUN"
023446  000207                 1$:     RTS     PC              ;RETURN
```

```
                                        ST.LCLK:
023450                                          BIT     #SW05,C.SWR     ;ALLOW SOFTWARE TIMEOUTS?
023450  032737  000040  001314                  BNE     1$              ;NO--BRANCH
023456  001011                                  MOV     #SRVCLK,@LKV    ;SETUP THE KW11-L VECTOR
023460  012777  023504  156054                  MOV     #300,@LKV+2
023466  012777  000300  156050                  MOV     #100,@LKS       ;START THE KW11-L
023474  012777  000100  156044          1$:     RTS     PC              ;RETURN
023502  000207
                                        SRVCLK: MOV     TICKMS,-(SP)    ;TIME PER TICK IN MILLISECONDS
023504  013746  001344                          JSR     PC,RMTMR        ;COUNT THE ELASPED TIME
023510  004737  035034                          RTI                     ;RETURN AFTER INTERRUPT
023514  000002
                                        ;THIS ROUTINE SETS UP DEFAULT PARAMETER VALUES WHEN THE PROGRAM IS
                                        ;STARTED OR WHEN THE VALUE OF BIT00 IN 'C.SWR' IS CHANGED.
                                        ;CALL
                                        ;       JSR     PC,LODFLT
                                        ;       RETURN
                                        LODFLT:
023516                                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
023516  010046                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
023520  010146                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
023522  010246                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
023524  010346                                  MOV     #13377,TSTNMS   ;SELECT TESTS 0-7(SEEK) AND 11,12 & 14(TIMING)
023526  012737  013377  001332                  MOV     #0,TSTNMS+2     ;SELECT TESTS
023534  012737  000000  001334                  MOV     #DFLT,R0        ;DEFAULT PARAMETERS POINTER
023542  012700  002456                          MOV     #PRM0,R1        ;TABLE POINTER
023546  012701  002726                          MOV     R1,R2           ;STOP ADDRESS
023552  0101C2                                  MOV     (R0)+,(R1)+     ;MOVE DEFAULT PARAMETERS INTO
023554  012021                          1$:     CMP     R0,R2           ;RUN TIME TABLES ** DONE?
023556  020002                                  BLO     1$              ;NO--BRANCH
023560  103775                                  BIT     #BIT00,C.SWR    ;16 BIT MODE ?
023562  032737  000001  001314                  BNE     3$              ;BR IF 18 BIT MODE
023570  001012                                  MOV     #30.,PRMLMT+16  ;SET 'FS' LIMIT TO 30.
023572  012737  000036  002426                  MOV     #30.,PRMLMT+20  ;SET 'LS' LIMIT TO 30.
023600  012737  000036  002430                  MOV     #-<256.*31.>,TRCKWC     ;WORD COUNT FOR A 16 BIT TRACK
023606  012737  160400  001452                  BR      4$              ;CONTINUE
023614  000411                                  MOV     #29.,PRMLMT+16  ;SET 'FS' LIMIT TO 29.
023616  012737  000035  002426          3$:     MOV     #29.,PRMLMT+20  ;SET 'LS' LIMIT TO 29.
023624  012737  000035  002430                  MOV     #-<256.*30.>,TRCKWC     ;WORD COUNT FOR AN 18 BIT TRACK
023632  012737  161000  001452                  MOV     #PRMPT,R1       ;ADDRESS OF PARAMETER POINTER TABLE
023640  012701  002352                  4$:     TST     (R1)            ;END OF THE TABLE ?
023644  005711                          5$:     BEQ     8$              ;BR IF END
023646  001425                                  BIT     #BIT08,@(R1)+   ;'LS' SELECTED ?
023650  032731  000400                          BEQ     5$              ;BR IF NOT
023654  001773                                  MOV     -2(R1),R2       ;PARAMETER TABLE ADDRESS
023656  016102  177776                          MOV     (R2),-(SP)      ;PARAMETER ALLOCATION BITS
023662  011246                                  MOV     #9.,R3          ;NUMBER OF PARAMETERS (MAXIMUM) BEFORE 'LS'
023664  012703  000011                  6$:     ASR     (SP)            ;COUNT THE PARAMETER
023670  006216                                  BCC     7$              ;BR IF NOT USED
023672  103002                                  ADD     #2,R2           ;INCREMENT THE PARAMETER TABLE ADDRESS
023674  062702  000002                  7$:     DEC     R3              ;COUNT THE PARAMETER
023700  005303                                  BNE     6$              ;BR IF NOT THERE YET
023702  001372                                  TST     (SP)+           ;CORRECT THE STACK POINTER
023704  005726                                  CMP     (R2),PRMLMT+16  ;IS 'LS' TOO LARGE FOR THE MODE SELECTED ?
023706  021237  002426                          BLOS    5$              ;BR IF NOT
023712  101754                                  MOV     PRMLMT+16,(R2)  ;RESET VALUE FOR MODE USED
023714  013712  002426
```

```
                                                        BR      5$              ;CONTINUE
                                                8$:
                                                        MOV     (SP)+,R3        ;;POP STACK INTO R3
                                                        MOV     (SP)+,R2        ;;POP STACK INTO R2
                                                        MOV     (SP)+,R1        ;;POP STACK INTO R1
                                                        MOV     (SP)+,R0        ;;POP STACK INTO R0
                                                        RTS     PC              ;RETURN

                                                ;THIS ROUTINE FILLS THE PARAMETER TABLE THE CURRENT TEST.
                                                ;CALL
                                                ;       MOV     #TESTNUM,$TSTNM ;LOAD THE TEST NUMBER
                                                ;       JSR     PC,LODPRM
                                                ;       RETURN

                                                LODPRM:
                                                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
                                                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
                                                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
                                                        CLR     R4              ;CLEAR R4
                                                        MOVB    $TSTNM,R4       ;GET THE TEST NUMBER
                                                        ASL     R4              ;SETUP TO ADDRESS WORDS
                                                        MOV     PRMPT(R4),R1    ;GET THE TEST'S PARAMETER TABLE ADDRESS
                                                        MOV     #PRM,R2         ;PARAMETER EXECUTION TABLE
                                                        MOV     CHKDRV,R4       ;GET DRIVE ADDRESS
                                                        MOV     (R1)+,(R2)+     ;LOAD PARAMETER SPECIFIER
                                                1$:     ASR     PRM             ;IS THIS PARAMETER USED IN THE TEST ?
                                                        BCC     2$              ;BR IF NOT
                                                        MOV     (R1)+,(R2)+     ;LOAD THE VALUE
                                                        BR      3$              ;CONTINUE
                                                2$:     CLR     (R2)+           ;CLEAR THE UNUSED PARAMETER LOCATION
                                                3$:     CMP     #PAT+2,R2       ;FINISHED ?
                                                        BEQ     4$              ;BR IF YES
                                                        MOV     PRMLMT+12,LSTRK ;GET LAST TRACK FOR AN RM80
                                                        BR      1$              ;KEEP GOING
                                                4$:
                                                        MOV     (SP)+,R4        ;;POP STACK INTO R4
                                                        MOV     (SP)+,R2        ;;POP STACK INTO R2
                                                        MOV     (SP)+,R1        ;;POP STACK INTO R1
                                                        RTS     PC              ;RETURN

                                                ;THIS ROUTINE LOADS A READ HEADER AND DATA COMMAND OR A SEEK COMMAND
                                                ;INTO DPB.B+2 AND DPB.C+2, DEPENDING ON THE STATE OF "CONTROL SWITCH"
                                                ;BIT07.
                                                ;CALL
                                                ;       JSR     PC,LDCMD
                                                ;       RETURN

                                                LDCMD:  BIT     #SW07,C.SWR     ;DO EXPLICIT SEEKS?
                                                        BNE     1$              ;YES--BRANCH
                                                        MOV     #READHD,DPB.B+2 ;NO--SET UP FOR READ HEADER AND
                                                        MOV     #READHD,DPB.C+2 ;DATA COMMAND
                                                        BR      2$
                                                1$:     MOV     #SEEK,DPB.B+2   ;SETUP FOR SEEK COMMAND
                                                        MOV     #SEEK,DPB.C+2
                                                2$:     RTS     PC

                                                ;THIS ROUTINE WILL CALL THE RM80 DRIVER AND THEN WAIT ON THE FUNCTION
```

```
                                        ;TO COMPLETE. IF AN ERROR OCCURS IT IS REPORTED.
                                        ;CALL
                                        ;       FILL 'DPB'' WITH COMMAND INFORMATION
                                        ;       JSR     R0,CALL.A
                                        ;       RETURN
                                        ;
024076  005037  001222          CALL.A: CLR     $ESCAPE ;NO ESCAPE ADDRESS
024102  004037  032054                  JSR     R0,RM80         ;CALL RM80 DRIVER
024106  036506                          DPB.A
024110  000772                          BR      CALL.A
024112  005737  036524          1$:     TST     DPB.A+16        ;DONE?
024116  001775                          BEQ     1$              ;NO--LOOP
024120  100050                          BPL     5$              ;BRANCH IF NO ERROR
024122  012737  024216  001222          MOV     #3$,$ESCAPE     ;;ESCAPE TO 3$ ON ERROR
024130  013737  036520  001366          MOV     DPB.A+12,CYL.DS ;CYLINDER
024136  113737  036517  001372          MOVB    DPB.A+11,TRK.DS ;TRACK
024144  113737  036516  001370          MOVB    DPB.A+10,SEC.DS ;SECTOR
024152  012746  036524                  MOV     #DPB.A+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
024156  004737  025062                  JSR     PC,ERINDX       ;FORM DISPATCH INDEX
024162  062607                          ADD     (SP)+,PC        ;REPORT PROPER ERROR
024164  104041                          EMT     41              ;NON-EXIST DRIVE
024166  104042                          EMT     42              ;PORT TIMEOUT
024170  104043                          EMT     43              ;UNSAFE ERROR
024172  104044                          EMT     44              ;NON-I/O ERROR
024174  000240                          NOP                     ;TO SYNC THE CALLING SEQ OF ERINDX
024176  005737  036654                  TST     RM.REG+RMER1    ;ANY DRIVE ERROR
024202  001004                          BNE     2$              ;BRANCH IF SO
024204  032737  100000  036702          BIT     #BSE,RM.REG+RMER2       ;BAD SPOT ERROR
024212  001007                          BNE     4$              ;BRANCH IF SO
024214                          2$:
024214  104045                          EMT     45
024216  032737  040000  036702  3$:     BIT     #SKI,RM.REG+RMER2       ;SKI ERROR ?
024224  001402                          BEQ     4$              ;BR IF NO
024226  004037  024710                  JSR     R0,CALL.R       ;DO RECALIBRATE COMMAND
024232  013746  036524          4$:     MOV     DPB.A+16,-(SP)  ;STATUS WORD
024236  004737  025022                  JSR     PC,LOP.CK       ;SEE IF LOOP, ABORT, OR CONTINUE
024242  000200                  5$:     RTS     R0              ;RETURN

                                        ;THIS ROUTINE IS THE SAME AS ''CALL.A'' EXCEPT FOR THE DPB USED AND IF
                                        ;THE COMMAND IS A READ HEADER AND DATA THE HEADER (CYLINDER, TRACK,
                                        ;AND SECTOR) READ IS CHECKED FOR VALIDITY.
                                        ;CALL
                                        ;       FILL DPB
                                        ;       JSR     R0,CALL.B
                                        ;       RETURN

024244  005037  001222          CALL.B: CLR     $ESCAPE         ;NO ESCAPE ADDRESS
024250  005037  036550                  CLR     DPB.B+$SSENB    ;CLEAR SKIP SECTOR ENABLED INDICATOR
024254  004037  032054                  JSR     R0,RM80         ;CALL RM80 DRIVER
024260  036530                          DPB.B
024262  000770                          BR      CALL.B
024264  005737  036546          1$:     TST     DPB.B+16        ;DONE?
024270  001775                          BEQ     1$              ;NO--BRANCH
024272  100055                          BPL     5$              ;BRANCH IF NO ERROR
024274  012737  024400  001222          MOV     #3$,$ESCAPE     ;;ESCAPE TO 3$ ON ERROR
024302  013737  036542  001366          MOV     DPB.B+12,CYL.DS ;CYLINDER
024310  113737  036541  001372          MOVB    DPB.B+11,TRK.DS ;TRACK
```

```
                024316  113737  036540  001370          MOVB    DPB.B+10,SEC.DS ;SECTOR
                024324  012746  036546                  MOV     #DPB.B+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
                024330  004737  025062                  JSR     PC,ERINDX       ;FORM DISPATCH INDEX
                024334  062607                          ADD     (SP)+,PC        ;REPORT PROPER ERROR
                024336  104041                          EMT     41              ;NON-EXIST DRIVE
                024340  104042                          EMT     42              ;PORT TIMEOUT
                024342  104043                          EMT     43              ;UNSAFE ERROR
                024344  104044                          EMT     44              ;NON-I/O ERROR
    265         024346  000240                          NOP                     ;TO SYNC THE CALLING SEQ OF ERINDX: RT.
    266         024350  005737  036654                  TST     RM.REG+RMER1    ;DRIVE ERROR ?
    267         024354  001404                          BEQ     2$              ;BR IF NOT
    268         024356  022737  000200  036654          CMP     #HCE,RM.REG+RMER1       ;SEE IF ONLY 'HCE' SET
    269         024364  001420                          BEQ     5$              ;BR IF YES
    270         024366  032737  100000  036702  2$:     BIT     #BSE,RM.REG+RMER2       ;BAD SPOT ERROR
    271         024374  001033                          BNE     7$              ;BRANCH IF SO
    272         024376  104045                          EMT     45
    273         024400  032737  040000  036702  3$:     BIT     #SKI,RM.REG+RMER2       ;SKI ERROR ?
    274         024406  001402                          BEQ     4$              ;BR IF NO
    275         024410  004037  024710                  JSR     R0,CALL.R       ;DO RECALIBRATE COMMAND
    276         024414  013746  036546          4$:     MOV     DPB.B+16,-(SP)  ;STATUS WORD
    277         024420  004737  025022                  JSR     PC,LOP.CK       ;SEE IF LOOP, ABORT, OR CONTINUE
    278         024424  000410                          BR      6$              ;CHECK FOR STALL
    279         024426  123727  036532  000173  5$:     CMPB    DPB.B+2,#READHD ;DOING IMPLIED SEEKS?
    280         024434  001004                          BNE     6$              ;NO--BRANCH
    281         024436  004037  025302                  JSR     R0,VERIFY       ;YES--GO CHECK THE DATA
    282         024442  036540                          DPB.B+10
    283         024444  000407                          BR      7$              ;ERROR DURING VERIFY
    284         024446  032737  040000  001314  6$:     BIT     #SW14,C.SWR     ;STALL?
    285         024454  001403                          BEQ     7$              ;NO--BRANCH
    286         024456  004037  025220                  JSR     R0,STALL        ;YES--CALL STALL ROUTINE
    287         024462  001454                          .WORD   STALL1          ;STALL TIME POINTER
    288         024464  000200                  7$:     RTS     R0              ;RETURN
    289
    290                                                 ;THIS ROUTINE IS THE SAME AS "CALL.B" EXCEPT FOR THE DPB USED.
    291                                                 ;CALL
    292                                                 ;       FILL DPB
    293                                                 ;       JSR     R0,CALL.C
    294                                                 ;       RETURN
    295
    296         024466  005037  001222          CALL.C: CLR     $ESCAPE         ;NO ESCAPE ADDRESS
    297         024472  005037  036572                  CLR     DPB.C+$SSENB    ;CLEAR SKIP SECTOR ENABLED INDICATOR
    298         024476  004037  032054                  JSR     R0,RM80         ;CALL RM80 DRIVER
    299         024502  036552                          DPB.C
    300         024504  000770                          BR      CALL.C
    301         024506  005737  036570          1$:     TST     DPB.C+16        ;DONE?
    302         024512  001775                          BEQ     1$              ;NO--LOOP
    303         024514  100055                          BPL     5$              ;YES--BRANCH IF NO ERROR
    304         024516  012737  024622  001222          MOV     #3$,$ESCAPE     ;;ESCAPE TO 3$ ON ERROR
    305         024524  013737  036564  001366          MOV     DPB.C+12,CYL.DS ;CYLINDER
                024532  113737  036563  001372          MOVB    DPB.C+11,TRK.DS ;TRACK
                024540  113737  036562  001370          MOVB    DPB.C+10,SEC.DS ;SECTOR
                024546  012746  036570                  MOV     #DPB.C+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
                024552  004737  025062                  JSR     PC,ERINDX       ;FORM DISPATCH INDEX
                024556  062607                          ADD     (SP)+,PC        ;REPORT PROPER ERROR
                024560  104041                          EMT     41              ;NON-EXIST DRIVE
                024562  104042                          EMT     42              ;PORT TIMEOUT
                024564  104043                          EMT     43              ;UNSAFE ERROR
```

```
                024566  104044                          EMT     44              ;NON-I/O ERROR
                024570  000240                          NOP                     ;TO SYNC THE CALLING SEQ OF ERINDX: RT.
                024572  005737  036654                  TST     RM.REG+RMER1    ;DRIVE ERROR ?
                024576  001404                          BEQ     2$              ;BR IF NOT
                024600  022737  000200  036654          CMP     #HCE,RM.REG+RMER1       ;SEE IF ONLY 'HCE' SET
                024606  001420                          BEQ     5$              ;BR IF YES
                024610  032737  100000  036702  2$:     BIT     #BSE,RM.REG+RMER2       ;BAD SPOT ERROR ONLY ?
                024616  001033                          BNE     7$              ;BRANCH IF SO
                024620  104045                          EMT     45
                024622  032737  040000  036702  3$:     BIT     #SKI,RM.REG+RMER2       ;SKI ERROR ?
                024630  001402                          BEQ     4$              ;BR IF NO
                024632  004037  024710                  JSR     R0,CALL.R       ;DO RECALIBRATE COMMAND
                024636  013746  036570          4$:     MOV     DPB.C+16,-(SP)  ;STATUS WORD
                024642  004737  025022                  JSR     PC,LOP.CK       ;SEE IF LOOP, ABORT, OR CONTINUE
                024646  000410                          BR      6$
                024650  123727  036554  000173  5$:     CMPB    DPB.C+2,#READHD ;DOING IMPLIED SEEK?
                024656  001004                          BNE     6$              ;NO--EXIT
                024660  004037  025302                  JSR     R0,VERIFY       ;YES--CHECK THE DATA
                024664  036562                          DPB.C+10
                024666  000407                          BR      7$              ;ERROR DURING VERIFY
                024670  032737  040000  001314  6$:     BIT     #SW14,C.SWR     ;STALL?
                024676  001403                          BEQ     7$              ;NO--BRANCH
                024700  004037  025220                  JSR     R0,STALL        ;YES--CALL STALL ROUTINE
                024704  001454                          .WORD   STALL1          ;STALL TIME POINTER
                024706  000200                  7$:     RTS     R0

                                        ; THIS ROUTINE WILL ISSUE A RECALIBRATE COMMAND TO THE RM80 DRIVER
                                        ; AND WAIT FOR THE FUNCTION TO COMPLETE.
                                        ;CALL
                                        ;       JSR     R0,CALL.R
                                        ;       RETURN

                024710  005037  001222  CALL.R: CLR     $ESCAPE         ;NO ESCAPE ADDRESS
                024714  004037  032054          JSR     R0,RM80         ;CALL RM80 DRIVER
                024720  036616                  DPB.R
                024722  000772                  BR      CALL.R
                024724  005737  036634  1$:     TST     DPB.R+16        ;DONE?
                024730  001775                  BEQ     1$              ;NO--LOOP
                024732  100032                  BPL     3$              ;BRANCH IF NO ERROR
                024734  012737  025010  001222  MOV     #2$,$ESCAPE     ;;ESCAPE TO 2$ ON ERROR
                024742  013737  036630  001366  MOV     DPB.R+12,CYL.DS ;CYLINDER
                024750  113737  036627  001372  MOVB    DPB.R+11,TRK.DS ;TRACK
                024756  113737  036626  001370  MOVB    DPB.R+10,SEC.DS ;SECTOR
                024764  012746  036634          MOV     #DPB.R+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
                024770  004737  025062          JSR     PC,ERINDX       ;FORM DISPATCH INDEX
                024774  062607                  ADD     (SP)+,PC        ;REPORT PROPER ERROR
                024776  104041                  EMT     41              ;NON-EXIST DRIVE
                025000  104042                  EMT     42              ;PORT TIMEOUT
                025002  104043                  EMT     43              ;UNSAFE ERROR
                025004  104044                  EMT     44              ;NON-I/O ERROR
                025006  000240                  NOP                     ;TO SYNC THE CALLING SEQ OF ERINDX
                025010  013746  036524  2$:     MOV     DPB.A+16,-(SP)  ;STATUS WORD
                025014  004737  025022          JSR     PC,LOP.CK       ;SEE IF LOOP, ABORT, OR CONTINUE
                025020  000200          3$:     RTS     R0              ;RETURN

                                        ;THIS SUBROUTINE CHECK FOR LOOP, ABORT, OR CONTINUE SWITCHES AFTER
```

```
                                        ;ERRORS 41, 42, 43, 44, 45, AND 46.
                                        ;CALL
                                        ;       MOV     DTA+16,-(SP)    ;STATUS WORD FROM DPB IN USE
                                        ;       JSR     PC,LOP.CK
                                        ;       RETURN
025022  032777  001000  154124  LOP.CK: BIT     #SW9,@SWR       ;LOOP ON ERROR
025030  001402                          BEQ     1$              ;BR IF NOT
025032  000177  154066                  JMP     @$LPERR         ;START AT THE LOOP ADDRESS
025036  005037  001222          1$:     CLR     $ESCAPE         ;CLEAR ERROR ESCAPE FLAG
025042  032766  072006  000002          BIT     #BIT14!BIT13!BIT12!BIT10!BIT02!BIT01,2(SP)      ;CHECK ERROR TYPE
025050  001402                          BEQ     2$              ;BR IF DRIVE NOT OFFLINE, UNLOADED, OR
                                                                ;PERSISTENT UNSAFE OR FATAL MASSBUS PARITY
025052  000137  015142                  JMP     $EOP            ;TERMINATE DRIVE
025056  012616                  2$:     MOV     (SP)+,(SP)      ;ADJUST RETURN ADDRESS
025060  000207                          RTS     PC

                                        ;THIS ROUTINE FORMS AN INDEX THAT WILL BE USED TO DISPATCH
                                        ;TO THE PROPER ERROR CALL.  THE INDEX IS FORMED BY EXAMINING
                                        ;THE STATUS/ERROR INDICATOR OF THE APPLICABLE DPB.
                                        ;INDEX          STATUS/ERROR
                                        ;-----          ------------
                                        ;  0            BIT08!BIT01
                                        ;  2            BIT02
                                        ;  4            BIT14!BIT04
                                        ;  6            BIT05 & <BIT09!COMMAND = NON-I/O>
                                        ; 10            BIT06 & <BIT09!COMMAND = I/O>
                                        ;CALL
                                        ;       JSR     #DPB+16,-(SP)   ;ADDRESS OF STATUS/ERROR INDICATOR
                                        ;       JSR     PC,ERINDX       ;FORM INDEX
                                        ;       RETURN                  ;INDEX IS ON THE STACK

025062  010046                  ERINDX: MOV     R0,-(SP)        ;SAVE R0
025064  010146                          MOV     R1,-(SP)        ;SAVE R1
025066  016600  000006                  MOV     6(SP),R0        ;GET STATUS/ERROR INDICATOR POINTER
025072  011037  001356                  MOV     (R0),SVSTAT     ;SAVE THE STATUS/ERROR INDICATOR
025076  005001                          CLR     R1              ;START INDEX AT ZERO
025100  032710                          BIT     (PC)+,(R0)      ;FORM INDEX OF 0?
025102  000402                          .WORD   BIT08!BIT01
025104  001037                          BNE     5$              ;YES--BRANCH
025106  032710                          BIT     (PC)+,(R0)      ;FORM PARITY ERROR OR PORT REQUEST INDEX (2)?
025110  000004                          .WORD   BIT02
025112  001033                          BNE     4$              ;YES--BRANCH
025114  032710                          BIT     (PC)+,(R0)      ;FORM UNSAFE INDEX (4)?
025116  040020                          .WORD   BIT14!BIT04
025120  001027                          BNE     3$              ;YES--BRANCH
025122  032710                          BIT     (PC)+,(R0)      ;FORM NON-I/O ERROR INDEX (6)?
025124  000040                          .WORD   BIT05
025126  001023                          BNE     2$              ;YES--BRANCH
025130  032710                          BIT     (PC)+,(R0)      ;FORM I/O ERROR INDEX (10)?
025132  000100                          .WORD   BIT06
025134  001017                          BNE     1$              ;YES--BRANCH
025136  032710                          BIT     (PC)+,(R0)      ;SOFTWARE TIMEOUT?
025140  001000                          .WORD   BIT09
025142  001420                          BEQ     5$              ;NO--FORM INDEX OF 0
025144  122760  000150  177762          CMPB    #150,-16(R0)    ;YES--I/O?
025152  003011                          BGT     2$              ;NO--BRANCH
```

```
                                                TST     RM.REG+RMER1    ;ANY DRIVE ERROR ?
                                                BNE     1$              ;BRANCH IF SO
                                                BIT     #BSE,RM.REG+RMER2       ;BAD SPOT ERROR
                                                BEQ     1$              ;BRANCH IF NOT
                                                INC     R1              ;SKIP , NOT REPORT BAD SPOT ERROR
                                        1$:     INC     R1              ;INDEX=10---ERROR=45 OR 46
                                        2$:     INC     R1              ;INDEX=6---ERROR=44
                                        3$:     INC     R1              ;INDEX=4---ERROR=43
                                        4$:     INC     R1              ;INDEX=2---ERROR=42
                                        5$:     ASL     R1              ;INDEX=0---ERROR=41
                                                MOV     R1,6(SP)        ;RETURN INDEX TO USER
                                                MOV     (SP)+,R1        ;RESTORE R1
                                                MOV     (SP)+,R0        ;RESTORE R0
                                                RTS     PC              ;RETURN FROM CALL

                                        ;THIS ROUTINE WILI PROVIDE A STALL IN MILLISECONDS FOR A SPECIFIC
                                        ;AMOUNT OF TIME I  BIT13 OF C.SWR = 0 OR A RANDOM AMOUNT OF TIME
                                        ;IF BIT 13 OF C.SWR = 1.
                                        ;STALL1 CONTAINS SPECIFIED TIME FOR TESTS 0 - 7, AND STALL2
                                        ;CONTAINS THE TIME FOR TESTS 16-21.
                                        ;CALL
                                        ;       JSR     R0,STALL
                                        ;       TIME POINTER            ;WHERE TO FIND THE STALL TIME

                                        STALL:  MOV     @(R0)+,-(SP)    ;PICKUP STALL TIME
                                                BIT     #SW13,C.SWR     ;USE A RANDOM TIME?
                                                BEQ     1$              ;NO--BRANCH
                                                JSR     PC,$RAND        ;YES--FORM RANDOM NUMBER
                                                MOV     $LONUM,(SP)     ;AND USE IT FOR THE STALL TIME
                                                BIC     #^C77,(SP)      ;BUT NEVER > 64 MILLISECONDS
                                        1$:     CLR     -(SP)           ;CLEAR TEMP. LOCATION
                                        2$:     SUB     #1,2(SP)        ;MORE STALL REQUIRED?
                                                BLO     4$              ;NO--BRANCH
                                                MOV     #100.,(SP)      ;STALL FOR ABOUT 1 MILLISECOND
                                        3$:     TST     R0              ;NOP TO KILL TIME
                                                DEC     0(SP)           ;COUNT
                                                BNE     3$              ;LOOP IF MORE COUNTS NEEDED
                                                BR      2$
                                        4$:     CMP     (SP)+,(SP)+     ;CLEAN OFF THE STACK
                                                RTS     R0              ;EXIT

                                        ;ROUTINE TO SOFTWARE COMPARE HEADER ON IMPLIED SEEKS
                                        ;CALL
                                        ;       JSR     R0,VERIFY
                                        ;       ADR POINTER             ;ADDRESS OF DPB+10 (SECTOR NUMBER)
                                        ;       BR      ???             ;ERROR RETURN
                                        ;       RETURN

                                        VERIFY: MOV     R1,-(SP)        ;SAVE R1
                                                MOV     (R0)+,R1        ;GET ADDRESS OF DPB+10
                                                BIC     #170000,BUFFER  ;STRIP FORMAT AND BAD SECTOR BITS FROM CYLINDER NUMBER
                                                MOVB    (R1),6$         ;SAVE EXPECTED SECTOR ADDRESS
                                                CMP     BUFFER,2(R1)    ;CYLINDER NUMBER OK?
                                                BNE     2$              ;NO--BRANCH
                                                CMPB    BUFFER+3,1(R1)  ;IS THE TRACK ADDRESS GOOD ?
                                                BNE     2$              ;BR IF NO
                                                TST     $SSENB-10(R1)   ;WAS SKIP SECTOR ENABLED ?
```

```
                                        BEQ     1$              ;BR IF NO
                                        INCB    6$              ;INCREMENT EXPECTED SECTOR ADDRESS BY 1
                                1$:     CMPB    BUFFER+2,6$     ;IS THE SECTOR ADDRESS GOOD ?
                                        BEQ     4$              ;BR IF YES
                                2$:     MOV     BUFFER,CYL.RD   ;GET RECIEVED SECTOR ADDRESS
                                        MOVB    BUFFER+3,TRK.RD ;       ''      TRACK   ''
                                        MOVB    BUFFER+2,SEC.RD ;       ''      CYLINDER  ''
                                        MOVB    6$,SEC.DS       ;GET EXPECTED SECTOR ADDRESS
                                        TSTB    (R1)+           ;ADJUST R1 POINTER
                                        MOVB    (R1)+,TRK.DS    ;GET EXPECTED TRACK ADDRESS
                                        MOV     (R1),CYL.DS     ;       ''      CYLINDER  ''
                                        MOV     #3$,$ESCAPE     ;;ESCAPE TO 3$ ON ERROR
                                        EMT     12
                                3$:     MOV     #RECAL,DPB.A+2  ;LOAD RECALIBRATE ORDER CODE
                                        JSR     R0,CALL.A       ;GO EXECUTE THE COMMAND
                                        CLR     $ESCAPE         ;CLEAR ERROR ESCAPE FLAG
                                        BIT     #SW9,@SWR       ;LOOP ON ERROR ?
                                        BEQ     4$              ;BR IF NOT
                                        MOV     $LPERR,R0       ;RETURN TO ERROR LOOP ADDRESS
                                        BR      5$
                                4$:     TST     (R0)+           ;INCREMENT RETURN ADDRESS
                                5$:     MOV     (SP)+,R1        ;RESTORE R1
                                        RTS     R0              ;EXIT

                                6$:     .WORD   0               ;TEMPORARY STORAGE FOR EXPECTED SECTOR ADDRESS

                                ;THIS ROUTINE WILL PERFORM A "MASSBUS" INIT. FOLLOWED BY
                                ;A "RECALIBRATE" ON THE DRIVE UNDER TEST.
                                ;NOTE: THIS ROUTINE DESTROYS R1 AND R4
                                ;CALL
                                ;       JSR     R0,SRCH00       ;DO A MASSBUS INIT. AND RECAL
                                ;       RETURN1                 ;RETURN HERE IF NO ERROR
                                ;       RETURN2                 ;RETURN HERE ON ERROR

                                SRCH00: CLR     R1              ;INCASE OF ERROR (TYPTIM)
                                        CLR     PS
                                        MOV     #ISR,@RMVEC     ;SETUP INTERRUPT VECTOR
                                        MOV     RMADR,R4        ;PICKUP ADDRESS OF RMCS1
                                        MOV     #CLR,RMCS2(R4)  ;MASSBUS INIT.
                                        CLR     DTADPB+10       ;TRACK=0 & SECTOR=0
                                        CLR     DTADPB+12       ;CYLINDER =0
                                        MOV     #RECAL,DTADPB+2 ;COMMAND = RECALIBRATE
                                        CLR     $ESCAPE         ;NO ESCAPE ADDRESS
                                        JSR     R0,RM80         ;CALL THE DRIVER
                                        DTADPB                  ;DPB POINTER
                                        BR      4$              ;QUEUE IS FULL
                                1$:     TST     DTADPB+16       ;WAIT ON DONE
                                        BEQ     1$
                                        BPL     3$              ;TAKE NORMAL EXIT IF NO ERROR
                                        MOV     #2$,$ESCAPE     ;;ESCAPE TO 2$ ON ERROR
                                        MOV     DTADPB+12,CYL.DS        ;CYLINDER
                                        MOVB    DTADPB+11,TRK.DS        ;TRACK
                                        MOVB    DTADPB+10,SEC.DS        ;SECTOR
                                        MOV     #DTADPB+16,-(SP)        ;STATUS/ERROR INDICATOR ADDRESS
                                        JSR     PC,ERINDX       ;FORM DISPATCH INDEX
                                        ADD     (SP)+,PC        ;REPORT PROPER ERROR
                                        EMT     41              ;NON-EXIST DRIVE
```

```
                                                EMT     42              ;PORT TIMEOUT
                                                EMT     43              ;UNSAFE ERROR
                                                EMT     44              ;NON-I/O ERROR
                                                EMT     45              ;I/O ERROR
612 025644  005720                      2$:     TST     (R0)+           ;ADJUST FOR ERROR EXIT
613 025646  000404                              BR      4$              ;GO TO THE EXIT
614 025650  005064  000006              3$:     CLR     RMDA(R4)        ;TRACK AND SECTOR = 0
615 025654  005064  000034                      CLR     RMDC(R4)        ;CYLINDER = 0
616 025660  000200                      4$:     RTS     R0              ;RETURN
617
618                                     ;THIS IS AN RTI WHICH IS USED BY THE TIMING TESTS & THE SERVO SETTLE DOWN TEST
619
620 025662  000002                      DORTI:  RTI                     ;RETURN FROM INTERRUPT
621
622                                     ;THIS ROUTINE WILL INITIALIZE THE TIMERS USED BY THE ''TIMING ROUTINES
623                                     ;CALL
624                                     ;       JSR     PC,STRTMR
625                                     ;       RETURN
626
627 025664  104412                      STRTMR: SAVREG                  ;SAVE R0-R5
628 025666  012700  001376                      MOV     #TIM.UP,R0      ;INITIALIZE TIME TABLES
629 025672  005020                      1$:     CLR     (R0)+           ;CLEAR
630 025674  020027  001432                      CMP     R0,#TIM.PT      ;DONE?
631 025700  103774                              BLO     1$              ;NO--BRANCH
632 025702  012710  043540                      MOV     #BUFFER,(R0)    ;SETUP TIME TABLE POINTER
633 025706  012737  077777  001376              MOV     #^CBIT15,TIM.UP ;SET MINIMUM TIME TO MAXIMUM
634 025714  012737  077777  001414              MOV     #^CBIT15,TIM.DN ;POSITIVE NUMBER
635 025722  104413                              RESREG                  ;RESTORE R0-R5
636 025724  000207                              RTS     PC              ;RETURN
637
638                                     ;THIS ROUTINE WILL ADD THE ELAPSED TIME TO THE AVERAGE COUNTER AND
639                                     ;MAINTAIN THE MINIMUM AND MAXIMUM TIMES.
640                                     ;NOTE: THIS ROUTINE DESTROYS R2
641                                     ;CALL
642                                     ;       MOV     #TP,R3          ;PARAMETER POINTER
643                                     ;       MOV     FLAG,R5         ;FLAG=0=COUNT UP
644                                     ;                               ;FLAG=-1=COUNT DOWN
645                                     ;       JSR     PC,COUNT
646                                     ;       RETURN
647
648 025726  012702  001376              COUNT:  MOV     #TIM.UP,R2      ;PICKUP THE ''UP'' POINTER
649 025732  005705                              TST     R5              ;USE IT?
650 025734  001402                              BEQ     1$              ;YES--BRANCH
651 025736  012702  001414                      MOV     #TIM.DN,R2      ;NO--PICKUP ''DOWN'' POINTER
652 025742  027722  153572              1$:     CMP     @PKC,(R2)+      ;LESS THAN PREVIOUS LOW?
653 025746  002003                              BGE     2$              ;NO--BRANCH
654 025750  017762  153564  177776              MOV     @PKC,-2(R2)     ;YES--SAVE IT
655 025756  027763  153556  000004  2$:         CMP     @PKC,4(R3)      ;LESS THAN THE LOW LIMIT?
656 025764  002001                              BGE     3$              ;NO--BRANCH
657 025766  005212                              INC     (R2)            ;YES--COUNT IT
658 025770  005722                      3$:     TST     (R2)+           ;ADVANCE THE POINTER
659 025772  027722  153542                      CMP     @PKC,(R2)+      ;GREATER THAN PREVIOUS HIGH?
660 025776  003403                              BLE     4$              ;NO--BRANCH
661 026000  017762  153534  177776              MOV     @PKC,-2(R2)     ;YES--SAVE IT
662 026006  027763  153526  000006  4$:         CMP     @PKC,6(R3)      ;GREATER THAN THE HIGH LIMIT?
663 026014  003401                              BLE     5$              ;NO--BRANCH
664 026016  005212                              INC     (R2)            ;YES--COUNT IT
```

```
                                        5$:     TST     (R2)+           ;ADVANCE THE POINTER
                                                ADD     @PKC,(R2)+      ;ADD THIS COUNT TO THE TOTAL
                                                ADC     (R2)+
                                                INC     (R2)            ;COUNT THIS READING
                                                CMP     #BUFFER+<4*561.>,TIM.PT ;SAVE THIS COUNT?
                                                BLOS    6$              ;NO--BRANCH
                                                MOV     @PKC,@TIM.PT    ;YES--WELL SAVE IT THEN
                                                ADD     #2,TIM.PT       ;ADVANCE THE POINTER
                                        6$:     RTS     PC              ;RETURN

                                        ;THIS ROUTINE PRINTS THE SPEC OF ALL TIMING TESTS
                                        ;CALL
                                        ;       JSR     R0,SPTYP
                                        ;       TABLE ADDRESS
                                        ;
                                        ;TABLE: .WORD   ASCIZ MESSAGE POINTER
                                        ;       .WORD   MIN VALUE
                                        ;       .WORD   MAX VALUE

                                        SPTYP:  MOV     (R0)+,R2        ;THE TABLE ADDRESS
                                                BIT     #SW06,@SWR      ;ALLOW PRINT
                                                BNE     3$              ;EXIT IF NOT
                                                TYPE    ,$CRLF
                                                TYPE    ,$CRLF
                                                MOV     (R2)+,1$
                                                TYPE
                                        1$:     .WORD   0
                                                MOV     (R2)+,-(SP)     ;LOAD MIN VALUE
                                                BEQ     2$              ;SKIP IF MIN VALUE IS 0
                                                TYPE    ,MSGMIN
                                                JSR     PC,$SB2D        ;CONVERT TO DECIMAL
                                                JSR     PC,$SUPRS       ;TYPE IT
                                                TYPE    ,MSG0US         ;0 US
                                        2$:     TYPE    ,MSGMAX
                                                MOV     (R2),-(SP)      ;MAXMUM VALUE
                                                JSR     PC,$SB2D
                                                JSR     PC,$SUPRS
                                                TYPE    ,MSG0US
                                                TYPE    ,$CRLF          ;CR-LF
                                        3$:     RTS     R0


                                        ;;THIS ROUTINE IS USED TO TYPE THE MINIMUM,
                                        ;MAXIMUM, AND AVERAGE TIMES FOR THE TIMING TESTS
                                        ;IT WILL ALSO CHECK THE TIMES TO ENSURE
                                        ;THEY ARE WITHIN TOLERANCE AND IF NOT FLAG THE BAD TIMES.
                                        ;CALL
                                        ;       JSR     R0,TYPTIM       ;GO REPORT THE TIMES
                                        ;       TABLE                   ;POINT TO THE PROPER TABLE
                                        ;       RETURN
                                        ;
                                        ;TABLE: MSGADR1                 ;ADDRESS OF ASCIZ MESSAGE NUMBER 1
                                        ;       MSGADR2                 ;ADDRESS OF ASCIZ MESSAGE NUMBER 2
                                        ;       MIN.ALLOWED             ;MINIMUM TIME ALLOWED
                                        ;       MAX.ALLOWED             ;MAXIMUM TIME ALLOWED
```

```
                                        TYPTIM:
    722 026166                                      MOV     R2,-(SP)        ;;PUSH R2 ON STACK
        026166  010246                              MOV     R3,-(SP)        ;;PUSH R3 ON STACK
        026170  010346                              MOV     R4,-(SP)        ;;PUSH R4 ON STACK
        026172  010446                              MOV     R5,-(SP)        ;;PUSH R5 ON STACK
        026174  010546
    723 026176  012002                              MOV     (R0)+,R2        ;PICKUP THE TABLE POINTER
    724 026200  032777  000100  152746              BIT     #SW06,@SWR      ;INHIBIT TIME REPORTS?
    725 026206  001154                              BNE     9$              ;YES--BRANCH
    726 026210  012237  026230                      MOV     (R2)+,2$        ;ADDRESS OF MESSAGE NUMBER 1
    727 026214  012205                              MOV     (R2)+,R5        ;ADDRESS OF MESSAGE NUMBER 2
    728 026216  012203                              MOV     (R2)+,R3        ;PICKUP THE LOW LIMIT
    729 026220  011202                              MOV     (R2),R2         ;AND THE HIGH LIMIT
    730 026222  012704  001376                      MOV     #TIM.UP,R4      ;PARAMETER POINTER
    731 026226  104401                      1$:     TYPE                    ;TYPE THE MESSAGE
    732 026230  000000                      2$:     .WORD   0               ;ASCIZ MESSAGE POINTER GOES HERE
    733 026232  005764  000014                      TST     14(R4)          ;DID ANY COUNTS OCCUR?
    734 026236  001536                              BEQ     8$              ;NO--BRANCH
    735 026240  104401  037674                      TYPE    ,MSGMIN         ;'MIN='
    736 026244  012446                              MOV     (R4)+,-(SP)     ;PUT (R4)+ ON THE STACK
        026246  004737  022212                      JSR     PC,$SB2D        ;CHANGE TO DECIMAL ASCIZ
        026252  004737  022442                      JSR     PC,$SUPRS       ;TYPE WITHOUT LEADING ZEROS
    737 026256  104401  037716                      TYPE    ,MSGOUS         ;'0 US'
    738 026262  005724                              TST     (R4)+           ;ANY SEEKS BELOW THE LOW LIMIT
    739 026264  001421                              BEQ     3$              ;NO--BRANCH
    740 026266  104401  037160                      TYPE    ,COMMA          ;TYPE ', '
    741 026272  016446  177776                      MOV     -2(R4),-(SP)    ;PUT -2(R4) ON THE STACK
        026276  004737  022212                      JSR     PC,$SB2D        ;CHANGE TO DECIMAL ASCIZ
        026302  004737  022442                      JSR     PC,$SUPRS       ;TYPE WITHOUT LEADING ZEROS
    742 026306  104401  037723                      TYPE    ,MBELOW         ;'BELOW THE MINIMUM OF'
    743 026312  010346                              MOV     R3,-(SP)        ;PUT R3 ON THE STACK
        026314  004737  022212                      JSR     PC,$SB2D        ;CHANGE TO DECIMAL ASCIZ
        026320  004737  022442                      JSR     PC,$SUPRS       ;TYPE WITHOUT LEADING ZEROS
    744 026324  104401  037716                      TYPE    ,MSGOUS         ;TYPE '0 US'
    745 026330  104401  037702              3$:     TYPE    ,MSGMAX         ;'MAX='
    746 026334  012446                              MOV     (R4)+,-(SP)     ;PUT (R4)+ ON THE STACK
        026336  004737  022212                      JSR     PC,$SB2D        ;CHANGE TO DECIMAL ASCIZ
        026342  004737  022442                      JSR     PC,$SUPRS       ;TYPE WITHOUT LEADING ZEROS
    747 026346  104401  037716                      TYPE    ,MSGOUS         ;TYPE '0 US'
    748 026352  005724                              TST     (R4)+           ;ANY SEEKS ABOVE THE HIGH LIMIT
    749 026354  001421                              BEQ     4$              ;NO--BRANCH
    750 026356  104401  037160                      TYPE    ,COMMA          ;TYPE ', '
    751 026362  016446  177776                      MOV     -2(R4),-(SP)    ;PUT -2(R4) ON THE STACK
        026366  004737  022212                      JSR     PC,$SB2D        ;CHANGE TO DECIMAL ASCIZ
        026372  004737  022442                      JSR     PC,$SUPRS       ;TYPE WITHOUT LEADING ZEROS
    752 026376  104401  037752                      TYPE    ,MABOVE         ;'ABOVE THE MAXIMUM OF'
    753 026402  010246                              MOV     R2,-(SP)        ;PUT R2 ON THE STACK
        026404  004737  022212                      JSR     PC,$SB2D        ;CHANGE TO DECIMAL ASCIZ
        026410  004737  022442                      JSR     PC,$SUPRS       ;TYPE WITHOUT LEADING ZEROS
    754 026414  104401  037716                      TYPE    ,MSGOUS         ;TYPE '0 US'
    755 026420  104401  037710              4$:     TYPE    ,MSGAVG         ;'AVG='
    756 026424  012446                              MOV     (R4)+,-(SP)     ;FORM THE AVERAGE
    757 026426  012446                              MOV     (R4)+,-(SP)
    758 026430  012446                              MOV     (R4)+,-(SP)
    759 026432  004737  022604                      JSR     PC,$DIV
    760 026436  006126                              ROL     (SP)+           ;IS THE REMAINDER OVER HALF?
    761 026440  100001                              BPL     5$              ;NO--BRANCH
    762 026442  005216                              INC     (SP)            ;YES--ROUND UP
```

```
763 026444                                  5$:
    026444  004737  022212                          JSR     PC,$SB2D        ;CHANGE TO DECIMAL ASCIZ
    026450  004737  022442                          JSR     PC,$SUPRS       ;TYPE WITHOUT LEADING ZEROS
764 026454  104401  037716                          TYPE    ,MSGOUS         ;TYPE "0 US"
765 026460  104401  037160                          TYPE    ,COMMA          ;TYPE ", "
766 026464  016446  177776                          MOV     -2(R4),-(SP)    ;PUT -2(R4) ON THE STACK
    026470  004737  022212                          JSR     PC,$SB2D        ;CHANGE TO DECIMAL ASCIZ
    026474  004737  022442                          JSR     PC,$SUPRS       ;TYPE WITHOUT LEADING ZEROS
767 026500  022737  000011  001116                  CMP     #11,$TSTNM      ;TEST 11 ?
768 026506  001403                                 BEQ     6$              ;BRANCH IF SO
769 026510  104401  040021                          TYPE    ,MSGSEK         ;TYPE "SEEKS TIMED"
770 026514  000402                                 BR      7$              ;TYPE IT
771 026516  104401  040001              6$:         TYPE    ,MSGSCH         ;TYPE "SEARCH TIME"
772
773 026522  010537  026230              7$:         MOV     R5,2$           ;NEXT MESSAGE POINTER
774 026526  001404                                 BEQ     9$              ;IF NONE EXIT
775 026530  005005                                 CLR     R5              ;NO MORE THAN 2
776 026532  000635                                 BR      1$
777 026534  104401  040036              8$:         TYPE    ,MSGNOT         ;TYPE "NOT TIMED"
778 026540                              9$:
    026540  012605                                  MOV     (SP)+,R5        ;;POP STACK INTO R5
    026542  012604                                  MOV     (SP)+,R4        ;;POP STACK INTO R4
    026544  012603                                  MOV     (SP)+,R3        ;;POP STACK INTO R3
    026546  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
779 026550  000200                                  RTS     R0              ;EXIT
780
781                                         ;THIS SUBROUTINE WILL INCREMENT THE TRACK
782                                         ;NUMBER (R2) BY THE AMOUNT SPECIFIED BY 'IT'.
783                                         ;CALL
784                                         ;       JSR     R0,INCTRK
785                                         ;       RETURN1                 ;TRACK NUMBER GREATER THAN LT15
786                                         ;       RETURN2                 ;TRACK NUMBER INCREMENTED
787
788 026552  020237  002334          INCTRK: CMP     R2,LT           ;LAST TRACK COMPLETED?
789 026556  001410                          BEQ     2$              ;YES--EXIT
790 026560  063702  002336                  ADD     IT,R2           ;NO--UPDATE TRACK
791 026564  020237  002334                  CMP     R2,LT           ;TRACK TO BIG?
792 026570  003402                          BLE     1$              ;NO--EXIT
793 026572  013702  002334                  MOV     LT,R2           ;YES--SET TRACK TO LAST TRACK
794 026576  005720                  1$:     TST     (R0)+           ;ADJUST FOR RETURN 2
795 026600  000200                  2$:     RTS     R0              ;RETURN
796
797
798                                         ;THIS SUBROUTINE WILL INCREMENT THE CYLINDER
799                                         ;NUMBER (R1) BY THE AMOUNT SPECIFIED BY 'IC'.
800                                         ;CALL
801                                         ;       JSR     R0,INCCYL
802                                         ;       RETURN1                 ;CYLINDER NUMBER GREATER THAN LC15
803                                         ;       RETURN2                 ;CYLINDER NUMBER INCREMENTED
804
805 026602  020137  002326          INCCYL: CMP     R1,LC           ;LAST CYLINDER COMPLETED?
806 026606  001410                          BEQ     2$              ;YES--EXIT
807 026610  063701  002330                  ADD     IC,R1           ;NO--UPDATE CYLINDER
808 026614  020137  002326                  CMP     R1,LC           ;CYLINDER TO BIG?
809 026620  003402                          BLE     1$              ;NO--EXIT
810 026622  013701  002326                  MOV     LC,R1           ;YES--SET CYLINDER TO LAST CYLINDER
811 026626  005720                  1$:     TST     (R0)+           ;ADJUST FOR RETURN 2
```

```
  812 026630  000200                  2$:     RTS     R0              ;RETURN
  813
  814                                  ;THIS ROUTINE DECREMENTS THE SECTOR ADDRESS.
  815                                  ;CALL
  816                                  ;       CLR     -(SP)           ;CLEAR THE STACK
  817                                  ;       JSR     PC,DECSEC       ;SUBROUTINE ENTRY
  818                                  ;       RETURN
  819
  820 026632  113766  036646  000002  DECSEC: MOVB    RM.REG+RMDA,2(SP)       ;PUT THE SECTOR ADDRESS ON THE STACK
  821 026640  005366  000002                  DEC     2(SP)           ;DECREMENT THE ADDRESS
  822 026644  100003                          BPL     1$              ;BR IF NOT CORRECTION NEEDED
  823 026646  013766  002426  000002          MOV     PRMLMT+16,2(SP) ;OVERFLOW OCCURED, FORCE TO MAXIMUM ADDRESS
  824 026654  000207                  1$:     RTS     PC              ;RETURN
  825
 1142
 1143                                  ;THIS ROUTINE IS USED TO INPUT THE "CONTROL SWITCHES".
 1144                                  ;IF SWR<07>=1 THE PRESENT SETTING WILL BE TYPED AND THE NEW
 1145                                  ;SETTING IS READ AND STORED.
 1146                                  ;NOTE: THIS ROUTINE DESTROYS R3 AND R4
 1147                                  ;CALL
 1148                                  ;       JSR     PC,GETSWR
 1149                                  ;       RETURN                  ;(C.SWR)=DESIRED CONTROL SWITCHES
 1150
 1151 026656  032777  000200  152270  GETSWR: BIT     #SW07,@SWR      ;READ CONTROL SWITCHES?
 1152 026664  001430                          BEQ     20$             ;NO--BRANCH
 1153 026666  104401  026674                  TYPE    ,65$            ;;TYPE ASCIZ STRING
      026672  000410                          BR      64$             ;;GET OVER THE ASCIZ
                                       ;;65$:  .ASCIZ  <CRLF>/SET SWR<07>=0/
      026714                           64$:
 1154 026714  012703  036750          10$:    MOV     #MSG.CS,R3      ;"CONTROL SWITCHES="
 1155 026720  013704  001314                  MOV     C.SWR,R4        ;PRESENT CONTROL SWITCH SETTINGS
 1156 026724  004037  026750                  JSR     R0,GETNUM       ;GET THE NEW SWITCH SETTINGS
 1157 026730  000771                          BR      10$             ; COMMA
 1158 026732  000240                          NOP                     ;PERIOD
 1159 026734  013737  001314  001320          MOV     C.SWR,SAVCSW    ;SAVE PREVIOUS VALUE
 1160 026742  010437  001314                  MOV     R4,C.SWR        ; DOUBLE PERIOD-SAVE NEW SWITCH SETTING
 1161 026746  000207                  20$:    RTS     PC              ;RETURN FROM CALL
 1162
 1163                                  ;THIS ROUTINE WILL TYPE AN ASCIZ MESSAGE AND THEN
 1164                                  ;INPUT AN ASCIZ STRING AND CHANGE THE STRING TO OCTAL
 1165                                  ;IF REQUIRED.
 1166                                  ;NOTE: THIS ROUTINE DESTROYS R1
 1167                                  ;CALL
 1168                                  ;       MOV     #ADR,R3         ;ADDRESS OF ASCIZ MESSAGE
 1169                                  ;       MOV     #NUM,R4         ;OCTAL NUMBER
 1170                                  ;       JSR     R0,GETNUM
 1171                                  ;       RETURN1                 ;INPUT TERMINATED WITH A COMMA
 1172                                  ;       RETURN2                 ;WITH A PERIOD
 1173                                  ;       RETURN3                 ;WITH A DOUBLE PERIOD
 1174                                  ;                               ;R4=INPUT NUMBER AND
 1175                                  ;                               ;R2=R4*32 FOR ALL
 1176                                  ;                               ;THREE RETURNS
 1177
 1178 026750  010337  026756          GETNUM: MOV     R3,2$           ;SAVE MESSAGE POINTER
 1179 026754  104401                  1$:     TYPE                    ;TYPE THE MESSAGE
 1180 026756  000000                  2$:     .WORD   0               ;MESSAGE POINTER GOES HERE
 1181 026760  010446                          MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
```

```
        026762  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1182    026764  104401  036773          TYPE    ,SLASH          ;' / '
1183    026770  104411                  RDLIN                   ;READ AN ASCIZ STRING
1184    026772  012601                  MOV     (SP)+,R1        ;ADDRESS OF FIRST CHARACTER
1185    026774  004037  030644          JSR     R0,CK.CHR       ;CHECK ONE CHARACTER
        027000  026754                  1$                      ;ILLEGAL CHARACTER
        027002  026754                  1$                      ;CARRIAGE RETURN
        027004  027016                  4$                      ;"/"
        027006  027042                  8$                      ;"."
        027010  027050                  9$                      ;","
        027012  027014                  3$                      ;DIGIT 0-9
1186    027014  005301          3$:     DEC     R1              ;DECREMENT THE INPUT POINTER
1187    027016                  4$:
        027016  004037  031104          JSR     R0,CK.NUM       ;CHECK THE NUMBER
        027022  026754                  1$                      ;ILLEGAL INPUT
        027024  027036                  7$                      ;TERMINATED WITH A "," OR "CR"
        027026  027034                  6$                      ;TERMINATED WITH A "."
        027030  027032                  5$                      ;TERMINATED WITH A ".."
1188    027032  005720          5$:     TST     (R0)+           ;DOUBLE PERIOD
1189    027034  005720          6$:     TST     (R0)+           ;SINGLE PERIOD
1190    027036  010204          7$:     MOV     R2,R4           ;COMMA--SAVE INPUT NUMBER
1191    027040  000414                  BR      11$             ;GO TO EXIT
1192    027042  105711          8$:     TSTB    (R1)            ;TERMINATOR AFTER A COMMA?
1193    027044  001343                  BNE     1$              ;NO--LOOP
1194    027046  000411                  BR      11$             ;YES--EXIT
1195    027050  105711          9$:     TSTB    (R1)            ;TERMINATOR AFTER A PERIOD?
1196    027052  001406                  BEQ     10$             ;YES--EXIT
1197    027054  122721  000056          CMPB    #'.,(R1)+       ;NO--DOUBLE PERIOD?
1198    027060  001335                  BNE     1$              ;NO--LOOP
1199    027062  105711                  TSTB    (R1)            ;YES--TERMINATOR?
1200    027064  001333                  BNE     1$              ;NO--LOOP
1201    027066  005720                  TST     (R0)+           ;DOUBLE PERIOD
1202    027070  005720          10$:    TST     (R0)+           ;PERIOD
1203    027072  010402          11$:    MOV     R4,R2           ;COMMA--POSITION THE
1204    027074  000302                  SWAB    R2              ;NUMBER IN CASE IT
1205    027076  006202                  ASR     R2              ;IS THE PRIORITY LEVEL
1206    027100  006202                  ASR     R2
1207    027102  006202                  ASR     R2
1208    027104  000200                  RTS     R0              ;EXIT
1209
1210                            ;THIS ROUTINE IS USED TO CHANGE OR MODIFY
1211                            ;THE TEST PARAMETERS.  IT GIVES THE OPERATOR
1212                            ;THE CAPABILITY OF SPECIFYING WHICH DRIVES TO TEST, WHICH
1213                            ;TESTS TO RUN AND HOW MANY TIMES TO REPEAT EACH TEST
1214
1215    027106  104412          GT.PRM: SAVREG                  ;SAVE R0 - R5
1216    027110  005037  001330  GT.PR1: CLR     DRVSEL          ;NO DRIVE SELECTED
1217    027114  104401  027122          TYPE    ,65$            ;;TYPE ASCIZ STRING
        027120  000406                  BR      64$             ;;GET OVER THE ASCIZ
        027136                  ;;65$:  .ASCIZ  <CRLF>/DRIVE(S)=/
                                64$:
1218    027136  104411                  RDLIN                   ;READ TTY
1219    027140  012601                  MOV     (SP)+,R1        ;ADDRESS OF ASCIZ STRING
1220    027142  004037  030644          JSR     R0,CK.CHR       ;CHECK ONE CHARACTER
        027146  027110                  GT.PR1                  ;ILLEGAL CHARACTER
        027150  027110                  GT.PR1                  ;CARRIAGE RETURN
        027152  027110                  GT.PR1                  ;"/"
```

```
                027154  027110                          GT.PR1                  ;''.''
                027156  027110                          GT.PR1                  ;''.''
                027160  027162                          1$                      ;DIGIT 0-9
     1221       027162  00530'                  1$:     DEC     R1
     1222       027164                          2$:
                027164  012702  000007                  MOV     #7,R2           ;UPPER LIMIT OF INPUT
                027170  004037  030720                  JSR     R0,CK.DIG       ;CHECK THE DIGIT(S)
                027174  027110                          GT.PR1                  ;ILLEGAL INPUT
                027176  027110                          GT.PR1                  ;INPUT TO LARGE
                027200  027206                          3$                      ;TERMINATED WITH A ''.'' OR ''CR''
                027202  027232                          4$                      ;TERMINATED WITH A ''.''
                027204  027232                          4$                      ;TERMINATED WITH A ''..''
     1223       027206  156237  031356  001330  3$:     BISB    ATABIT(R2),DRVSEL       ;SET THE DRIVE SELECTED BIT
     1224       027214  105741                          TSTB    -(R1)           ;WAS THE LINE TERMINATED?
     1225       027216  001362                          BNE     2$              ;NO-GET THE NEXT DRIVE
     1226       027220  005037  001332                  CLR     TSTNMS          ;DESELECT ALL TESTS
     1227       027224  005037  001334                  CLR     TSTNMS+2
     1228       027230  000405                          BR      GTTST1          ;YES--SELECT TEST
     1229       027232  156237  031356  001330  4$:     BISB    ATABIT(R2),DRVSEL       ;SET THE SELECTED DRIVE BITS
     1230       027240  104413                  GT.PR2: RESREG                  ;RESTORE R0 - R5
     1231       027242  000207                          RTS     PC              ;EXIT
     1232
     1233       027244                          GTTST1:
                027244  104401  027252                  TYPE    ,65$            ;;TYPE ASCIZ STRING
                027250  000403                          BR      64$             ;;GET OVER THE ASCIZ
                                                ;;65$:  .ASCIZ  /TEST=/
                027260                          64$:
     1234       027260  104411                          RDLIN                   ;READ AN ASCIZ STRING
     1235       027262  012601                          MOV     (SP)+,R1        ;POINTER TO R1
     1236       027264  122711  000056                  CMPB    #'.,(R1)        ;DOUBLE PERIOD?
     1237       027270  001007                          BNE     1$              ;NO--BRANCH
     1238       027272  122761  000056  000001          CMPB    #'.,1(R1)
     1239       027300  001003                          BNE     1$
     1240       027302  105761  000002                  TSTB    2(R1)           ;''CR''?
     1241       027306  001754                          BEQ     GT.PR2          ;YES--EXIT
     1242       027310  005037  001332          1$:     CLR     TSTNMS          ;NO TEST SELECTED
     1243       027314  005037  001334                  CLR     TSTNMS+2
     1244       027320  005037  001336                  CLR     OPNFLG          ;NO TESTS TO BE OPENED
     1245       027324  005037  001340                  CLR     OPNFLG+2
     1246
     1247       027330  121127  000123          GTTST2: CMPB    (R1),#'S        ;ALL SEEK TESTS?
     1248       027334  001004                          BNE     1$              ;NO--BRANCH
     1249       027336  052737  000377  001332          BIS     #377,TST'IMS    ;YES--SELECT TESTS 0-7
     1250       027344  000525                          BR      GTTST3
     1251
     1252       027346  121127  000124          1$:     CMPB    (R1),#'T        ;ALL TIMING TESTS?
     1253       027352  001004                          BNE     2$              ;NO--BRANCH
     1254       027354  052737  013000  001332          BIS     #13000,TSTNMS   ;YES--SELECT TESTS 11,12 & 14
     1255       027362  000516                          BR      GTTST3
     1256
     1257       027364  004037  030570          2$:     JSR     R0,CK.OCT       ;OCTAL DIGIT?
     1258       027370  000514                          BR      GTTST4          ;NO--BRANCH
     1259       027372  010205                          MOV     R2,R5           ;YES--SAVE IT
     1260       027374  005201                          INC     R1              ;MOVE TO NEXT CHARACTER
     1261       027376  004037  030570                  JSR     R0.CK.OCT       ;OCTAL DIGIT
     1262       027402  000405                          BR      3$              ;NO--BRANCH
     1263       027404  005201                          INC     R1              ;MOVE TO NEXT CHARACTER
```

```
1264 027406  006305                        ASL     R5              ;SCALE HIGH DIGIT
1265 027410  006305                        ASL     R5
1266 027412  006305                        ASL     R5
1267 027414  060502                        ADD     R5,R2           ;COMBINE HIGH & LOW DIGITS
1268 027416                          3$:
1271 027416  020227  000015                CMP     R2,#15          ;VALID TEST NUMBER?
1272 027422  003310                        BGT     GTTST1          ;NO--BRANCH
1273 027424  010237  027616                MOV     R2,6$           ;SAVE THE TEST NUMBER
1274 027430  010204                        MOV     R2,R4           ;CONVERT TEST NUMBER INTO AN INDEX
1275 027432  042704  000017                BIC     #17,R4          ;CLEAR UNWANTED BITS
1276 027436  006204                        ASR     R4              ;SHIFT THE BITS
1277 027440  006204                        ASR     R4
1278 027442  006204                        ASR     R4
1279 027444  006302                        ASL     R2
1280 027446  056264  001560  001332        BIS     BITS(R2),TSTNMS(R4)     ;SELECT TEST
1281 027454  121127  000055                CMPB    (R1),#'-        ;TEST STRING?
1282 027460  001060                        BNE     GTTST4          ;NO--BRANCH
1283 027462  005201                        INC     R1              ;YES--MOVE TO NEXT CHARACTER
1284 027464  004037  030570                JSR     R0,CK.OCT       ;OCTAL DIGIT?
1285 027470  000665                        BR      GTTST1          ;NO--BRANCH
1286 027472  010205                        MOV     R2,R5           ;YES--SAVE IT
1287 027474  005201                        INC     R1              ;MOVE TO NEXT CHARACTER
1288 027476  004037  030570                JSR     R0,CK.OCT       ;OCTAL DIGIT?
1289 027502  000405                        BR      4$              ;NO--BRANCH
1290 027504  005201                        INC     R1              ;YES--MOVE TO NEXT CHARACTER
1291 027506  006305                        ASL     R5              ;SCALE HIGH DIGIT
1292 027510  006305                        ASL     R5
1293 027512  006305                        ASL     R5
1294 027514  060502                        ADD     R5,R2           ;COMBINE HIGH & LOW DIGIT
1295 027516                          4$:
1298 027516  020227  000015                CMP     R2,#15          ;VALID TEST NUMBER?
1299 027522  003250                        BGT     GTTST1          ;NO--BRANCH
1300 027524  023702  027616                CMP     6$,R2           ;IS THE FIRST NUMBER OF THE
1301                                                               ;STRING SMALLER THAN THE LAST?
1302 027530  002245                        BGE     GTTST1          ;NO--BRANCH
1303 027532  010246                        MOV     R2,-(SP)        ;SAVE ENDING TEST NUMBER
1304 027534  013702  027616                MOV     6$,R2           ;GET STARTING TEST NUMBER
1305 027540  012637  027616                MOV     (SP)+,6$        ;STORE ENDING TEST NUMBER
1306 027544  006337  027616                ASL     6$              ;SHIFT ENDING TEST NUMBER
1307 027550  006302                        ASL     R2              ;SHIFT TEST NUMBER
1308 027552  010204                  5$:   MOV     R2,R4           ;COPY TEST NUMBER INTO R4
1309 027554  042704  000037                BIC     #37,R4          ;CLEAR LOWER BITS
1310 027560  006204                        ASR     R4              ;SHIFT THE TEST NUMBER
1311 027562  006204                        ASR     R4
1312 027564  006204                        ASR     R4
1313 027566  006204                        ASR     R4
1314 027570  056264  001560  001332        BIS     BITS(R2),TSTNMS(R4)     ;SELECT THE TEST
1315 027576  062702  000002                ADD     #2,R2           ;INECREMENT THE TEST NUMBER
1316 027602  020237  027616                CMP     R2,6$           ;SEE IF FINISHED
1317 027606  101761                        BLOS    5$              ;BR IF NOT
1318 027610  162702  000002                SUB     #2,R2           ;CORRECT TEST NUMBER
1319 027614  000402                        BR      GTTST4          ;CONTINUE
1320 027616  000000                  6$:   .WORD   0               ;STORE TEST NUMBER HERE
1321
1322 027620  005201                GTTST3: INC     R1              ;MOVE TO NEXT CHARACTER
1323 027622  121127  000056        GTTST4: CMPB    (R1),#'.        ;'PERIOD'?
1324 027626  001461                        BEQ     GTTST5          ;YES--BRANCH
```

```
1325 027630  005737  001332                TST     TSTNMS          ;ANY TEST SELECTED THIS CYCLE?
1326 027634  001005                       BNE     1$              ;BR IF YES
1327 027636  005737  001334                TST     TSTNMS+2        ;ANY TEST SELECTED THIS CYCLE ?
1328 027642  001002                       BNE     1$              ;BR IF YES
1329 027644  000137  027244                JMP     GTTST1          ;NO
1330
1331                                ;CHECK TO OPEN TEST FOR PARAMETER CHANGE
1332
1333 027650  121127  000057          1$:   CMPB    (R1),#'/        ;''OPEN''?
1334 027654  001024                       BNE     4$              ;NO--BRANCH
1335 027656  126127  177777  000123        CMPB    -1(R1),#'S      ;ALL SEEK TESTS?
1336 027664  001004                       BNE     2$              ;NO--BRANCH
1337 027666  052737  000377  001336        BIS     #377,OPNFLG     ;YES--OPEN TESTS 0-7
1338 027674  000421                       BR      5$
1339 027676  126127  177777  000124  2$:   CMPB    -1(R1),#'T      ;ALL TIMING TESTS?
1340 027704  001004                       BNE     3$              ;NO--BRANCH
1341 027706  052737  013000  001336        BIS     #13000,OPNFLG   ;YES--OPEN TESTS 11,12 & 14
1342 027714  000411                       BR      5$
1343 027716  056264  001560  001336  3$:   BIS     BITS(R2),OPNFLG(R4)     ;YES--SET BITS FOR TEST TO OPEN
1344 027724  000405                       BR      5$
1345
1346 027726  121127  000054          4$:   CMPB    (R1),#',        ;''COMMA''?
1347 027732  001402                       BEQ     5$              ;BR IF YES
1348 027734  000137  027244                JMP     GTTST1          ;NO
1349 027740  005201                  5$:   INC     R1              ;MOVE TO NEXT CHARACTER
1350 027742  105711                       TSTB    (R1)            ;''CR''?
1351 027744  001402                       BEQ     6$              ;BR IF 'CR'
1352 027746  000137  027330                JMP     GTTST2          ;NO--GO GET NEXT CHARACTER
1353 027752  005737  001336          6$:   TST     OPNFLG          ;ANY TESTS TO OPEN ?
1354 027756  001042                       BNE     OPNTST          ;BR IF YES
1355 027760  005737  001340                TST     OPNFLG+2        ;ANY TESTS TO OPEN ?
1356 027764  001037                       BNE     OPNTST          ;BR IF YES
1357 027766  000137  027244                JMP     GTTST1          ;NO--START AGAIN
1358
1359 027772  005201                GTTST5: INC     R1              ;MOVE TO NEXT CHARACTER
1360 027774  121127  000056                CMPB    (R1),#'.        ;''PERIOD''?
1361 030000  001414                       BEQ     GTTST6          ;YES--BRANCH
1362 030002  105711                       TSTB    (R1)            ;''CR''?
1363 030004  001402                       BEQ     1$              ;YES--BRANCH
1364 030006  000137  027244                JMP     GTTST1
1365 030012  005737  001336          1$:   TST     OPNFLG          ;ANY TESTS TO OPEN ?
1366 030016  001022                       BNE     OPNTST          ;BR IF YES
1367 030020  005737  001340                TST     OPNFLG+2        ;ANY TESTS TO OPEN ?
1368 030024  001017                       BNE     OPNTST          ;BR IF YES
1369 030026  000137  027240                JMP     GT.PR2          ;NO--GO START TESTING
1370
1371 030032  005201                GTTST6: INC     R1              ;MOVE TO NEXT CHARACTER
1372 030034  105711                       TSTB    (R1)            ;''CR''?
1373 030036  001402                       BEQ     1$              ;YES--BRANCH
1374 030040  000137  027244                JMP     GTTST1          ;NO--GO ASK FOR TEST
1375 030044  005737  001336          1$:   TST     OPNFLG          ;ANY TESTS TO OPEN ?
1376 030050  001005                       BNE     OPNTST          ;BR IF YES
1377 030052  005737  001340                TST     OPNFLG+2        ;ANY TESTS TO OPEN ?
1378 030056  001002                       BNE     OPNTST          ;BR IF YES
1379 030060  000137  027240                JMP     GT.PR2          ;NO--GO START TESTING
1380
1381                                ;OPEN THE SELECTED TEST FOR CHANGES
```

```
1382
1383 030064  104412                  OPNTST: SAVREG                  ;SAVE R0 - R5
1384 030066  005027                          CLR     (PC)+            ;START WITH TEST 0
1385 030070  000000                  OPN.CT: .WORD   0                ;COUNT STORED HERE
1386 030072  000411                          BR      OPN.2            ;SKIP THE INCREMENT
1387
1388 030074  005237  030070          OPN.1:  INC     OPN.CT           ;MOVE TO THE NEXT TEST
1391 030100  022737  000015  030070          CMP     #15,OPN.CT       ;TEST NUMBER TOO BIG?
1392 030106  002003                          BGE     OPN.2            ;NO--OPEN THE NEXT TEST
1393 030110  104413                          RESREG                   ;RESTORE R0 - R5
1394 030112  000137  027244                  JMP     GTTST1           ;YES--GO ASK FOR MORE TESTS
1395
1396 030116  013705  030070          OPN.2:  MOV     OPN.CT,R5        ;SETUP TO USE THE
1397 030122  006305                          ASL     R5               ;TEST NUMBER AS AN INDEX
1398 030124  013703  030070                  MOV     OPN.CT,R3        ;GET INDEX
1399 030130  042703  000017                  BIC     #17,R3           ;CLEAR LOWER TEST BITS
1400 030134  006203                          ASR     R3               ;SHIFT TEST NUMBER
1401 030136  006203                          ASR     R3
1402 030140  006203                          ASR     R3
1403 030142  036563  001560  001336          BIT     BITS(R5),OPNFLG(R3)     ;OPEN THIS TEST?
1404 030150  001751                          BEQ     OPN.1            ;NO--MOVE TO NEXT TEST
1405 030152  104401  030160                  TYPE    ,65$             ;;TYPE ASCIZ STRING
     030156  000404                          BR      64$              ;;GET OVER THE ASCIZ
                                     ;;65$:  .ASCIZ  / TEST /
     030170                          64$:
1406 030170  013746  030070                  MOV     OPN.CT,-(SP)     ;;SAVE OPN.CT FOR TYPEOUT
     030174  104403                          TYPOS                    ;;GO TYPE--OCTAL ASCII
     030176     002                          .BYTE   2                ;;TYPE 2 DIGIT(S)
     030177     000                          .BYTE   0                ;;SUPPRESS LEADING ZEROS
1407 030200  104401  001231                  TYPE    ,$CRLF           ;CR-LF
1408 030204  016500  002352                  MOV     PRMPT(R5),R0     ;PICKUP PARAMETER POINTER
1409 030210  011046                          MOV     (R0),-(SP)       ;SAVE THE VARIABLE INDICATOR
1410 030212  012702  002320                  MOV     #PRM,R2          ;FIRST ADDRESS OF TABLE
1411 030216  000405                          BR      2$
1412 030220  006216                  1$:     ASR     (SP)             ;CHECK FOR A VARIABLE
1413 030222  103403                          BCS     2$               ;GO MOVE THIS ONE
1414 030224  001404                          BEQ     OPNPRM           ;DONE
1415 030226  005722                          TST     (R2)+            ;BUMP THE POINTER
1416 030230  000773                          BR      1$
1417 030232  012022                  2$:     MOV     (R0)+,(R2)+      ;MOVE THIS VARIABLE INTO THE
1418 030234  000771                          BR      1$               ;COMMON AREA
1419
1420 030236  013716  002320          OPNPRM: MOV     PRM,(SP)         ;GET THE VARIABLE INDICATOR
1421 030242  005004                          CLR     R4               ;ZERO THE INDEX
1422 030244  006216                  1$:     ASR     (SP)             ;CHECK FOR A VARIABLE
1423 030246  103403                          BCS     3$               ;GO GET IT
1424 030250  001772                          BEQ     OPNPRM           ;OUT OF VARIABLES
1425 030252  005724                  2$:     TST     (R4)+            ;UPDATE THE INDEX
1426 030254  000773                          BR      1$
1427 030256                          3$:
1432 030256  104401  040053                  TYPE    ,BLNKS2          ;TYPE 2 BLANKS
1433 030262  016437  002434  030272          MOV     PRMMSG(R4),4$    ;TYPE THE NAME OF THIS VARIABLE
1434 030270  104401                          TYPE
1435 030272  000000                  4$:     .WORD   0
1436 030274  104401  036746                  TYPE    ,MSG.EQ          ;TYPE "="
1437 030300  016446  002322                  MOV     RPT(R4),-(SP)    ;PUT RPT(R4) ON THE STACK
     030304  004737  022212                  JSR     PC,$SB2D         ;CHANGE TO DECIMAL ASCIZ
```

```
                030310  004737  022442                  JSR     PC,$SUPRS       ;TYPE WITHOUT LEADING ZEROS
1438            030314  104401  036773                  TYPE    ,SLASH          ;TYPE ' / '
1439            030320  104411                          RDLIN
1440            030322  012601                          MOV     (SP)+,R1        ;READ AN ASCIZ STRING
1441            030324  004037  030644                  JSR     R0,CK.CHR       ;CHECK ONE CHARACTER
                030330  030256                          3$                      ;ILLEGAL CHARACTER
                030332  030252                          2$                      ;CARRIAGE RETURN
                030334  030402                          9$                      ;"/"
                030336  030344                          5$                      ;"."
                030340  030352                          6$                      ;","
                030342  030400                          8$                      ;DIGIT 0-9
1442            030344  105711          5$:     TSTB    (R1)                    ;"CR"?
1443            030346  001343                  BNE     3$                      ;NO--STAY ON THIS VARIABLE
1444            030350  000740                  BR      2$                      ;YES--MOVE TO NEXT VARIABLE
1445            030352  105711          6$:     TSTB    (R1)                    ;IS THERE A "CR" AFTER THE PERIOD?
1446            030354  001002                  BNE     7$                      ;NO
1447            030356  000137  030436          JMP     OPN.N2                  ;YES--GO CLOSE THIS TEST
1448            030362  122721  000056  7$:     CMPB    #'.,(R1)+               ;DOUBLE PERIOD?
1449            030366  001333                  BNE     3$                      ;NO--GO ASK FOR THIS VARIABLE
1450            030370  105711                  TSTB    (R1)                    ;YES--IS A "CR" AFTER THE DOUBLE PERIOD?
1451            030372  001331                  BNE     3$                      ;NO--ASK FOR THIS VARIABLE AGAIN
1452            030374  000137  030454          JMP     OPN.X2                  ;YES--CLOSE ALL TEST
1453
1454            030400  005301          8$:     DEC     R1                      ;BACK THE POINTER UP BY ONE
1455            030402                  9$:
                030402  016402  002410          MOV     PRMLMT(R4),R2                   ;UPPER LIMIT OF INPUT
                030406  004037  030720          JSR     R0,CK.DIG               ;CHECK THE DIGIT(S)
                030412  030256                  3$                              ;ILLEGAL INPUT
                030414  030256                  3$                              ;INPUT TO LARGE
                030416  030424                  10$                             ;TERMINATED WITH A "," OR "CR"
                030420  030432                  OPN.N1                          ;TERMINATED WITH A "."
                030422  030450                  OPN.X1                          ;TERMINATED WITH A ".."
1456            030424  010264  002322  10$:     MOV     R2,RPT(R4)              ;SAVE THIS VARIABLE
1457            030430  000710                  BR      2$                      ;MOVE TO NEXT VARIABLE
1458
1530            030432  010264  002322  OPN.N1: MOV     R2,RPT(R4)              ;SAVE THIS VARIABLE
1531            030436  005726          OPN.N2: TST     (SP)+                   ;CLEAN OFF THE STACK
1532            030440  004737  030510          JSR     PC,CLOSE                ;CLOSE THIS TEST
1533            030444  000137  030074          JMP     OPN.1                   ;GO OPEN THE NEXT TEST
1534
1535            030450  010264  002322  OPN.X1: MOV     R2,RPT(R4)              ;SAVE THIS VARIABLE
1536            030454  005726          OPN.X2: TST     (SP)+                   ;CLEAN OFF THE STACK
1537            030456  004737  030510  1$:     JSR     PC,CLOSE                ;CLOSE THIS TEST
1538            030462  005725          2$:     TST     (R5)+                   ;UPDATE THE INDEX
1539            030464  020527  000034          CMP     R5,#16*2                ;INDEX TO BIG?
1540            030470  002403                  BLT     3$                      ;NO--BRANCH
1541            030472  104413                  RESREG                          ;RESTORE R0 - R5
1542            030474  000137  027240          JMP     GT.PR2                  ;GO TO EXIT
1543            030500  036503  001560  3$:     BIT     BITS(R5),R3             ;IS THIS TEST OPEN FOR CHANGE?
1544            030504  001364                  BNE     1$                      ;YES--GO CLOSE IT
1545            030506  000765                  BR      2$                      ;NO--MOVE TO NEXT TEST
1546
1547                                    ;CLOSE CURRENT TEST THAT WAS OPEN FOR CHANGES
1548
1549            030510  104412  CLOSE:  SAVREG                          ;SAVE R0 - R5
1550            030512  012700  002320          MOV     #PRM,R0                 ;"FROM" ADDRESS
1551            030516  016501  002352          MOV     PRMPT(R5),R1            ;"TO" ADDRESS
```

DOUBLE PRECISION DIVISION SUBROUTINE

```
1552 030522  012002                        MOV     (R0)+,R2        ;"FROM" INDICATOR
1553 030524  012103                        MOV     (R1)+,R3        ;"TO" INDICATOR
1554 030526  012704  000001                MOV     #1,R4           ;TEST BIT START A "RPT"
1555 030532  030402                1$:     BIT     R4,R2           ;PARAMETER TO BE MOVED?
1556 030534  001403                        BEQ     2$              ;NO--BRANCH
1557 030536  030403                        BIT     R4,R3           ;A PLACE TO PUT IT?
1558 030540  001404                        BEQ     3$              ;NO--BRANCH
1559 030542  011011                        MOV     (R0),(R1)       ;YES--MOVE "FROM" TO "TO"
1560 030544  030403                2$:     BIT     R4,R3           ;"TO" PARAMETER?
1561 030546  001401                        BEQ     3$              ;NO--BRANCH
1562 030550  005721                        TST     (R1)+           ;YES--UPDATE THE POINTER
1563 030552  005720                3$:     TST     (R0)+           ;UPDATE FROM POINTER
1564 030554  006304                        ASL     R4              ;POSITION THE TEST BIT
1565 030556  032704  002000                BIT     #BIT10,R4       ;DONE?
1566 030562  001763                        BEQ     1$              ;NO--BRANCH
1567 030564  104413                        RESREG                  ;RESTORE R0 - R5
1568 030566  000207                        RTS     PC              ;RETURN
1569
1570                                ;THIS ROUTINE IS USED TO CHECK IF AN
1571                                ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
1572                                ;CALL
1573                                ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
1574                                ;       JSR     R0,CK.OCT       ;CHECK THE CHARACTER
1575                                ;       RETURN1                 ;CHARACTER IS NOT BETWEEN 0-7
1576                                ;       RETURN2                 ;CHARACTER IS IN R2 AS A
1577                                ;                               ;OCTAL DIGIT
1578
1579 030570  121127  000060        CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
1580 030574  103407                        BLO     1$              ;YES -- BRANCH
1581 030576  121127  000067                CMPB    (R1),#'7        ;GREATER THAN SEVEN?
1582 030602  101004                        BHI     1$              ;YES -- BRANCH
1583 030604  111102                        MOVB    (R1),R2         ;GET THE CHARACTER
1584 030606  042702  177770                BIC     #^C7,R2         ;STRIP AWAY THE ASCII
1585 030612  005720                        TST     (R0)+           ;ADJUST FOR RETURN
1586 030614  000200                1$:     RTS     R0              ;RETURN
1587
1588                                ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
1589                                ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
1590                                ;CALL
1591                                ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
1592                                ;       JSR     R0,CK.DEC       ;CHECK THE CHARACTER
1593                                ;       RETURN1                 ;NOT BETWEEN 0 AND 9
1594                                ;       RETURN2                 ;BETWEEN 0 AND 9
1595                                ;                               ;R2 = DIGIT
1596
1597 030616  121127  000060        CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
1598 030622  103407                        BLO     1$              ;YES -- BRANCH
1599 030624  121127  000071                CMPB    (R1),#'9                ;GREATER THAN NINE?
1600 030630  101004                        BHI     1$              ;YES -- BRANCH
1601 030632  111102                        MOVB    (R1),R2         ;GET THE CHARACTER
1602 030634  042702  000060                BIC     #'0,R2          ;STRIP AWAY THE ASCII
1603 030640  005720                        TST     (R0)+           ;ADJUST FOR RETURN
1604 030642  000200                1$:     RTS     R0              ;RETURN
1605
1606                                ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
1607                                ;DETERMINE WHAT IT IS.
1608                                ;CALL
```

```
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                        ;       JSR     R0,CK.CHR       ;CHECK CHARACTER
                                        ;       RETURN  ADR1            ;UNKNOWN CHARACTER
                                        ;       RETURN  ADR2            ;CARRIAGE RETURN * (R1)=ADR+1
                                        ;       RETURN  ADR3            ;SLASH * (R1)=ADR+1
                                        ;       RETURN  ADR4            ;COMMA * (R1)=ADR+1
                                        ;       RETURN  ADR5            ;PERIOD * (R1)=ADR+1
                                        ;       RETURN  ADR6            ;DIGIT BETWEEN 0 AND 9.
                                        ;                               ;R2 = DIGIT * (R1)=ADR+1
                                        ;
030644  105711                  CK.CHR: TSTB    (R1)            ;''CARRIAGE RETURN''?
030646  001420                          BEQ     4$              ;YES -- BRANCH
030650  121127  000057                  CMPB    (R1),#'/        ;''SLASH''?
030654  001414                          BEQ     3$              ;YES -- BRANCH
030656  121127  000054                  CMPB    (R1),#',        ;''COMMA''?
030662  001410                          BEQ     2$              ;YES -- BRANCH
030664  121127  000056                  CMPB    (R1),#'.        ;''PERIOD''?
030670  001404                          BEQ     1$              ;YES -- BRANCH
030672  004037  030616                  JSR     R0,CK.DEC       ;''DIGIT''?
030676  000406                          BR      5$              ;NO -- BRANCH
030700  005720                          TST     (R0)+           ;DIGIT BETWEEN 0-9
030702  005720                  1$:     TST     (R0)+           ;PERIOD
030704  005720                  2$:     TST     (R0)+           ;COMMA
030706  005720                  3$:     TST     (R0)+           ;SLASH
030710  005720                  4$:     TST     (R0)+           ;CARRIAGE RETURN
030712  005201                          INC     R1              ;MOVE POINTER TO NEXT CHARACTER
030714  011000                  5$:     MOV     (R0),R0         ;UNKNOWN CHARACTER
030716  000200                          RTS     R0              ;RETURN
                                
                                ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
                                ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
                                ;CALL
                                ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
                                ;       MOV     #NUM,R2         ;MAX. MAGNITUDE OF INPUT NUMBER
                                ;       JSR     R0,CK.DIG       ;CHECK DIGITS
                                ;       RETURN  ADR1            ;ILLEGAL CHARACTER -- R2=?
                                ;       RETURN  ADR2            ;INPUT NUMBER TO LARGE -- R2=?
                                ;       RETURN  ADR3            ;''COMMA'' -- R2 = NUMBER
                                ;       RETURN  ADR4            ;''PERIOD'' -- R2 = NUMBER
                                ;       RETURN  ADR5            ;''PERIOD-PERIOD'' -- R2 = NUMBER
                                
030720  010446                  CK.DIG: MOV     R4,-(SP)        ;SAVE R4
030722  010346                          MOV     R3,-(SP)        ;SAVE R3
030724  010246                          MOV     R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK
030726  005002                          CLR     R2              ;START WITH 0
030730  005003                          CLR     R3
030732  005004                          CLR     R4
030734  004037  030644                  JSR     R0,CK.CHR       ;CHECK ONE CHARACTER
030740  031070                          9$                      ;ILLEGAL CHARACTER
030742  031070                          9$                      ;CARRIAGE RETURN
030744  031070                          9$                      ;''/''
030746  031070                          9$                      ;''-''
030750  031070                          9$                      ;''.''
030752  030754                          1$                      ;DIGIT 0-9
030754  006303                  1$:     ASL     R3              ;2
030756  010346                          MOV     R3,-(SP)        ;SAVE *2
030760  006303                          ASL     R3              ;4
```

```
1660 030762  006303                    ASL     R3              ;8
1661 030764  062603                    ADD     (SP)+,R3        ;(*8)+(*2)=*10.
1662 030766  060203                    ADD     R2,R3           ;UPDATE THE INPUT NUMBER
1663 030770  004037  030644            JSR     R0,CK.CHR       ;CHECK ONE CHARACTER
     030774  031070                    9$                      ;ILLEGAL CHARACTER
     030776  031010                    2$                      ;CARRIAGE RETURN
     031000  031070                    9$                      ;"/"
     031002  031016                    4$                      ;","
     031004  031014                    3$                      ;"."
     031006  030754                    1$                      ;DIGIT 0-9
1664 031010  005301            2$:     DEC     R1              ;BACKUP THE CHARACTER POINTER
1665 031012  000401                    BR      4$              ;CONTINE
1666 031014  005724            3$:     TST     (R4)+           ;"PERIOD"
1667 031016  005724            4$:     TST     (R4)+           ;"COMMA" OR "CR"
1668 031020  004037  030644            JSR     R0,CK.CHR       ;CHECK ONE CHARACTER
     031024  031070                    9$                      ;ILLEGAL CHARACTER
     031026  031060                    7$                      ;CARRIAGE RETURN
     031030  031070                    9$                      ;"/"
     031032  031070                    9$                      ;","
     031034  031040                    5$                      ;"."
     031036  031050                    6$                      ;DIGIT 0-9
1669 031040  005724            5$:     TST     (R4)+           ;"PERIOD-PERIOD"
1670 031042  105711                    TSTB    (R1)            ;"CR"?
1671 031044  001405                    BEQ     7$              ;YES--BRANCH
1672 031046  000410                    BR      9$
1673 031050  126127  177776  000054  6$:     CMPB    -2(R1),#',      ;WAS CHARACTER BEFORE THE DIGIT A COMMA?
1674 031056  001004                    BNE     9$              ;NO--EXIT
1675 031060  020316            7$:     CMP     R3,(SP)         ;INPUT TO LARGE?
1676 031062  101001                    BHI     8$              ;YES -- BRANCH
1677 031064  060400                    ADD     R4,R0           ;ADJUST RETURN ADDRESS
1678 031066  005720            8$:     TST     (R0)+
1679 031070  010302            9$:     MOV     R3,R2           ;NUMBER TO R2
1680 031072  005726                    TST     (SP)+           ;CLEAN MAX. SIZE OFF OF STACK
1681 031074  012603                    MOV     (SP)+,R3        ;RESTORE R3
1682 031076  012604                    MOV     (SP)+,R4        ;RESTORE R4
1683 031100  011000                    MOV     (R0),R0         ;GET RETURN ADDRESS
1684 031102  000200                    RTS     R0              ;RETURN
1685
1686                                   ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
1687                                   ;AND FORMS AN OCTAL NUMBER IN R2
1688                                   ;CALL:
1689                                   ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
1690                                   ;       JSR     R0,CK.NUM       ;GO FORM THE NUMBER
1691                                   ;       RETURN  ADR1            ;ILLEGAL CHARACTER IN THE INPUT STRING
1692                                   ;       RETURN  ADR2            ;"COMMA" OR "CR"--R2=NUMBER
1693                                   ;       RETURN  ADR3            ;"PERIOD"--R2=NUMBER
1694                                   ;       RETURN  ADR4            ;"PERIOD-PERIOD"--R2=NUMBER
1695
1696 031104  010346            CK.NUM: MOV     R3,-(SP)        ;SAVE R3
1697 031106  005003                    CLR     R3              ;START NUMBER AT ZERO
1698 031110  004037  030570            JSR     R0,CK.OCT       ;OCTAL DIGIT?
1699 031114  000440                    BR      6$              ;NO--BRANCH
1700 031116  005201            1$:     INC     R1              ;MOVE TO NEXT CHARACTER
1701 031120  006303                    ASL     R3              ;FOR THE OCTAL NUMBER IN R3
1702 031122  103435                    BCS     6$              ;DON'T LET IT GET TO BIG
1703 031124  006303                    ASL     R3
1704 031126  103433                    BCS     6$
```

```
1705 031130  006303                    ASL     R3
1706 031132  103431                    BCS     6$
1707 031134  060203                    ADD     R2,R3
1708 031136  004037  030570            JSR     R0,CK.OCT       ;IS THIS AN OCTAL DIGIT?
1709 031142  000401                    BR      2$              ;NO--FIND OUT WHAT IT IS
1710 031144  000764                    BR      1$              ;YES--MAKE IT PART OF THE NUMBER
1711 031146  010302             2$:    MOV     R3,R2           ;SAVE THE OCTAL NUMBER
1712 031150  005003                    CLR     R3              ;START WITH ZERO INDEX
1713 031152  004037  030644            JSR     R0,CK.CHR       ;CHECK ONE CHARACTER
     031156  031216                    6$                      ;ILLEGAL CHARACTER
     031160  031206                    5$                      ;CARRIAGE RETURN
     031162  031216                    6$                      ;"/"
     031164  031206                    5$                      ;","
     031166  031172                    3$                      ;"."
     031170  031216                    6$                      ;DIGIT 0-9
1714 031172  005723             3$:    TST     (R3)+           ;"PERIOD"
1715 031174  121127  000056            CMPB    (R1),#'.        ;"PERIOD-PERIOD"?
1716 031200  001002                    BNE     5$              ;NO--BRANCH
1717 031202  005201                    INC     R1              ;YES--ADVANCE THE POINTER
1718 031204  005723             4$:    TST     (R3)+           ;"PERIOD-PERIOD"
1719 031206  005723             5$:    TST     (R3)+           ;"COMMA"
1720 031210  105711                    TSTB    (R1)            ;"CR"?
1721 031212  001001                    BNE     6$              ;NO--BRANCH
1722 031214  060300                    ADD     R3,R0           ;YES--SAVE THE OCTAL NUMBER
1723 031216  012603             6$:    MOV     (SP)+,R3        ;RESTORE R3
1724 031220  011000                    MOV     (R0),R0         ;PICKUP EXIT ADDRESS
1725 031222  000200                    RTS     R0              ;RETURN
```

```
                                .SBTTL  SINGLE/DUAL PORT RH70/RM80 DRIVER (REV 0.17)

                                ;COPYRIGHT (C) 1979,1981
                                ;DIGITAL EQUIPMENT CORP.
                                ;MAYNARD, MA 01754
                                ;AUTHOR(S): CHUCK HESS
                                ;REVISED BY: MIKE LEAVITT

                                ;;*****************************************************************

                                ;STORAGE FOR RMDS, RMER1 AND RMER2 ON AN ERROR "2"
                                        ;RMERRS   = RMDS
                                        ;RMERRS+2 = RMER1
                                        ;RMERRS+4 = RMER2

031224  000000  000000  000000  RMERRS: .WORD   0,0,0

                                ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
                                        ;DRVACT=0 IF DRIVE IS IDLE
                                        ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
                                        ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION

031232      000                 DRVACT: .BYTE   0               ;DRIVE 0
031233      000                         .BYTE   0               ;DRIVE 1
031234      000                         .BYTE   0               ;DRIVE 2
031235      000                         .BYTE   0               ;DRIVE 3
031236      000                         .BYTE   0               ;DRIVE 4
031237      000                         .BYTE   0               ;DRIVE 5
031240      000                         .BYTE   0               ;DRIVE 6
031241      000                         .BYTE   0               ;DRIVE 7

                                ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
                                        ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXSITENT
                                        ;DRVSTA>0 IF DRIVE IS ONLINE
                                        ;DRVSTA<0 IF DRIVE IS UNSAFE

031242      000                 DRVSTA: .BYTE   0               ;DRIVE 0
031243      000                         .BYTE   0               ;DRIVE 1
031244      000                         .BYTE   0               ;DRIVE 2
031245      000                         .BYTE   0               ;DRIVE 3
031246      000                         .BYTE   0               ;DRIVE 4
031247      000                         .BYTE   0               ;DRIVE 5
031250      000                         .BYTE   0               ;DRIVE 6
031251      000                         .BYTE   0               ;DRIVE 7

                                ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
                                        ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
                                        ;DRVTYP=1 IF DRIVE IS RM80
                                        ;DRVTYP=-1 IF NOT RM80

031252      000                 DRVTYP: .BYTE   0               ;DRIVE 0
031253      000                         .BYTE   0               ;DRIVE 1
031254      000                         .BYTE   0               ;DRIVE 2
031255      000                         .BYTE   0               ;DRIVE 3
```

```
        031256      000                         .BYTE   0               ;DRIVE 4
        031257      000                         .BYTE   0               ;DRIVE 5
        031260      000                         .BYTE   0               ;DRIVE 6
        031261      000                         .BYTE   0               ;DRIVE 7
     74
     75                                 ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
     76                                         ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
     77                                         ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
     78
     79 031262      000                 DPINT:  .BYTE   0               ;DRIVE 0
     82 031263      000                         .BYTE   0               ;DRIVE 1
        031264      000                         .BYTE   0               ;DRIVE 2
        031265      000                         .BYTE   0               ;DRIVE 3
        031266      000                         .BYTE   0               ;DRIVE 4
        031267      000                         .BYTE   0               ;DRIVE 5
        031270      000                         .BYTE   0               ;DRIVE 6
        031271      000                         .BYTE   0               ;DRIVE 7
     83
     84                                 ;TABLE OF PENDING DUAL PORT REQUESTS
     85                                         ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
     86                                         ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
     87
     88 031272      000                 DPRQS:  .BYTE   0               ;DRIVE 0
     91 031273      000                         .BYTE   0               ;DRIVE 1
        031274      000                         .BYTE   0               ;DRIVE 2
        031275      000                         .BYTE   0               ;DRIVE 3
        031276      000                         .BYTE   0               ;DRIVE 4
        031277      000                         .BYTE   0               ;DRIVE 5
        031300      000                         .BYTE   0               ;DRIVE 6
        031301      000                         .BYTE   0               ;DRIVE 7
     92
     93                                 ;DRIVE REQUEST QUE WORDS
     94
     95 031302   000000                 QDRV:   .WORD   0               ;DRIVE 0
     98 031304   000000                         .WORD   0               ;DRIVE 1
        031306   000000                         .WORD   0               ;DRIVE 2
        031310   000000                         .WORD   0               ;DRIVE 3
        031312   000000                         .WORD   0               ;DRIVE 4
        031314   000000                         .WORD   0               ;DRIVE 5
        031316   000000                         .WORD   0               ;DRIVE 6
        031320   000000                         .WORD   0               ;DRIVE 7
     99
    100                                 ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
    101                                         ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
    102                                         ;''DPB'' OF THE I/O OPERATION.
    103
    104 031322   000000                 TRNSWT: .WORD   0
    105
    106                                 ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
    107                                         ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
    108                                         ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
    109                                         ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
    110                                         ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
    111
    112 031324   000000                 SRCHWT: .WORD   0
    113
    114                                 ;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
```

```
                                        ;ACTDRV=0 IF DRIVER IS INACTIVE
                                        ;ACTDRV>0 IF DRIVER IS ACTIVE

031326      000                 ACTDRV: .BYTE   0

                                ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
                                        ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
                                        ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE

031327      000                 ACTSTR: .BYTE   0

                                ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
                                        ;SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
                                        ;OPERATION IS COMPLETED AS PER (DPB+14).
                                        ;SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
                                        ;(DPB+14), AFTER AN ERROR.

031330  000000                  SAVEFG: .WORD   0

                                ;SEEK FLAG (SEEKFG=1 WORD)
                                        ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
                                        ;FOR A DATA TRANSFER START A SEARCH COMMAND
                                        ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
                                        ;DISREGARD THE WINDOW

031332  177777                  SEEKFG: .WORD   -1

                                ;TIMEOUT TABLE (TIMER=8 WORDS)
                                        ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION

031334  177777                  TIMER:  .WORD   -1               ;DRIVE 0
148 031336  177777                      .WORD   -1               ;DRIVE 1
    031340  177777                      .WORD   -1               ;DRIVE 2
    031342  177777                      .WORD   -1               ;DRIVE 3
    031344  177777                      .WORD   -1               ;DRIVE 4
    031346  177777                      .WORD   -1               ;DRIVE 5
    031350  177777                      .WORD   -1               ;DRIVE 6
    031352  177777                      .WORD   -1               ;DRIVE 7

                                ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
                                        ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
                                        ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N

031354  177777                  DTUW:   .WORD   -1

                                ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
                                        ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
                                        ;ATTENTION BIT

C31356      001     002     004 ATABIT: .BYTE   1,2,4,10,20,40,100,200

                                ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH70/RM80),
                                ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).

031366  176700                  RMADR:  .WORD   176700
166 031370  000254  000240      RMVEC:  .WORD   254,5*32.
```

```
                                ;SEARCH DIFFERENCE IS 5 FIVE SECTORS

031374  000005                MXWNDW: .WORD   5

                                ;DEFINITIONS OF THE RH70/RM80 ADDRESS INDEXES

        000000                  RMCS1   = 0             ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 0)
        000002                  RMWC    = 2             ;WORD COUNT REGISTER (NOT A DRIVE REG)
        000004                  RMBA    = 4             ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
        000006                  RMDA    = 6             ;DESIRED TRK/SEC ADDRESS REGISTER (DRIVE REG. 5)
        000010                  RMCS2   = 10            ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
        000012                  RMDS    = 12            ;DRIVE STATUS REGISTER (DRIVE REG 1)
        000014                  RMER1   = 14            ;ERROR REGISTER #1 (DRIVE REG. 2)
        000016                  RMAS    = 16            ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 4)
        000020                  RMLA    = 20            ;LOOK AHEAD REGISTER (DRIVE REG. 7)
        000022                  RMDB    = 22            ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
        000024                  RMMR1   = 24            ;MAINTAINABILITY REGISTER #1  (DRIVE REG. 3)
        000026                  RMDT    = 26            ;DRIVE TYPE REGISTER (DRIVE REG. 6)
        000030                  RMSN    = 30            ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
        000032                  RMOF    = 32            ;OFFSET REGISTER (DRIVE REG. 11)
        000034                  RMDC    = 34            ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
        000036                  RMHR    = 36            ;"HOLDING REGISTER" (DRIVE REG. 13)
        000040                  RMMR2   = 40            ;MAINTENANCE REGISTER #2 (DRIVE REG. 14)
        000042                  RMER2   = 42            ;ERROR REGISTER #2 (DRIVE REG. 15)
        000044                  RMEC1   = 44            ;ECC POSITION REGISTER (DRIVE REG. 16)
        000046                  RMEC2   = 46            ;ECC PATTERN REGISTER (DRIVE REG.  17)

```

```
                                        ;RH70/RM80 DRIVER INITIALIZATION CODE
                                                ;THIS ROUTINE WILL DETERMINE WHICH RM80 DRIVES ARE
                                                ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
                                                ;TO THE PROPER STATE FOR EACH DRIVE.
                                                ;NOTE: THIS ROUTINE CALLS DRVINT
                                        ;
                                        ;CALL
                                        ;
                                        ;       JSR     PC,RMINIT
                                        ;       RETURN
                                        ;
                                        ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
                                        ;
031376  104412                 RMINIT: SAVREG                  ;SAVE R0 - R5
031400  013746  177776                  MOV     @#PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
031404  012737  000240  177776          MOV     #<5*32.>,@#PS   ;CHANGE THE PRIORITY TO 5
031412  004737  036134                  JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
031416  012701  031224                  MOV     #RMERRS,R1      ;FIRST ADDRESS TO BE CLEARED
031422  012702  031332                  MOV     #SEEKFG,R2      ;LAST ADDRESS TO BE CLEARED
031426  005021                  1$:     CLR     (R1)+           ;CLEAR
031430  020102                          CMP     R1,R2           ;ARE WE DONE?
031432  101775                          BLOS    1$              ;BRANCH IF NO
031434  012702  031354                  MOV     #DTUW,R2        ;LAST ADDRESS
031440  012721  177777          2$:     MOV     #-1,(R1)+       ;INITIALIZE
031444  020102                          CMP     R1,R2           ;DONE?
031446  101774                          BLOS    2$              ;LOOP IF NO
031450  005037  031242                  CLR     DRVSTA          ;SET ALL DRIVES TO OFFLINE
031454  005037  031244                  CLR     DRVSTA+2
031460  005037  031246                  CLR     DRVSTA+4
031464  005037  031250                  CLR     DRVSTA+6
031470  013703  031370                  MOV     RMVEC,R3        ;SETUP THE RH70/RM80 VECTOR
031474  012723  033526                  MOV     #ISR,(R3)+
031500  013713  031372                  MOV     RMVEC+2,(R3)
031504  013704  031366                  MOV     RMADR,R4        ;FIRST ADDRESS OF RH70/RM80
031510  012764  000040  000010          MOV     #40,RMCS2(R4)   ;MASSBUS INIT
031516  005001                          CLR     R1              ;START WITH DRIVE 0
031520  004037  031610          3$:     JSR     R0,DRVINT       ;INIT THE DRIVE
031524  000401                          BR      4$              ;'DVA' NOT SET
031526  000402                          BR      5$              ;NORMAL RETURN
031530  105061  031242          4$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
031534  005201                  5$:     INC     R1              ;GO TO NEXT DRIVE
031536  042701  177770                  BIC     #^C7,R1         ;MASK OUT UNUSED BITS
031542  001366                          BNE     3$              ;BR IF MORE DRIVES TO GO
031544  012701  000007                  MOV     #7,R1           ;START WITH DRIVE 7
031550  005037  177776                  CLR     @#PS            ;CLEAR THE PROCESSOR STATUS
031554  105761  031262          6$:     TSTB    DPINT(R1)       ;WAITING FOR DRIVE TO SWITCH PORTS ?
031560  001405                          BEQ     8$              ;BR NOT WAITING
031562  004737  036062                  JSR     PC,SET.IE       ;SET INTERRUPT
031566  105761  031262          7$:     TSTB    DPINT(R1)       ;DRIVE SWITCHED PORTS ?
031572  001375                          BNE     7$              ;BR IF NOT
031574  005301                  8$:     DEC     R1              ;GO TO THE NEXT DRIVE
031576  100366                          BPL     6$              ;CHECK NEXT DRIVE
031600  012637  177776                  MOV     (SP)+,@#PS      ;RESTORE THE PROCESSOR STATUS
031604  104413                          RESREG                  ;RESTORE R0 - R5
031606  000207                          RTS     PC              ;BYE-BYE

                                        ;DRIVE INITILIZATION ROUTINE
```

```
                                                ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
                                                ;AN RM80. IF IT IS, A 'READ-IN PRESET" IS ISSUED AND FMT16
                                                ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
                                                ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE.
                                                ;DRVSTA IS SET TO THE PROPER CONDITION.
                                        ;CALL
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                        ;       MOV     RMADR,R4        ;UNIBUS ADDRESS OF RH70/RM80 (RMCS1)
                                        ;       JSR     R0,DRVINT       ;CALLED BY A JSR
                                        ;       RETURN1                 ;ERROR OCCURRED ('NED')
                                        ;       RETURN2                 ;NORMAL RETURN
                                        ;

031610  010546                 DRVINT: MOV     R5,-(SP)        ;SAVE R5
031612  105061  031242                 CLRB    DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
031616  105061  031252                 CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
031622  010164  000010                 MOV     R1,RMCS2(R4)    ;SELECT A DRIVE
031626  112764  000111  000000         MOVB    #111,RMCS1(R4)  ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
031634  032764  010000  000010         BIT     #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
031642  001403                         BEQ     1$              ;NO---BRANCH
031644  004737  036062                 JSR     PC,SET.IE       ;GO SET "IE" WITHOUT A "TRE"
031650  000476                         BR      4$              ;LEAVE THIS ROUTINE
031652  105061  031242         1$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
031656  032764  004000  000000         BIT     #BIT11,RMCS1(R4)  ;SEE IF DRIVE AVAILABLE
031664  001470                         BEQ     4$              ;BR IF DRIVE NOT AVAILABLE
031666  004037  035552                 JSR     R0,RD.RM        ;CALL THE READ ROUTINE
031672  000026                         RMDT                    ;REGISTER OFFSET
031674  032050                         5$                      ;'NED' RETURN
031676  012605                         MOV     (SP)+,R5        ;PUT DRIVE TYPE IN R5
031700  112761  000001  031252         MOVB    #1,DRVTYP(R1)   ;SET RM80 INDICATOR
031706  022705  020026                 CMP     #20026,R5       ;IS IT A SINGLE PORT RM80?
031712  001407                         BEQ     2$              ;BRANCH IF YES
031714  022705  024026                 CMP     #24026,R5       ;IS IT A DUAL PORT RM80?
031720  001404                         BEQ     2$              ;BR IF YES
031722  112761  177777  031252         MOVB    #-1,DRVTYP(R1)  ;SET INDICATOR TO 'OTHER'
031730  000446                         BR      4$              ;EXIT
031732  012746  000121         2$:     MOV     #121,-(SP)      ;DO A "READ-IN PRESET"
031736  004037  035632                 JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
031742  000000                         RMCS1                   ;REGISTER OFFSET
031744  032050                         5$                      ;'NED' RETURN
031746  012746  010000                 MOV     #BIT12,-(SP)    ;SET FMT16=1
031752  004037  035632                 JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
031756  000032                         RMOF                    ;REGISTER OFFSET
031760  032050                         5$                      ;'NED' RETURN
031762  004037  035552                 JSR     R0,RD.RM        ;CALL THE READ ROUTINE
031766  000012                         RMDS                    ;REGISTER OFFSET
031770  032050                         5$                      ;'NED' RETURN
031772  012605                         MOV     (SP)+,R5        ;AND SAVE IT IN R5
031774  100015                         BPL     3$              ;BRANCH IF ATA=0
031776  116164  031356  000016         MOVB    ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
032004  004037  035552                 JSR     R0,RD.RM        ;CALL THE READ ROUTINE
032010  000014                         RMER1                   ;REGISTER OFFSET
032012  032050                         5$                      ;'NED' RETURN
032014  006126                         ROL     (SP)+           ;IS IT UNSAFE?
032016  100004                         BPL     3$              ;BR IF NOT
032020  112761  177777  031242         MOVB    #-1,DRVSTA(R1)  ;SET UNSAFE INDICATOR
032026  000407                         BR      4$              ;EXIT
```

```
105 032030  005105                    3$:     COM     R5                  ;CHECK MOL, DPR, DRY, AND VV
106 032032  042705  167077                    BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
107 032036  001003                            BNE     4$                  ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
108 032040  112761  000001  031242            MOVB    #1,DRVSTA(R1)       ;SET DRIVE STATUS TO ONLINE
109 032046  005720                    4$:     TST     (R0)+               ;STEP OVER THE ERROR RETURN
110 032050  012605                    5$:     MOV     (SP)+,R5            ;RESTORE R5
111 032052  000200                            RTS     R0                  ;EXIT
```

```
                                        ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
                                        ;
                                        ;CALL
                                        ;
                                        ;       JSR     R0,RM80         ;CALL THE RM80 DRIVER
                                        ;       PNTADR                  ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
                                        ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
                                        ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
                                                                        ;IS AN ERROR CONDITION

032054  013746  177776          RM80:   MOV     @#PS,-(SP)      ;SAVE THE CALLING STATUS
032060  013737  031372  177776          MOV     RMVEC+2,@#PS    ;DON'T ALLOW ANY RM80 INTERRUPTS
032066  112737  000001  031326          MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
032074  104412                          SAVREG                  ;SAVE R0 - R5
032076  011002                          MOV     (R0),R2         ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
032100  005062  000016                  CLR     16(R2)          ;CLEAR THE STATUS/ERROR INDICATOR
032104  111201                          MOVB    (R2),R1         ;PICKUP THE DRIVE NUMBER
032106  013704  031366                  MOV     RMADR,R4        ;UNIBUS ADDRESS OF RMCS1
032112  105761  031242                  TSTB    DRVSTA(R1)      ;CHECK DRIVES STATUS
032116  003011                          BGT     1$              ;BRANCH IF ONLINE
032120  105761  031262                  TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE
032124  001027                          BNE     4$              ;BR IF YES
032126  004037  031610                  JSR     R0,DRVINT       ;GO INIT. THE DRIVE
032132  000421                          BR      3$              ;ERROR RETURN
032134  105761  031242                  TSTB    DRVSTA(R1)      ;IS DRIVE STATUS ONLINE?
032140  003432                          BLE     5$              ;BR IF NOT
032142  105761  031272          1$:     TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
032146  001016                          BNE     4$              ;BR IF YES
032150  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
032154  004037  036200                  JSR     R0,DRVQUE       ;PUT THIS REQUEST IN QUEUE
032160  000445                          BR      8$              ;QUEUE IS FULL
032162  105761  031232          2$:     TSTB    DRVACT(R1)      ;IS THIS DRIVE ACTIVE?
032166  001037                          BNE     7$              ;BR IF YES
032170  004737  032312                  JSR     PC,OPT          ;CALL THE OPTIMIZER
032174  000434                          BR      7$
032176  004737  033410          3$:     JSR     PC,CI8          ;GO HANDLE THE 'NED'
032202  000431                          BR      7$
032204  004037  036200          4$:     JSR     R0,DRVQUE       ;PUT REQUEST IN QUEUE
032210  000431                          BR      8$              ;QUEUE IS FULL
032212  032714  000100                  BIT     #BIT6,(R4)      ;IS 'IE' SET ALREADY ?
032216  001023                          BNE     7$              ;BR IF IT IS
032220  004737  036062                  JSR     PC,SET.IE       ;SET INTERRUPT
032224  000420                          BR      7$              ;RETURN, REQUEST IN QUEUE
032226  105761  031242          5$:     TSTB    DRVSTA(R1)      ;SEE IF DRIVE OFFLINE OR UNSAFE
032232  002412                          BLT     6$              ;BR IF UNSAFE
032234  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE ERROR INDICATOR
032242  105761  031252                  TSTB    DRVTYP(R1)      ;SEE IF OFFLINE OR NONEXISTENT
032246  001007                          BNE     7$              ;BR IF OFFLINE
032250  012762  100002  000016          MOV     #BIT15!BIT01,16(R2)  ;REPORT DRIVE NONEXISTENT
032256  000403                          BR      7$              ;GO TO EXIT
032260  012762  110000  000016  6$:     MOV     #BIT15!BIT12,16(R2)  ;DRIVE IS UNSAFE
032266  104413                  7$:     RESREG                  ;RESTORE R0 - R5
032270  005720                          TST     (R0)+           ;SETUP FOR NORMAL RETURN
032272  000401                          BR      9$              ;FINISH UP, THEN EXIT
032274  104413                  8$:     RESREG                  ;RESTORE R0 - R5
032276  005720                  9$:     TST     (R0)+           ;CORRECT THE RETURN ADDRESS
032300  105037  031326                  CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
```

```
     58 032304  012637  177776                    MOV     (SP)+,@#PS      ;RETURN 'PS' TO USER LEVEL
     59 032310  000200                            RTS     R0              ;RETURN TO CALLER
     60
     61                                 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
     62                                 ;
     63                                 ;CALL
     64                                 ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
     65                                 ;       JSR     PC,OPT          ;SETUP A COMMAND
     66                                 ;
     67 032312  104412                  OPT:    SAVREG                  ;SAVE R0 - R5
     68 032314  013746  177776                  MOV     @#PS,-(SP)      ;SAVE PROC. STATUS
     69 032320  004737  036222                  JSR     PC,GETREQ       ;GET ''DPB'' POINTER OF REQUEST
     70 032324  005702                          TST     R2              ;IS THERE A REQUEST IN QUEUE?
     71 032326  001510                          BEQ     9$              ;NO--BRANCH TO EXIT
     72 032330  032764  004000  000000          BIT     #BIT11,RMCS1(R4)        ;IS DVA SET?
     73 032336  001407                          BEQ     2$              ;BRANCH IF NOT
     74 032340  032764  000100  000012          BIT     #BIT6,RMDS(R4)  ;IS VV SET ?
     75 032346  001003                          BNE     2$              ;BR IF IT IS
     76 032350  004037  031610          1$:     JSR     R0,DRVINT       ;SEE IF DRIVE STILL ONLINE ?
     77 032354  000473                          BR      8$              ;'DVA' NOT SET
     78 032356  105761  031242          2$:     TSTB    DRVSTA(R1)      ;IS DRIVE ONLINE?
     79 032362  003014                          BGT     3$              ;YES--BRANCH
     80 032364  004737  036234                  JSR     PC,POPQUE       ;NO--REMOVE REQUEST FROM QUEUE
     81 032370  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE STATUS/ERROR INDICATOR
     82 032376  105761  031242                  TSTB    DRVSTA(R1)      ;IS DRIVE UNSAFE ?
     83 032402  100067                          BPL     10$             ;BR TO EXIT IF NOT
     84 032404  012762  110000  000016          MOV     #BIT15!BIT12,16(R2)  ;SET UNSAFE STATUS/ERROR INDICATOR
     85 032412  000463                          BR      10$             ;BRANCH TO EXIT
     86 032414  012746  000111          3$:     MOV     #111,-(SP)      ;LOAD COMMAND ONTO THE STACK
     87 032420  004037  035632                  JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
        032424  000000                          RMCS1                   ;REGISTER OFFSET
        032426  032544                          8$                      ;'NED' RETURN
     88 032430  032714  004000                  BIT     #BIT11,(R4)     ;DRIVE AVAILABLE ?
     89 032434  001432                          BEQ     7$              ;BR IF NOT
     90 032436  122762  000150  000002          CMPB    #150,2(R2)      ;IS THE REQUEST FOR I/O?
     91 032444  002403                          BLT     4$              ;YES--BRANCH
     92 032446  004737  033032                  JSR     PC,CI4          ;CALL THE COMMAND INITIATOR
     93 032452  000443                          BR      10$             ;BRANCH TO EXIT
     94 032454  005737  031354          4$:     TST     DTUW            ;DATA TRANSFER UNDERWAY?
     95 032460  002015                          BGE     6$              ;YES--GO START A SEARCH
     96 032462  136137  031356  031324          BITB    ATABIT(R1),SRCHWT  ;FINISHED A SEARCH ?
     97 032470  001003                          BNE     5$              ;IF NE, YES
     98 032472  005737  031332                  TST     SEEKFG          ;DO IMPLIED SEEKS?
     99 032476  100006                          BPL     6$              ;IF PL DO A SEARCH
    100 032500  146137  031356  031324  5$:     BICB    ATABIT(R1),SRCHWT  ;CLEAR ''SEARCH WAIT'' KEY
    101 032506  004737  032572                  JSR     PC,CI1          ;START A DATA TRANSFER
    102 032512  000423                          BR      10$
    103 032514  004737  032724          6$:     JSR     PC,CI3          ;START A SEARCH
    104 032520  000420                          BR      10$             ;GO TO THE EXIT
    105 032522  112761  177777  031272  7$:     MOVB    #-1,DPRQS(R1)   ;SET PORT REQUEST INDICATOR
    106 032530  010103                          MOV     R1,R3           ;SET UP TO ADDRESS WORDS
    107 032532  006303                          ASL     R3              ;CONVERT TO WORD INDEX
    108 032534  012763  035230  031334          MOV     #15000.,TIMER(R3)  ;START 15 SECOND TIMER
    109 032542  000402                          BR      9$              ;EXIT
    110 032544  004737  033410          8$:     JSR     PC,CI8          ;PROCESS THE 'NED'
    111 032550  032714  000100          9$:     BIT     #BIT06,(R4)     ;SEE IF 'IE' ALREADY SET
    112 032554  001002                          BNE     10$             ;BR IF SET
```

```
                                        JSR     PC,SET.IE       ;SET "IE" WITHOUT A "TRE"
                                10$:    MOV     (SP)+,@#PS      ;RESTORE PROC. STATUS
                                        RESREG                  ;RESTORE R0 - R5
                                        RTS     PC

                                ;COMMAND INITIATOR
                                ;
                                ;CALL
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                ;       MOV     #DPB,R2         ;ADDRESS OF DPB
                                ;       JSR     PC,CI?          ;CI?= CI1,CI3, OR CI4
                                ;                               ;WHERE:
                                ;                               ;CI1=DATA TRANSFER
                                ;                               ;CI2=SEARCH REQUESTED BY DATA XFER
                                ;                               ;CI4=NOT DATA TRANSFER

                                ;START A DATA TRANSFER

032572  004737  036234         CI1:    JSR     PC,POPQUE       ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
032576  010237  031322                 MOV     R2,TRNSWT       ;PUT REQ. IN TRANSFER WAIT QUEUE
032602  010203                         MOV     R2,R3           ;DPB ADDRESS TO R3
032604  013704  031366                 MOV     RMADR,R4        ;RMCS1 ADDRESS
032610  010164  000010                 MOV     R1,RMCS2(R4)    ;SELECT DRIVE
032614  062703  000004                 ADD     #4,R3           ;DESIRED WORD COUNT
032620  062704  000002                 ADD     #2,R4           ;RMWC ADDRESS
032624  012324                         MOV     (R3)+,(R4)+     ;LOAD WORD COUNT
032626  012324                         MOV     (R3)+,(R4)+     ;LOAD BUFFER ADDRESS
032630  012346                         MOV     (R3)+,-(SP)     ;LOAD SECTOR AND TRACK
032632  004037  035632                 JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
032636  000006                         RMDA                    ;REGISTER OFFSET
032640  033410                         CI8                     ;"NED" RETURN
032642  012346                         MOV     (R3)+,-(SP)     ;LOAD CYLINDER ADDRESS
032644  004037  035632                 JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
032650  000034                         RMDC                    ;REGISTER OFFSET
032652  033410                         CI8                     ;"NED" RETURN
032654  004037  035552                 JSR     R0,RD.RM        ;CALL THE READ ROUTINE
032660  000032                         RMOF                    ;REGISTER OFFSET
032662  033410                         CI8                     ;"NED" RETURN
032664  042716  001000                 BIC     #BIT09,(SP)     ;CLEAR "SKIP SECTOR ERROR INHIBIT"
032670  004037  035632                 JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
032674  000032                         RMOF                    ;REGISTER OFFSET
032676  033410                         CI8                     ;"NED" RETURN
032700  016246  000002                 MOV     2(R2),-(SP)     ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
032704  004037  035632                 JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
032710  000000                         RMCS1                   ;REGISTER OFFSET
032712  033410                         CI8                     ;"NED" RETURN
032714  010137  031354                 MOV     R1,DTUW         ;SET "DATA TRANSFER UNDERWAY"
032720  000137  033366                 JMP     CI5

                                ;START A SEARCH

032724  013704  031366         CI3:    MOV     RMADR,R4        ;RMCS1 ADDRESS
032730  010164  000010                 MOV     R1,RMCS2(R4)    ;SELECT DRIVE
032734  016246  000012                 MOV     12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
032740  004037  035632                 JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
032744  000034                         RMDC                    ;REGISTER OFFSET
032746  033410                         CI8                     ;"NED" RETURN
```

```
                                        MOVB    10(R2),R3       ;PICKUP SECTOR ADDRESS
                                        SUB     MXWNDW,R3       ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
                                        BGE     1$
                                        ADD     #31.,R3
                                1$:     MOV     R3,-(SP)        ;COMBINE THE ADJUSTED SECTOR WITH
                                        MOVB    11(R2),1(SP)    ;THE DESIRED TRACK
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMDA                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        MOV     #131,-(SP)      ;START A SEARCH
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMCS1                   ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        BISB    ATABIT(R1),SRCHWT       ;SET "SEARCH WAIT" KEY
                                        BR      CI5

                                ;INITIATE A NON-I/O OPERATION

                                CI4:    MOV     RMADR,R4        ;RMCS1 ADDRESS
                                        MOV     R1,RMCS2(R4)    ;SELECT DRIVE
                                        MOVB    2(R2),R3        ;PICKUP THE REQUESTED COMMAND
                                        CMPB    #131,R3         ;IS IT A SEARCH COMMAND?
                                        BNE     1$              ;BRANCH IF NO
                                        MOV     10(R2),-(SP)    ;LOAD DESIRED TRACK & SECTOR
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMDA                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        BR      2$              ;GO LOAD CYLINDER
                                1$:     CMPB    #105,R3         ;IS IT A SEEK COMMAND
                                        BNE     3$              ;BRANCH IF NO
                                2$:     MOV     12(R2),-(SP)    ;LOAD DESIRED CYLINDER
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMDC                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        BR      14$
                                3$:     CMPB    #115,R3         ;IS IT AN "OFFSET" COMMAND?
                                        BNE     4$              ;BR IF NO
                                        JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                        RMOF                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        MOVB    1(R2),(SP)      ;BYTE WHEN LOADING THE REGISTER
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMOF                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        BR      14$
                                4$:     CMPB    #107,R3         ;IS IT A "RECALIBRATE" COMMAND?
                                        BNE     5$              ;IF NE, NO
                                        CLR     -(SP)           ;CYLINDER ZERO
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMDC                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
                                        BR      14$             ;CONTINUE
                                5$:     CMPB    #143,R3         ;IS IT A "SET FORMAT" COMMAND?
                                        BNE     6$              ;BRANCH IF NO
                                        JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                        RMOF                    ;REGISTER OFFSET
                                        CI8                     ;'NED' RETURN
```

```
                                                MOVB    1(R2),1(SP)     ;COMBINE 'FMT16','ECI','HCI', & 'SSEI'
                                                JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                                RMOF                    ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN
                                                BR      12$
                                        6$:     CMPB    #141,R3         ;IS IT A "GET REGISTER" COMMAND?
                                                BNE     10$             ;BRANCH IF NO
                                        7$:     MOV     6(R2),R3        ;POINTS TO 1ST ADDRESS OF WHERE
                                                                        ;TO PUT THE REGISTER(S)
                                                MOVB    10(R2),9$       ;INIT. THE INDEX FOR THE FIRST REG.
                                                MOVB    11(R2),R5       ;INDEX OF LAST REG. TO MOVE
                                        8$:     JSR     R0,RD.RM        ;READ RM80 REGISTER
                                        9$:     RMCS1                   ;INDEX OF REG. TO READ
                                                CI8
                                                MOV     (SP)+,(R3)+     ;GET THE CONTENTS OF RH70/RM80 REG.
                                                CMP     9$,R5           ;LAST REG. BEEN READ?
                                                BEQ     12$             ;GET OUT IF YES
                                                ADD     #2,9$           ;INCREASE THE INDEX BY 2
                                                BR      8$              ;LOOP--MORE TO READ
                                       10$:     CMPB    #145,R3         ;IS IT A "SELECT DRIVE" COMMAND?
                                                BEQ     12$             ;BRANCH IF YES
                                       11$:     MOV     R3,-(SP)        ;LOAD THE COMMAND
                                                JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                                RMCS1                   ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN
                                       12$:     JSR     PC,POPQUE       ;REMOVE REQ. FROM QUEUE
                                                BIS     #BIT07,16(R2)   ;SET THE "DONE" BIT
                                                TST     SAVEFG          ;SAVE THE RH70/RM80 REGISTERS?
                                                BPL     13$             ;BRANCH IF NO
                                                JSR     PC,SVRH70       ;YES--GO SAVE THE REGISTERS
                                       13$:     RTS     PC              ;RETURN TO USER

                                        ;START A NON-DATA TRANSFER OPERATION

                                        14$:    MOV     R3,-(SP)        ;LOAD THE COMMAND
                                                JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                                RMCS1                   ;REGISTER OFFSET
                                                CI8                     ;'NED' RETURN

                                        ;START THE COMMAND TIMER

                                        CI5:    ASL     R1
                                                MOV     #10000.,TIMER(R1)       ;SET A 10 SECOND TIMER
                                                ASR     R1
                                                MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
                                                RTS     PC              ;RETURN TO THE USER

                                        ;PROCESS A NON-EXISTENT DRIVE

                                        CI8:    SAVREG                  ;SAVE R0 - R5
                                                TSTB    DRVACT(R1)      ;DRIVE ACTIVE?
                                                BEQ     3$              ;BRANCH IF NO
                                                MOV     TRNSWT,R2       ;GET THE "TRANSFER WAIT" QUEUE
                                                CMP     R1,DTUW         ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
                                                BEQ     1$              ;BRANCH IF YES
                                                JSR     PC,GETREQ       ;GET THE DPB POINTER
                                        1$:     TST     R2              ;QUEUE ENTRY FOR DRIVE ?
```

```
033440  001403                          BEQ     2$                      ;BR IF NOT
033442  012762  100002  000016          MOV     #BIT15!BIT01, 6(R2)     ;SET 'DRIVE NON-EXISTENT' INDICATOR
033450  012763  177777  031334  2$:     MOV     #-1,TIMER(R3)           ;STOP THE TIMER
033456  105061  031232                  CLRB    DRVACT(R1)              ;SET 'DRIVE ACTIVE" TO IDLE
033462  020137  031354                  CMP     R1,DTUW                 ;IS THIS DRIVE SETUP FOR A TRANSFER
033466  001005                          BNE     3$                      ;BR IF NOT
033470  012737  177777  031354          MOV     #-1,DTUW                ;RESET THE INDICATOR
033476  005037  031322                  CLR     TRNSWT                  ;CLEAR THE TRANSFER QUEUE
033502  004737  036166          3$:     JSR     PC,EMPTYQ               ;CLEAR THE DRIVE'S QUEUE
033506  105061  031242                  CLRB    DRVSTA(R1)              ;SET DRIVE TO OFFLINE
033512  105061  031252                  CLRB    DRVTYP(R1)              ;CLEAR THE DRIVE TYPE INDICATOR
033516  004737  036062          4$:     JSR     PC,SET.IE               ;SET "IE" WITHOUT "TRE"
033522  104413                          RESREG                          ;RESTORE R0 - R5
033524  000207                          RTS     PC                      ;RETURN
```

```
                                        ;INTERRUPT SERVICE ROUTINE
                                        
033526  112737  000001  031326  ISR:    MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
033534  104412                          SAVREG                  ;SAVE R0 - R5
033536  013704  031366                  MOV     RMADR,R4        ;ADDRESS OF RMCS1
033542  013701  031354                  MOV     DTUW,R1         ;GET "DATA TRANSFER UNDERWAY" INDICATOR
033546  002402                          BLT     1$              ;BRANCH IF NO DATA TRANSFER UNDERWAY
033550  004737  033570                  JSR     PC,TD           ;CALL TRANSFER DONE
033554  004737  034202          1$:     JSR     PC,SC           ;CALL SPECIAL CONDITIONS
033560  104413                          RESREG                  ;RESTORE R0 - R5
033562  105037  031326                  CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
033566  000002                          RTI                     ;RETURN
                                        
                                        ;TRANSFER DONE ROUTINE
                                        
033570  105061  031232          TD:     CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE INDICATOR TO IDLE
033574  012737  177777  031354          MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
033602  006301                          ASL     R1
033604  012761  177777  031334          MOV     #-1,TIMER(R1)   ;CANCEL TIMEOUT
033612  006201                          ASR     R1
033614  013702  031322                  MOV     TRNSWT,R2       ;GET "DPB" ADDRESS FROM THE
033620  005037  031322                  CLR     TRNSWT          ;TRANSFER WAIT QUEUE--CLEAR QUEUE
033624  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET DONE
033632  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
033636  004037  035552                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
033642  000000                          RMCS1                   ;REGISTER OFFSET
033644  033410                          CI8                     ;'NED' RETURN
033646  006126                          ROL     (SP)+           ;IS TRE=1 ?
033650  100413                          BMI     3$              ;BR IF YES
033652  005737  031330                  TST     SAVEFG          ;SAVE THE RH70/RM80 REGISTERS?
033656  100002                          BPL     1$              ;BRANCH IF NO
033660  004737  035744                  JSR     PC,SVRH70       ;YES--SAVE THE REGISTERS
033664  004737  032312          1$:     JSR     PC,OPT          ;CALL OPTIMIZER
033670  000207                          RTS     PC              ;RETURN
033672  012714  000113          2$:     MOV     #113,(R4)       ;RELEASE THE DRIVE
033676  000207                          RTS     PC              ;RETURN
                                        
033700                          3$:
033700  004037  035552                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
033704  000014                          RMER1                   ;REGISTER OFFSET
033706  033410                          CI8                     ;'NED' RETURN
033710  032726  000600                  BIT     #BIT8!BIT7,(SP)+  ;SEE IF HCRC OR HCE ERRORS
033714  001016                          BNE     4$              ;IF NE, YES
033716  004037  035552                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
033722  000042                          RMER2                   ;REGISTER OFFSET
033724  033410                          CI8                     ;'NED' RETURN
033726  032726  000040                  BIT     #SSE,(SP)+      ;SEE IF SKIP SECTOR ERROR
033732  001407                          BEQ     4$              ;IF EQ, NO
033734  004037  035552                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
033740  000032                          RMOF                    ;REGISTER OFFSET
033742  033410                          CI8                     ;'NED' RETURN
033744  032726  001000                  BIT     #SSEI,(SP)+     ;IS THE INHIBIT BIT ALREADY SET ?
033750  001416                          BEQ     SKIP            ;IF EQ, NO
033752  052762  100100  000016  4$:     BIS     #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
033760  004737  036166                  JSR     PC,EMPTYQ       ;EMPTY THE "DRIVE'S WAIT" QUEUE
033764  004737  035744                  JSR     PC,SVRH70       ;SAVE THE RH70/RM80 REGISTERS
033770  012714  040111                  MOV     #40111,(R4)     ;ISSUE A "DRIVE CLEAR"
```

```
     60 034000  012714  000113                    MOV     #113,(R4)       ;ISSUE A RELEASE TO THE DRIVE
     61 034004  000207                            RTS     PC              ;RETURN
     62
     63                                 ;SKIP SECTOR HANDLING ROUTINE
     64
     65 034006  010137  031354          SKIP:     MOV     R1,DTUW         ;LOAD ACTIVE DRIVE NUMBER
     66 034012  010237  031322                    MOV     R2,TRNSWT       ;RESTORE TRANSFER FLAG
     67 034016  005062  000016                    CLR     $TATUS(R2)      ;CLEAR THE DRIVER STATUS
     68 034022  132762  000002  000002           BITB    #2,$COMND(R2)   ;SEE IF HEADER ORDER
     72 034030  001405                            BEQ     1$              ;IF EQ, NO
     82 034032  016200  000004                    MOV     $WCNT(R2),R0    ;STARTING WORD COUNT
     84 034036  016203  000006                    MOV     $BUF(R2),R3     ;STARTING BUFFER ADDRESS
    101 034042  000417                            BR      2$
    102
    103 034044                         1$:
         034044  004037  035552                    JSR     R0,RD.RM        ;CALL THE READ ROUTINE
         034050  000002                            RMWC                    ;REGISTER OFFSET
         034052  033410                            CI8                     ;'NED' RETURN
    105 034054  166216  000004                    SUB     $WCNT(R2),(SP)  ;CALCULATE THE NUMBER OF WORD TRANSFERED
    106 034060  042716  000377                    BIC     #377,(SP)       ;LEAVE ONLY SECTOR MULTIPLES
    107 034064  011603                            MOV     (SP),R3         ;COPY THE DIFFERENCE
    108 034066  066216  000004                    ADD     $WCNT(R2),(SP)  ;NEW WORD COUNT
    109 034072  012600                            MOV     (SP)+,R0        ;COPY THE WORD COUNT
    110 034074  006303                            ASL     R3              ;CONVERT WORD DIFFERENCE TO A BYTE DIFFERENCE
    111 034076  066203  000006                    ADD     $BUF(R2),R3     ;NEW BUFFER ADDRESS
    115 034102  010346                 2$:       MOV     R3,-(SP)        ;BUFFER ADDRESS
    117 034104  004037  035632                    JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
         034110  000004                            RMBA                    ;REGISTER OFFSET
         034112  033410                            CI8                     ;'NED' RETURN
    118 034114  010046                            MOV     R0,-(SP)        ;WORD COUNT
    119 034116  004037  035632                    JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
         034122  000002                            RMWC                    ;REGISTER OFFSET
         034124  033410                            CI8                     ;'NED' RETURN
    120 034126  012714  040111                    MOV     #40111,(R4)     ;CLEAR THE DRIVE
    121 034132  004037  035552                    JSR     R0,RD.RM        ;CALL THE READ ROUTINE
         034136  000032                            RMOF                    ;REGISTER OFFSET
         034140  033410                            CI8                     ;'NED' RETURN
    122 034142  052716  001000                    BIS     #SSEI,(SP)      ;SET THE INHIBIT BIT
    123 034146  004037  035632                    JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
         034152  000032                            RMOF                    ;REGISTER OFFSET
         034154  033410                            CI8                     ;'NED' RETURN
    124 034156  012762  000001  000020           MOV     #1,$SSENB(R2)   ;INDICATE THAT SKIP SECTORING WAS ENABLED
    125 034164  016246  000002                    MOV     $COMND(R2),-(SP)        ;GET THE ORIGINAL COMMAND
    126 034170  004037  035632                    JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
         034174  000000                            RMCS1                   ;REGISTER OFFSET
         034176  033410                            CI8                     ;'NED' RETURN
    127 034200  000207                            RTS     PC              ;RETURN
    128
    151                                ;SPECIAL CONDITION ROUTINE
    152
    153 034202  116403  000016         SC:       MOVB    RMAS(R4),R3     ;READ "RMAS"
    154 034206  001014                            BNE     2$              ;BRANCH IF ANY 'ATA' BITS SET
    155 034210  004037  035552                    JSR     R0,RD.RM        ;CALL THE READ ROUTINE
         034214  000000                            RMCS1                   ;REGISTER OFFSET
         034216  033410                            CI8                     ;'NED' RETURN
    156 034220  106126                            ROLB    (SP)+           ;IS "IE"=1?
    157 034222  100405                            BMI     1$              ;YES, NO DRIVES TO CHECK
```

```
                                        JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
                                        EMT     1               ;REPORT AN ILLEGAL INTERRUPT
                                        JSR     PC,SET.IE       ;SET INTERRUPT ENABLE
                                1$:     RTS     PC              ;RETURN
                                2$:     CLR     -(SP)           ;PROCESS ALL DRIVES THAT HAVE
                                        MOVB    R3,(SP)         ;AN ''ATA''=1
                                        MOV     #1,R3
                                        CLR     R1

                                ;PROCESS ALL DRIVES WITH ''ATA'' SET

                                SC3:    BIT     R3,(SP)         ;ATA=1?
                                        BNE     SC5             ;YES--BRANCH
                                SC4:    INC     R1              ;MOVE TO THE NEXT DRIVE
                                        ASLB    R3
                                        BNE     SC3             ;BRANCH IF MORE TO CHECK?
                                        TST     (SP)+           ;CLEAN OFF THE STACK
                                        RTS     PC              ;RETURN TO USER

                                ;DETERMINE IF THE DRIVE WITH ''ATA'' SET IS ACTIVE WITH A COMMAND

                                SC5:    TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
                                        BEQ     1$              ;BR IF NOT
                                        JMP     SC13            ;PROCESS THE DRIVE
                                1$:     TSTB    DPRQS(R1)       ;PORT REQUEST OUTSTANDING ?
                                        BEQ     2$              ;BR IF NOT
                                        JMP     SC13            ;START THE OUTSTANDING COMMAND
                                2$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE ?
                                        BNE     SC6             ;BR IF ACTIVE

                                ;THE DRIVE WAS NOT ACTIVE, FIND OUT WHY IT INTERRUPTED

                                        JSR     PC,SC12         ;SAVE RMDS, RMER1 AND RMER2
                                                                ;ALSO DO A DRIVE INIT (DRVINT)
                                        TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE ?
                                        BNE     SC4             ;BR IF YES, CHECK ON MORE DRIVES
                                        BIT     #BIT13,RMERRS+6 ;ADDRESS PLUG CHANGED ?
                                        BNE     4$              ;BR IF YES
                                3$:     MOV     (SP),R5         ;PICKUP (RMAS) BEFORE THE ERROR CALL
                                        JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
                                        EMT     2               ;REPORT THE UNEXPECTED ATTENTION
                                        BR      SC4             ;GO CHECK FOR MORE ATA'S
                                4$:
                                        JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
                                        EMT     5               ;REPORT THE ADDRESS PLUG CHANGE
                                        BR      SC4             ;CHECK FOR MORE DRIVES

                                ;THE DRIVE COMPLETED A NON-I/O COMMAND

                                SC6:    ASL     R1              ;SETUP TO ADDRESS WORDS
                                        MOV     #-1,TIMER(R1)   ;STOP THE TIMER
                                        ASR     R1              ;RESTORE THE DRIVE ADDRESS
                                        JSR     PC,GETREQ       ;GET THE DPB POINTER FROM THE QUEUE
                                        MOV     R1,RMCS2(R4)    ;SELECT DRIVE
                                        JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                        RMDS                    ;REGISTER OFFSET
                                        SC8                     ;'NED' RETURN
```

```
                                        MOV     (SP),R5         ;AND PUT IT IN R5
                                        ROL     (SP)+           ;WAS THERE AN ERROR?
                                        BPL     SC11            ;BR IF NO ERROR
                                        JSR     R0,RD.RM        ;CALL THE READ ROUTINE
                                        RMER1                   ;REGISTER OFFSET
                                        SC8                     ;'NED' RETURN
                                        MOV     (SP)+,R5        ;AND SAVE IT IN R5
                                        JSR     PC,SVRH70       ;SAVE RH70/RM80 REGISTERS
                                        MOV     #111,-(SP)      ;ISSUE A DRIVE CLEAR
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMCS1                   ;REGISTER OFFSET
                                        SC8                     ;'NED' RETURN
                                        ROL     R5              ;WAS 'UNSAFE" CONDITION =1?
                                        BMI     1$              ;BRANCH IF YES
                                        BIS     #BIT15!BIT07!BIT05,16(R2)  ;INFORM USER OF ERROR
                                        BR      2$
1$:                                     BIS     #BIT15!BIT07!BIT04,16(R2)  ;INFORM USER OF UNSAFE ERROR
2$:                                     CLRB    DRVACT(R1)      ;SET DRIVE TO IDLE
                                        JSR     PC,EMPTYQ       ;DUMP THE QUEUE
                                        BICB    ATABIT(R1),SRCHWT  ;CLEAR THE SEARCH WAIT FLAG
                                        MOV     #113,-(SP)      ;RELEASE COMMAND
                                        JSR     R0,WRT.RM       ;CALL THE WRITE ROUTINE
                                        RMCS1                   ;REGISTER OFFSET
                                        SC8                     ;'NED' RETURN
                                        JMP     SC4             ;CHECK FOR MORE DRIVES

                                ;ISR 'NED" PROCESSOR

SC8:                            JSR     PC,EMPTYQ       ;CLEAR THE DRIVE'S QUEUE
                                JSR     PC,CI8          ;PROCESS THE 'NED'
                                JMP     SC4             ;CHECK MORE DRIVES

                                ;NON-I/O COMMAND TERMINATION ROUTINE

SC11:                           CLRB    DRVACT(R1)      ;SET DRIVE IDLE
                                BITB    ATABIT(R1),SRCHWT  ;DOING A SEARCH OPERATION FOR
                                                        ;AN I/O COMMAND?
                                BNE     1$              ;BRANCH IF YES
                                JSR     PC,POPQUE       ;REMOVE REQUEST FROM QUEUE
                                BIS     #BIT07,16(R2)   ;SET 'DONE" BIT
                                TST     SAVEFG          ;SAVE THE REGISTERS?
                                BPL     1$              ;BRANCH IF NO
                                JSR     PC,SVRH70       ;YES--SAVE ALL OF THE RH70/RM80 REG'S
1$:                             MOVB    ATABIT(R1),RMAS(R4)  ;CLEAR ATTENTION BIT
                                JSR     PC,OPT          ;START A REQUEST
                                JMP     SC4             ;CHECK FOR MORE DRIVES

                                ;ERROR PROCESSOR

SC12:                           MOV     R1,RMCS2(R4)    ;SELECT DRIVE
                                MOV     RMDS(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
                                MOV     RMER1(R4),RMERRS+2      ;WILL TELL US SOMETHING
                                MOV     RMER2(R4),RMERRS+4
                                JSR     R0,DRVINT       ;INIT. THE STATE OF THE DRIVE
                                BR      1$              ;TAKE ERROR EXIT
                                RTS     PC              ;RETURN
1$:                             TST     (SP)+           ;POP PC OFF OF THE STACK
```

```
                                        BR      SC8             ;PROCESS THE PARITY ERROR

                                ;DUAL PORT REQUEST PROCESSOR

                        SC13:   ASL     R1              ;SETUP TO ADDRESS WORDS
                                MOV     #-1,TIMER(R1)   ;STOP THE TIMER
                                ASR     R1              ;
                                MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
                                MOVB    ATABIT(R1),RMAS(R4)  ;CLEAR THE ATTENTION BIT
                                BIT     #BIT11,(R4)     ;DRIVE AVAILABLE ?
                                BNE     1$              ;BR IF AVAILABLE
                                ASL     R1
                                MOV     #15000.,TIMER(R1)  ;START 15 SECOND TIMER AGAIN
                                ASR     R1
                                BR      3$              ;EXIT
                        1$:     TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
                                BEQ     2$              ;BR IF NOT
                                CLRB    DPINT(R1)       ;CLEAR THE INIT INDICATOR
                                JSR     R0,DRVINT       ;GO INIT THE DRIVE
                                NOP                     ;DUMMY PARITY ERROR RETURN
                                TSTB    DRVSTA(R1)      ;DRIVE ONLINE ?
                                BGT     2$              ;BR IF YES -- START ORDER
                                TST     R2              ;QUEUE ENTRY FOR THE DRIVE
                                BEQ     3$              ;BR IF NOT
                                JSR     PC,GETREQ       ;GET DPB ADDRESS
                                BIS     #BIT15!BIT14,16(R2)  ;INFORM USER THAT DRIVE OFFLINE
                                JSR     PC,SVRH70       ;SAVE THE REGISTERS
                                JSR     PC,EMPTYQ       ;EMPTY THE REQUEST QUEUE
                                BR      3$
                        2$:     CLRB    DPRQS(R1)       ;CLEAR THE PORT REQUEST INDICATOR
                                JSR     PC,OPT          ;START THE PENDING REQUEST
                        3$:     JMP     SC4             ;PROCESS OTHER DRIVES
```

```
                                        ;RM80 TIMER ROUTINE
                                        ;CALL
                                        ;       MOV     #TIME,-(SP)     ;ELASPED TIME IN MILLISECONDS ON THE STACK
                                        ;       JSR     PC,RMTMR        ;CALL RM80 TIME ROUTINE

035034  005737  031326                  RMTMR:  TST     ACTDRV          ;CHECK "ACTDRV & ACTSTR"
035040  001030                          BNE     4$              ;IF NON ZERO EXIT
035042  112737  000001  031327          MOVB    #1,ACTSTR       ;SET "ACTSTR"
035050  104412                          SAVREG                  ;SAVE R0 - R5
035052  005001                          CLR     R1              ;START WITH DRIVE 0
035054  005003                          CLR     R3
035056  005763  031334                  1$:     TST     TIMER(R3)       ;IS THE TIMER RUNNING?
035062  002407                          BLT     2$              ;BRANCH IF NO
035064  166663  000002  031334          SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
035072  003003                          BGT     2$              ;BR IF NO SOFTWARE TIMEOUT
035074  004737  035126                  JSR     PC,STO          ;CALL SOFTWARE TIMEOUT ROUTINE
035100  000405                          BR      3$              ;GO TO THE EXIT
035102  005201                  2$:     INC     R1              ;MOVE TO NEXT DRIVE
035104  005723                          TST     (R3)+
035106  022701  000010                  CMP     #8.,R1          ;OUT OF DRIVES?
035112  003361                          BGT     1$              ;BRANCH IF NO
035114  104413                  3$:     RESREG                  ;RESTORE R0 - R5
035116  105037  031327                  CLRB    ACTSTR          ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
035122  012616                  4$:     MOV     (SP)+,(SP)      ;ADJUST THE STACK
035124  000207                          RTS     PC              ;RETURN

                                        ;SOFTWARE TIMEOUT ROUTINE
                                        ;
                                        ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
                                        ;       OR GREATER
                                        ;
                                        ;CALL:  STO
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                        ;       JSR     PC,STO          ;CALL
                                        ;       RETURN

035126  010146                  STO:    MOV     R1,-(SP)        ;SAVE R1
035130  010346                          MOV     R3,-(SP)        ;SAVE R3
035132  013704  031366                  MOV     RMADR,R4        ;GET ADDRESS OF "RMCS1"
035136  010164  000010                  MOV     R1,RMCS2(R4)    ;SELECT THE DRIVE
035142  004037  035552                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
035146  000012                          RMDS                    ;REGISTER OFFSET
035150  035440                          STO5                    ;'NED' RETURN
035152  105726                          TSTB    (SP)+           ;IS "DRY"=1?
035154  100473                          BMI     STO2            ;BR IF YES
035156  105761  031262          STO1:   TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
035162  001070                          BNE     STO2            ;BR IF YES
035164  105761  031272                  TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
035170  001065                          BNE     STO2            ;BR IF YES
035172  013702  031322                  MOV     TRNSWT,R2       ;PICKUP TRANSFER WAIT QUEUE
035176  020137  031354                  CMP     R1,DTUW         ;TRANSFER UNDERWAY ON THIS DRIVE?
035202  001402                          BEQ     1$              ;BRANCH IF YES
035204  004737  036222                  JSR     PC,GETREQ       ;GET DPB ADDRESS
035210  052762  101000  000016  1$:     BIS     #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
035216  004737  035744                  JSR     PC,SVRH70       ;SAVE RH70/RM80 REGISTERS
035222  012764  000040  000010          MOV     #BIT05,RMCS2(R4) ;"INIT" THE MASS BUS
035230  105061  031232                  CLRB    DRVACT(R1)      ;DRIVE IS IDLE
```

```
 56 035234  005001                          CLR     R1              ;START WITH DRIVE 0
 57 035236  005003                          CLR     R3
 58 035240  004037  031610          2$:     JSR     R0,DRVINT       ;INIT. THIS DRIVE
 59 035244  000475                          BR      ST05            ;PARITY ERROR RETURN
 60 035246  105761  031232                  TSTB    DRVACT(R1)      ;DRIVE IDLE BEFORE THE INIT.?
 61 035252  001414                          BEQ     4$              ;YES--BRANCH
 62 035254  013702  031322                  MOV     TRNSWT,R2       ;GET TRANSFER WAIT QUEUE
 63 035260  023701  031354                  CMP     DTUW,R1         ;WAS THERE I/O ON THIS DRIVE?
 64 035264  001402                          BEQ     3$              ;YES--BRANCH
 65 035266  004737  036222                  JSR     PC,GETREQ       ;GET THE DPB POINTER FROM QUEUE
 66 035272  052762  100400  000016  3$:     BIS     #BIT15!BIT08,16(R2) ;INFORM USER OF INIT.
 67 035300  105061  031232                  CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE TO IDLE
 68 035304  012763  177777  031334  4$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
 69 035312  005723                          TST     (R3)+           ;UPDATE THE INDEX
 70 035314  005201                          INC     R1              ;INCREMENT THE DRIVE NUMBER
 71 035316  022701  000010                  CMP     #8.,R1          ;LAST DRIVE BEEN CHECKED?
 72 035322  003346                          BGT     2$              ;NO--LOOP
 73 035324  012737  177777  031354          MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
 74 035332  005037  031322                  CLR     TRNSWT          ;CLEAR TRANSFER WAIT QUEUE
 75 035336  004737  036134                  JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
 76 035342  000500                          BR      ST09            ;EXIT
 77 035344  116405  000016          ST02:   MOVB    RMAS(R4),R5     ;READ ATTENTION REG
 78 035350  136105  031356                  BITB    ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
 79 035354  001017                          BNE     ST03            ;YES--BRANCH
 80 035356  105761  031262                  TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
 81 035362  001031                          BNE     ST06            ;BR IF YES - DRIVE NOT ONLINE
 82 035364  105761  031272                  TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
 83 035370  001045                          BNE     ST07            ;BR IF YES - NO RESPONSE TO REQUEST
 84 035372  020137  031354                  CMP     R1,DTUW         ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
 85 035376  001267                          BNE     ST01            ;BR IF NO
 86 035400  004037  035552                  JSR     R0,RD.RM        ;CALL THE READ ROUTINE
    035404  000000                          RMCS1                   ;REGISTER OFFSET
    035406  035440                          ST05                    ;'NED' RETURN
 87 035410  105726                          TSTB    (SP)+
 88 035412  100261                          BPL     ST01            ;BR IF 'RDY'=0
 89 035414  105761  031262          ST03:   TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
 90 035420  001003                          BNE     1$              ;BR IF INIT PENDING
 91 035422  105761  031272                  TSTB    DPRQS(R1)       ;PORT REQUEST PENDING ?
 92 035426  001446                          BEQ     ST09            ;BR IF NOT
 93 035430  012763  177777  031334  1$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
 94 035436  000442                          BR      ST09            ;EXIT
 95 035440  004737  033410          ST05:   JSR     PC,CI8          ;GO HANDLE THE 'NED'
 96 035444  000437                          BR      ST09
 97 035446  105061  031262          ST06:   CLRB    DPINT(R1)       ;CLEAR THE INITIALIZE INDICATOR
 98 035452  105061  031242                  CLRB    DRVSTA(R1)      ;SET UNIT OFFLINE
 99 035456  012763  177777  031334          MOV     #-1,TIMER(R3)   ;STOP THE TIMER
100 035464  004737  036222                  JSR     PC,GETREQ       ;GET THE DPB ADDRESS
101 035470  005702                          TST     R2              ;REQUEST IN QUEUE ?
102 035472  001424                          BEQ     ST09            ;BR IF NOT
103 035474  052762  140000  000016          BIS     #BIT15!BIT14,16(R2)  ;INFORM THE USER DRIVE NOT AVAILABLE
104 035502  000414                          BR      ST08            ;FINISH
105 035504  012763  177777  031334  ST07:   MOV     #-1,TIMER(R3)   ;STOP THE TIMER
106 035512  105061  031272                  CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST INDICATOR
107 035516  004737  036222                  JSR     PC,GETREQ       ;GET DPB ADDRESS
108 035522  005702                          TST     R2              ;QUEUE ENTRY FOR DRIVE ?
109 035524  001407                          BEQ     ST09            ;BR IF NONE
110 035526  012762  100004  000016          MOV     #BIT15!BIT2,16(R2)  ;INFORM USER OF PORT REQUEST ERROR
```

```
    111 035534  004737  036166          ST08:   JSR     PC,EMPTYQ       ;CLEAR THE QUEUE FOR THE DRIVE
    112 035540  004737  035744                  JSR     PC,SVRH70       ;SAVE THE REGISTERS
    113 035544  012603                  ST09:   MOV     (SP)+,R3        ;RESTORE R3
    114 035546  012601                          MOV     (SP)+,R1        ;RESTORE R1
    115 035550  000207                          RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO READ A RH70/RM80 REGISTER
                                        ;
                                        ;CALL
                                        ;       JSR     R0,RD.RM        ;GO READ A REGISTER
                                        ;       INDEX                   ;REG. INDEX FROM BASE
                                        ;       ERRADR                  ;ERROR ADDRESS--PROCESS ERROR STARTING
                                        ;                               ;AT THIS ADDRESS
                                        ;       RETURN                  ;CONTENTS OF REG. IS ON THE STACK
                                        ;
035552  011646                  RD.RM:  MOV     (SP),-(SP)      ;SAVE R0 FOR RETURN
035554  013737  031366  035570          MOV     RMADR,RD.ADR    ;FORM THE DESIRED ADDRESS
035562  062037  035570                  ADD     (R0)+,RD.ADR    ;USING THE BASE AND THE INDEX
035566  013727                  RD.RM1: MOV     @(PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RM80
035570  000000                  RD.ADR: .WORD   0               ;ADDRESS IS FORMED HERE
035572  000000                  RD.WRD: .WORD   0               ;REG. CONTENTS PUT HERE
035574  013766  035572  000002        MOV     RD.WRD,2(SP)    ;RETURN IT TO THE USER
035602  013746  031366                MOV     RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
035606  062716  000010                ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
035612  032736  010000                BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
035616  001002                        BNE     RD.RM3          ;IF NE, DRIVE NOT PRESENT
035620  005720                        TST     (R0)+           ;ERROR FREE RETURN
035622  000402                        BR      RD.RM4          ;EXIT
035624  011000                  RD.RM3: MOV     (R0),R0         ;ERROR EXIT
035626  012616                        MOV     (SP)+,(SP)
035630  000200                  RD.RM4: RTS     R0

                                        ;ROUTINE TO WRITE A REGISTER
                                        ;
                                        ;CALL
                                        ;       MOV     DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
                                        ;       JSR     R0,WRT.RM       ;CALL THE ROUTINE TO LOAD (WRITE) THE REG.
                                        ;       INDEX                   ;INDEX OF THE REGISTER TO BE LOADED
                                        ;       'NED' RETURN            ;ADDRESS TO RETURN TO IF 'NED' ERROR
                                        ;       RETURN                  ;ERROR FREE RETURN
                                        ;
035632  016637  000002  035712  WRT.RM: MOV     2(SP),WRT.WD    ;SAVE THE WORD TO WRITE
035640  012616                        MOV     (SP)+,(SP)      ;ADJUST THE STACK
035642  012037  035714                MOV     (R0)+,WRT.AD    ;GET INDEX OF REGISTER TO BE WRITTEN
035646  001015                        BNE     1$              ;BRANCH IF NOT RMCS1
035650  122737  000150  035712        CMPB    #150,WRT.WD     ;IS THE COMMAND FOR DATA TRANSFERS?
035656  002411                        BLT     1$              ;YES--DON'T GET THE OLD A16 & A17, & PSEL
035660  004037  035552                JSR     R0,RD.RM        ;CALL THE READ ROUTINE
035664  000000                        RMCS1                   ;REGISTER OFFSET
035666  035734                        WRT.R3                  ;'NED' RETURN
                                                              ;THE COMMAND BEFORE SENDING TO THE
                                                              ;THE RH70/RM80
035670  000316                        SWAB    (SP)
035672  042716  177770                BIC     #^C7,(SP)
035676  112637  035713                MOVB    (SP)+,WRT.WD+1
035702  063737  031366  035714  1$:   ADD     RMADR,WRT.AD    ;FORM THE ADDRESS OF THE DISK REG.
035710  012737                  WRT.R1: MOV     (PC)+,@(PC)+    ;LOAD THE DESIRED REG.
035712  000000                  WRT.WD: .WORD   0               ;WORD TO WRITE GOES HERE
035714  000000                  WRT.AD: .WORD   0               ;ADDRESS IS FORMED HERE
035716  013746  031366                MOV     RMADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
035722  062716  000010                ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
035726  032736  010000                BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
035732  001402                        BEQ     WRT.R4          ;IF EQ, DRIVE IS PRESENT
```

```
                                WRT.R3: MOV     (R0),R0         ;TAKE THE 'NED' EXIT
                                        BR      WRT.R5          ;EXIT
                                WRT.R4: TST     (R0)+           ;ADJUST FOR ERROR FREE EXIT
                                WRT.R5: RTS     R0

                                ;ROUTINE TO SAVE THE RH70/RM80 REGISTERS
                                ;
                                ;CALL
                                ;       MOV     #DPBADR,R2      ;DPB POINTER TO R2
                                ;       JSR     PC,SVRH70       ;SAVE THE DRIVES REG'S

                                SVRH70: SAVREG                  ;SAVE R0 - R5
                                        TST     R2              ;QUEUE ENTRY FOR THE DRIVE ?
                                        BEQ     6$              ;BR IF NONE
                                        MOV     RMADR,R4
                                        MOVB    (R2),RMCS2(R4)  ;SELECT DRIVE
                                        MOV     14(R2),R3       ;GET THE ERROR TABLE POINTER
                                        BEQ     6$              ;EXIT IF NO ADDRESS
                                        CLR     3$              ;COUNTER & POINTER
                                1$:     CMP     3$,#RMDB        ;REACHED THE BUFFER REGISTER ?
                                        BNE     2$              ;BR IF NOT
                                        BIT     #BIT07,RMCS2(R4)  ;'OR' SET ?
                                        BNE     2$              ;BR IF SET
                                        CLR     (R3)+           ;STORE RMDB AS ZEROES
                                        BR      4$              ;CONTINUE
                                2$:     JSR     R0,RD.RM        ;READ THE SELECTED REGISTER
                                3$:     .WORD   0               ;REGISTER INDEX
                                        5$                      ;ERROR RETURN ADDRESS
                                        MOV     (SP)+,(R3)+     ;STORE THE REGISTER CONTENTS
                                4$:     CMP     3$,#RMEC2       ;REACHED THE END ?
                                        BEQ     6$              ;BR IF YES
                                        ADD     #2,3$           ;INCREMENT THE REGISTER INDEX
                                        BR      1$              ;CONTINUE READING THE REGISTERS
                                5$:     JSR     PC,CI8          ;PROCESS THE 'NED'
                                6$:     RESREG                  ;RESTORE R0 - R5
                                        RTS     PC              ;RETURN

                                ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
                                ;CALL
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                ;       JSR     PC,SET.IE       ;SET "IE"
                                ;       RETURN

                                SET.IE: MOV     R4,-(SP)        ;SAVE R4
                                        MOV     RMADR,R4        ;PICKUP ADDRESS OF RMCS1
                                        MOV     R1,RMCS2(R4)    ;SELECT DRIVE
                                        MOV     (R4),-(SP)      ;READ RMCS1
                                        BIS     #BIT14,(SP)     ;SET THE "TRE" BIT OF THE WORD READ
                                        SWAB    (SP)            ;ADJUST FOR DATO
                                        MOVB    #BIT06,(R4)     ;SET "IE"
                                        BIT     #BIT12,RMCS2(R4) ;IS "NED"=1?
                                        BNE     1$              ;YES--CLEAR "TRE"
                                        TST     (SP)+           ;CLEAN OFF THE STACK
                                        BR      2$
                                1$:     MOVB    (SP)+,1(R4)     ;CLEAR "TRE"
                                2$:     MOV     (SP)+,R4        ;RESTORE R4
                                        RTS     PC              ;RETURN TO CALLER
```

```
                                        ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
                                        ;
                                        ;CALL
                                        ;       JSR     PC,CLRQUE
                                        ;
036134  104412                  CLRQUE: SAVREG                          ;SAVE R0 - R5
036136  012702  031302                  MOV     #QDRV,R2                ;QUEUE BASE ADDRESS
036142  005022                          CLR     (R2)+                   ;CLEAR ENTRY
036144  005022                          CLR     (R2)+                   ;CLEAR ENTRY
036146  005022                          CLR     (R2)+                   ;CLEAR ENTRY
036150  005022                          CLR     (R2)+                   ;CLEAR ENTRY
036152  005022                          CLR     (R2)+                   ;CLEAR ENTRY
036154  005022                          CLR     (R2)+                   ;CLEAR ENTRY
036156  005022                          CLR     (R2)+                   ;CLEAR ENTRY
036160  005022                          CLR     (R2)+                   ;CLEAR ENTRY
036162  104413                          RESREG                          ;RESTORE R0 - R5
036164  000207                          RTS     PC
                                        ;EMPTY THE QUEUE SPECIFIED BY R1
                                        ;
                                        ;CALL
                                        ;       MOV     DRVNUM,R1       ;DRIVE NUMBER TO R1
                                        ;       JSR     PC,EMPTYQ
                                        ;
036166  006301                  EMPTYQ: ASL     R1
036170  005061  031302                  CLR     QDRV(R1)                ;CLEAR DRIVE QUEUE
036174  006201                          ASR     R1                      ;RESTORE R1
036176  000207                          RTS     PC
                                        ;ROUTINE TO PUT A REQUEST IN QUEUE
                                        ;
                                        ;CALL
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                        ;       MOV     #DPB,R2         ;ADDRESS OF PARAMETER BLOCK
                                        ;       JSR     R0,DRVQUE       ;GO PUT REQUEST IN QUEUE
                                        ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
                                        ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE
                                        ;
036200  006301                  DRVQUE: ASL     R1
036202  005761  031302                  TST     QDRV(R1)                ;TEST THE QUEUE ENTRY
036206  001003                          BNE     1$                      ;IF NE, QUEUE ENTRY ALREADY THERE
036210  010261  031302                  MOV     R2,QDRV(R1)             ;ADD THE QUEUE ENTRY
036214  005720                          TST     (R0)+                   ;TAKE RETURN 2
036216  006201                  1$:     ASR     R1
036220  000200                  2$:     RTS     R0                      ;RETURN TO USER
                                        ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
                                        ;
                                        ;CALL
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                        ;       JSR     PC,GETREQ       ;GO GET THE REQUEST
                                        ;       RETURN                  ;R2="DPB" ADDRESS OF THE REQUEST
                                        ;                               ;R2=0 IF NO REQUEST IN QUEUE
                                        ;
036222  006301                  GETREQ: ASL     R1                      ;
036224  016102  031302                  MOV     QDRV(R1),R2             ;GET THE REQUEST
```

```
    180 036230  006201                          ASR     R1              ;
    181 036232  000207                          RTS     PC              ;RETURN
    182
    183                                 ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
    184                                 ;
    185                                 ;CALL
    186                                 ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
    187                                 ;       JSR     PC,POPQUE       ;CALL TO REMOVE REQUEST
    188                                 ;       RETURN                  ;R2=ADDRESS OF DPB REMOVED
    189
    190 036234  006301                  POPQUE: ASL     R1
    191 036236  016102  031302                  MOV     QDRV(R1),R2     ;GET THE QUEUE ENTRY
    192 036242  005061  031302                  CLR     QDRV(R1)        ;CLEAR THE QUEUE
    193 036246  006201                          ASR     R1
    194 036250  000207                          RTS     PC              ;RETURN TO USER
    195
    197                                 ;ROUTINE TO SAVE THE CONTENTS OF '$ESCAPE' WHEN THE DRIVER
    198                                 ;REPORTS AN ERROR DIRECTLY.
    199                                 ;
    200                                 ;CALL
    201                                 ;       JSR     R0,ES.SAV
    202                                 ;       ERROR   N               ;;THE ERROR CALL
    203                                 ;       RETURN                  ;THE RETURN IS PAST THE ERROR CALL
    204
    205 036252  012037  036266          ES.SAV: MOV     (R0)+,1$        ;GET THE ERROR CALL
    206 036256  013746  001222                  MOV     $ESCAPE,-(SP)   ;SAVE THE ADDRESS IN '$ESCAPE'
    207 036262  005037  001222                  CLR     $ESCAPE         ;CLEAR THE ESCAPE RETURN
    208 036266  000000                  1$:     .WORD   0               ;THE ERROR CALL IS MOVED HERE
    209 036270  012637  001222                  MOV     (SP)+,$ESCAPE   ;RESTORE THE ESCAPE ADDRESS
    210 036274  000200                          RTS     R0              ;RETURN
```

```
                                        .SBTTL  GETADR - GET BUS ADDRESS AND VECTOR ADDRESS
                                        ;THIS ROUTINE IS USED TO ENSURE THE BUS ADDRESS OF THE RH/RM
                                        ;IS SETUP TO READ THE PROPER VALUE. IT WILL ALSO READ THE
                                        ;ADDRESS FROM THE TTY IF REQUIRED.
                                        ;
                                        ;NOTE: THIS ROUTINE DESTROYS R0-R4
                                        ;CALL
                                        ;
                                        ;       JSR     PC,GETADR
                                        ;       RETURN
                                        
036276  005737  001324        GETADR: TST     BUSADR          ;INPUT FROM TTY REQUESTED?
036302  001436                        BEQ     5$              ;NO--BRANCH
036304  005037  001324                CLR     BUSADR          ;YES--CLEAR THE REQUEST FLAG
036310  104401  001231                TYPE    ,$CRLF          ;CR-LF
036314  005227  177777                INC     #-1             ;FIRST TIME THRU ?
036320  001402                        BEQ     1$              ;BR IF YES
036322  104401  001231                TYPE    ,$CRLF          ;CR-LF
036326  012700  001522        1$:     MOV     #RH.ADR,R0      ;FIRST ADDRESS
036332  012703  036466                MOV     #MRMCS1,R3      ;'RMCS1='
036336  011004                        MOV     (R0),R4         ;PRESENT RMCS1 ADDRESS
036340  004037  026750                JSR     R0,GETNUM       ;GET NEW RMCS1
036344  000402                        BR      2$              ;COMMA
036346  000767                        BR      1$              ;PERIOD
036350  000412                        BR      4$              ;DOUBLE PERIOD
036352  010420        2$:     MOV     R4,(R0)+        ;SAVE NEW RMCS1
036354  012703  036476                MOV     #MRMVEC,R3      ;'RMVEC='
036360  011004                        MOV     (R0),R4         ;PRESENT RH/RM VECTOR ADDRESS
036362  004037  026750                JSR     R0,GETNUM       ;GET NEW RMVEC
036366  000402                        BR      3$              ;COMMA
036370  000756                        BR      1$              ;PERIOD
036372  000401                        BR      4$              ;DOUBLE PERIOD
036374  010420                3$:     MOV     R4,(R0)+        ;SAVE NEW RMVEC
036376  010410                4$:     MOV     R4,(R0)         ;SAVE INPUT
036400  013701  000004        5$:     MOV     ERRVEC,R1       ;SAVE THE ERROR VECTOR
036404  012737  036440  000004        MOV     #6$,ERRVEC      ;SETUP FOR TRAP
036412  005777  143104                TST     @RH.ADR         ;CHECK FOR RH/RM
036416  010137  000004                MOV     R1,ERRVEC       ;RESTORE ERROR VECTOR
036422  012700  001522                MOV     #RH.ADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
036426  012701  031366                MOV     #RMADR,R1       ;FIRST ADDRESS OF WHERE TO PUT THEM
036432  012021                        MOV     (R0)+,(R1)+     ;BUS ADDRESS
036434  012021                        MOV     (R0)+,(R1)+     ;VECTOR ADDRESS
036436  000207                        RTS     PC              ;RETURN
036440  010137  000004        6$:     MOV     R1,ERRVEC       ;RESTORE ERROR VECTOR
036444  022626                        CMP     (SP)+,(SP)+     ;CLEAN OFF THE STACK
036446  104010                        EMT     10
036450  005737  000042                TST     @#42            ;IS THERE A MONITOR?
036454  001724                        BEQ     1$              ;NO--GO ASK FOR ADDRESS
036456  005037  001330                CLR     DRVSEL          ;DESELECT ALL DRIVES FROM TEST
036462  000137  015142                JMP     $EOP            ;RETURN TO $EOP

036466    200     122     115   MRMCS1: .ASCIZ  <CRLF>/RMCS1=/
036476    200     122     115   MRMVEC: .ASCIZ  <CRLF>/RMVEC=/
```

```
                                .SBTTL  DPB (DATA PARAMETER BLOCKS)

                        DPB.A:  .BYTE   0               ;(0) DRIVE NUMBER
                                .BYTE   0               ;(1) OFFSET VALUE OR FMT16, ECI, HCI, & SSEI
                                .BYTE   0               ;(2) COMMAND
                                .BYTE   0               ;(3) PSEL AND A17 AND A16
                                .WORD   0               ;(4) WORD COUNT (MUST BE NEG.)
                                .WORD   BUFFER          ;(6) BUFFER ADDRESS OR
                                                        ;REGISTER TABLE POINTER
                                .BYTE   0               ;(10) SECTOR ADDRESS OR
                                                        ;FIRST REG. INDEX
                                .BYTE   0               ;(11) TRACK ADDRESS OR
                                                        ;LAST REG. INDEX
                                .WORD   0               ;(12) CYLINDER ADDRESS
                                .WORD   RM.REG          ;(14) ERROR TABLE POINTER
                                                        ;POINTS TO THE FIRST OF TWENTY
                                                        ;LOCATIONS OF WHERE THE DRIVER
                                                        ;IS TO STORE THE RH/RM
                                                        ;REGISTERS ON AN ERROR. IF LEFT
                                                        ;ZERO REGISTERS ARE NOT SAVED.
                                .WORD   0               ;(16) STATUS/ERROR INDICATOR
                                                        ;BIT15=1=>ERROR OCCURRED
                                                        ;BIT07=1=>DONE
                                                        ;BIT14-BIT09 AND BIT06-BIT03
                                                        ;INDICATE TYPE OF ERROR
                                .WORD   0               ;SKIP SECTOR ENABLE INDICATOR (ENABLED =1)

                        DPB.B:  .BYTE   0               ;(0) DRIVE NUMBER
                                .BYTE   0               ;(1) OFFSET VALUE OR FMT16, ECI, HCI, & SSEI
                                .BYTE   0               ;(2) COMMAND
                                .BYTE   0               ;(3) PSEL AND A17 AND A16
                                .WORD   -2              ;(4) WORD COUNT (MUST BE NEG.)
                                .WORD   BUFFER          ;(6) BUFFER ADDRESS OR
                                                        ;REGISTER TABLE POINTER
                                .BYTE   0               ;(10) SECTOR ADDRESS OR
                                                        ;FIRST REG. INDEX
                                .BYTE   0               ;(11) TRACK ADDRESS OR
                                                        ;LAST REG. INDEX
                                .WORD   0               ;(12) CYLINDER ADDRESS
                                .WORD   RM.REG          ;(14) ERROR TABLE POINTER
                                                        ;POINTS TO THE FIRST OF TWENTY
                                                        ;LOCATIONS OF WHERE THE DRIVER
                                                        ;IS TO STORE THE RH/RM
                                                        ;REGISTERS ON AN ERROR. IF LEFT
                                                        ;ZERO REGISTERS ARE NOT SAVED.
                                .WORD   0               ;(16) STATUS/ERROR INDICATOR
                                                        ;BIT15=1=>ERROR OCCURRED
                                                        ;BIT07=1=>DONE
                                                        ;BIT14-BIT09 AND BIT06-BIT03
                                                        ;INDICATE TYPE OF ERROR
                                .WORD   0               ;SKIP SECTOR ENABLE INDICATOR (ENABLED =1)

                        DPB.C:  .BYTE   0               ;(0) DRIVE NUMBER
                                .BYTE   0               ;(1) OFFSET VALUE OR FMT16, ECI, HCI, & SSEI
                                .BYTE   0               ;(2) COMMAND
                                .BYTE   0               ;(3) PSEL AND A17 AND A16
                                .WORD   -2              ;(4) WORD COUNT (MUST BE NEG.)
```

DPB (DATA PARAMETER BLOCKS)

```
                                        .WORD   BUFFER          ;(6) BUFFER ADDRESS OR
                                                                ;REGISTER TABLE POINTER
                                        .BYTE   0               ;(10) SECTOR ADDRESS OR
                                                                ;FIRST REG. INDEX
                                        .BYTE   0               ;(11) TRACK ADDRESS OR
                                                                ;LAST REG. INDEX
                                        .WORD   0               ;(12) CYLINDER ADDRESS
                                        .WORD   RM.REG          ;(14) ERROR TABLE POINTER
                                                                ;POINTS TO THE FIRST OF TWENTY
                                                                ;LOCATIONS OF WHERE THE DRIVER
                                                                ;IS TO STORE THE RH/RM
                                                                ;REGISTERS ON AN ERROR. IF LEFT
                                                                ;ZERO REGISTERS ARE NOT SAVED.
                                        .WORD   0               ;(16) STATUS/ERROR INDICATOR
                                                                ;BIT15=1=>ERROR OCCURRED
                                                                ;BIT07=1=>DONE
                                                                ;BIT14-BIT09 AND BIT06-BIT03
                                                                ;INDICATE TYPE OF ERROR
                                        .WORD   0               ;SKIP SECTOR ENABLE INDICATOR (ENABLED =1)

DTADPB:                 .BYTE   0               ;(0) DRIVE NUMBER
                        .BYTE   0               ;(1) OFFSET VALUE OR FMT16, ECI, HCI, & SSEI
                        .BYTE   0               ;(2) COMMAND
                        .BYTE   0               ;(3) PSEL AND A17 AND A16
                        .WORD   0               ;(4) WORD COUNT (MUST BE NEG.)
                        .WORD   BUFFER          ;(6) BUFFER ADDRESS OR
                                                ;REGISTER TABLE POINTER
                        .BYTE   0               ;(10) SECTOR ADDRESS OR
                                                ;FIRST REG. INDEX
                        .BYTE   0               ;(11) TRACK ADDRESS OR
                                                ;LAST REG. INDEX
                        .WORD   0               ;(12) CYLINDER ADDRESS
                        .WORD   RM.REG          ;(14) ERROR TABLE POINTER
                                                ;POINTS TO THE FIRST OF TWENTY
                                                ;LOCATIONS OF WHERE THE DRIVER
                                                ;IS TO STORE THE RH/RM
                                                ;REGISTERS ON AN ERROR. IF LEFT
                                                ;ZERO REGISTERS ARE NOT SAVED.
                        .WORD   0               ;(16) STATUS/ERROR INDICATOR
                                                ;BIT15=1=>ERROR OCCURRED
                                                ;BIT07=1=>DONE
                                                ;BIT14-BIT09 AND BIT06-BIT03
                                                ;INDICATE TYPE OF ERROR
                        .WORD   0               ;SKIP SECTOR ENABLE INDICATOR (ENABLED =1)

DPB.R:                  .BYTE   0               ;(0) DRIVE NUMBER
                        .BYTE   0               ;(1) OFFSET VALUE OR FMT16, ECI, HCI, & SSEI
                        .BYTE   RECAL           ;(2) COMMAND
                        .BYTE   0               ;(3) PSEL AND A17 AND A16
                        .WORD   0               ;(4) WORD COUNT (MUST BE NEG.)
                        .WORD   BUFFER          ;(6) BUFFER ADDRESS OR
                                                ;REGISTER TABLE POINTER
                        .BYTE   0               ;(10) SECTOR ADDRESS OR
                                                ;FIRST REG. INDEX
                        .BYTE   0               ;(11) TRACK ADDRESS OR
                                                ;LAST REG. INDEX
                        .WORD   0               ;(12) CYLINDER ADDRESS
```

DPB (DATA PARAMETER BLOCKS)

```
036632  036640                  .WORD   RM.REG          ;(14) ERROR TABLE POINTER
                                                       ;POINTS TO THE FIRST OF TWENTY
                                                       ;LOCATIONS OF WHERE THE DRIVER
                                                       ;IS TO STORE THE RH/RM
                                                       ;REGISTERS ON AN ERROR. IF LEFT
                                                       ;ZERO REGISTERS ARE NOT SAVED.
036634  000000                  .WORD   0              ;(16) STATUS/ERROR INDICATOR
                                                       ;BIT15=1=>ERROR OCCURRED
                                                       ;BIT07=1=>DONE
                                                       ;BIT14-BIT09 AND BIT06-BIT03
                                                       ;INDICATE TYPE OF ERROR
036636  000000                  .WORD   0              ;SKIP SECTOR ENABLE INDICATOR (ENABLED =1)

                        ;SAVE RH/RM REGISTERS HERE ON ERROR

036640  000000          RM.REG: .WORD   0              ;RMCS1 (776700) CONTROL & STATUS #1
036642  000000                  .WORD   0              ;RMWC (776702) WORD COUNT
036644  000000                  .WORD   0              ;RMBA (776704) BUS ADDRESS
036646  000000                  .WORD   0              ;RMDA (776706) DESIRED SECTOR/TRACK
036650  000000                  .WORD   0              ;RMCS2 (776710) CONTROL & STATUS #2
036652  000000                  .WORD   0              ;RMDS (776712) DISK STATUS
036654  000000                  .WORD   0              ;RMER1 (776714) ERROR REG. #1
036656  000000                  .WORD   0              ;RMAS (776716) ATTENTION SUMMARY
036660  000000                  .WORD   0              ;RMLA (776720) LOOK AHEAD
036662  000000                  .WORD   0              ;RMDB (776722) DATA BUFFER
036664  000000                  .WORD   0              ;RMMR1 (776724) MAINTAINABILITY
036666  000000                  .WORD   0              ;RMDT (776726) DRIVE TYPE
036670  000000                  .WORD   0              ;RMSN (776730) SERIAL NUMBER
036672  000000                  .WORD   0              ;RMOF (776732) OFFSET
036674  000000                  .WORD   0              ;RMDC (776734) DESIRED CYLINDER
036676  000000                  .WORD   0              ;RMHR (776736) CURRENT CYLINDER
036700  000000                  .WORD   0              ;RMMR2 (776740) ERROR REG #2
036702  000000                  .WORD   0              ;RMER2 (776742) ERROR REG #3
036704  000000                  .WORD   0              ;RMEC1 (776744) ECC POSITION
036706  000000                  .WORD   0              ;RMEC2 (776746) ECC PATTERN
                        .EVEN
```

```
                                        .SBTTL  ASCIZ MESSAGES

                                MSG.R:  .ASCIZ  /R/
036712  106     103     000     MSG.FC: .ASCIZ  /FC/
036715  114     103     000     MSG.LC: .ASCIZ  /LC/
036720  111     103     000     MSG.IC: .ASCIZ  /IC/
036723  106     124     000     MSG.FT: .ASCIZ  /FT/
036726  114     124     000     MSG.LT: .ASCIZ  /LT/
036731  111     124     000     MSG.IT: .ASCIZ  /IT/
036734  106     123     000     MSG.FS: .ASCIZ  /FS/
036737  114     123     000     MSG.LS: .ASCIZ  /LS/
036742  120     101     124     MSG.PAT:.ASCIZ  /PAT/
036746  075     000             MSG.EQ: .ASCIZ  /=/
036750  200     103     117     MSG.CS: .ASCIZ  <CRLF>/CONTROL SWITCHES=/

036773  040     057     040     SLASH:  .ASCIZ  @ / @
037000  200     125     116     UNSTAT: .ASCIZ  <CRLF>/UNIT STATUS:/
037015  040     117     106     UNTOFF: .ASCIZ  / OFFLINE/
037026  040     117     116     UNTON:  .ASCIZ  / ONLINE/
037036  040     116     117     NOTPRS: .ASCIZ  / NOT PRESENT/
037053  040     125     116     NOTSAF: .ASCIZ  / UNSAFE/
037063  040     116     117     NOTRM:  .ASCIZ  @ NOT AN RM80@
037100  040     114     117     LODEV:  .ASCIZ  / LOAD DEVICE/
037115  122     115     070     $RM80:  .ASCIZ  /RM80/
037122  200     104     122     DRIVES: .ASCIZ  <CRLF>/DRIVE(S) TO BE TESTED, /
037153  116     117     116     NONE:   .ASCIZ  /NONE/
037160  054     040     000     COMMA:  .ASCIZ  /, /
037163  072     040     000     COLON:  .ASCIZ  /: /
037166  200     116     117     NOCLOK: .ASCIZ  <CRLF>/NO KW11-P CLOCK, TIMING TESTS WILL NOT BE PERFORMED/
037253  115     102     101     SERIAL: .ASCIZ  @MBA S/N: @
037265  200     124     105     MSGTST: .ASCIZ  <CRLF>/TEST/
037273  200     104     122     MSDRIV: .ASCIZ  <CRLF>/DRIVE/
037302  040     104     122     DROP:   .ASCIZ  / DROPPED/
037313  105     130     103     EXCEED: .ASCIZ  /EXCEEDED MAXIMUM ERROR LIMIT/<CRLF>
037351  200     116     117     NODRVS: .ASCIZ  <CRLF>/NO DRIVES TO TEST/<CRLF>
037375  200     116     117     NOTEST: .ASCIZ  <CRLF>/NO TESTS SPECIFIED/<CRLF>

037422  200     012     122     ROTATE: .ASCIZ  <CRLF><LF>/ROTATIONAL SPEED TIMES/
037453  200     012     117     ONECYL: .ASCIZ  <CRLF><LF>/ONE CYLINDER SEEK TIMES/<CRLF>/ * FORWARD/
037520  200     012     101     AVERGE: .ASCIZ  <CRLF><LF>/AVERAGE SEEK TIMES/<CRLF>/ * FORWARD/
037560  200     012     115     MXSEEK: .ASCII  <CRLF><LF>/MAXIMUM SEEK TIMES/
037604  200     040     052     FWD:    .ASCIZ  <CRLF>/ * FORWARD/
037620  200     040     052     REV:    .ASCIZ  <CRLF>/ * REVERSE/
037634  200     012     101     MEASUR: .ASCIZ  <CRLF><LF>/AVERAGE SEEK TIME MEASUREMENT/

037674  200     115     111     MSGMIN: .ASCIZ  <CRLF>/MIN=/
037702  200     115     101     MSGMAX: .ASCIZ  <CRLF>/MAX=/
037710  200     101     126     MSGAVG: .ASCIZ  <CRLF>/AVG=/
037716  060     040     125     MSGOUS: .ASCIZ  /0 US/
037723  040     102     105     MBELOW: .ASCIZ  / BELOW THE MINIMUM OF /
037752  040     101     102     MABOVE: .ASCIZ  / ABOVE THE MAXIMUM OF /
040001  040     123     105     MSGSCH: .ASCIZ  / SEARCHES TIMED/
040021  040     123     105     MSGSEK: .ASCIZ  / SEEKS TIMED/
040036  040     116     117     MSGNOT: .ASCIZ  / NOT TIMED/
040051  040                     BLNKS4: .ASCII  / /
040052  040                     BLNKS3: .ASCII  / /
040053  040                     BLNKS2: .ASCII  / /
```

ASCIZ MESSAGES

```
     58 040054      040     000          BLNKS1: .ASCIZ  / /
     59 040056      101     114     114  MSG11X: .ASCIZ  /ALLOWABLE ROTATIONAL SPEED LIMITS FOR RM80/
     60 040131      101     114     114  MSG12X: .ASCIZ  /ALLOWABLE ONE CYLINDER SEEK LIMIT/
     61 040173      101     114     114  MSG13X: .ASCIZ  /ALLOWABLE AVERAGE SEEK TIME LIMIT/
     62 040235      101     114     114  MSG14X: .ASCIZ  /ALLOWABLE MAXIMUM (FORWARD) SEEK TIME LIMIT/
```

```
                                        .SBTTL  ERROR HEADER (EM) MESSAGES

                                ;
 040311     122     110     040 EM1:    .ASCIZ  /RH CONTROLLER INTERRUPT OCCURRED (RMAS=0)/
 040363     125     116     105 EM2:    .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
 040421     101     104     104 EM5:    .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
 040455     122     110     040 EM10:   .ASCIZ  /RH CONTROLLER FAILED TO RESPOND TO ADDRESSING/
 040533     104     122     111 EM11:   .ASCIZ  /DRIVE SELECTED IS NOT ONLINE/
 040570     111     115     120 EM12:   .ASCIZ  /IMPROPER HEADER DATA/
 040615     104     101     124 EM13:   .ASCIZ  /DATA COMPARE FAILURE/
 040642     104     111     123 EM17:   .ASCIZ  /DISK ERROR IN TIMING TEST/
 040674     103     114     117 EM20:   .ASCIZ  /CLOCK (KW11-P) OVERFLOW IN TIMING TEST/
 040743     104     111     123 EM23:   .ASCIZ  /DISK ERROR DURING SEEK/
 040772     123     105     105 EM24:   .ASCIZ  /SEEK NOT COMPLETE WITHIN 120 MS/
 041032     122     110     057 EM41:   .ASCIZ  @RH/RM ERROR@
 041046     106     101     124 EM46:   .ASCIZ  /FATAL WRITE CHECK ERROR/
```

```
                                .SBTTL  STATUS/ERROR INDICATOR MESSAGES

041076      117     106     106 MSGB14: .ASCIZ  /OFFLINE OR UNSAFE DRIVE REQUESTED/
041140      123     117     106 MSGB09: .ASCIZ  /SOFTWARE TIMEOUT ON THIS DRIVE/
041177      123     117     106 MSGB08: .ASCIZ  /SOFTWARE TIMEOUT ON ANOTHER DRIVE/
041241      105     122     122 MSGB06: .ASCIZ  "ERROR OCCURRED DURING I/O OPERATION"
041305      105     122     122 MSGB05: .ASCIZ  "ERROR OCCURRED DURING NON-I/O OPERATION"
041355      125     116     123 MSGB04: .ASCIZ  /UNSAFE OCCURRED/
041375      104     122     111 MSGB02: .ASCIZ  /DRIVE HAS NOT RESPONDED TO PORT REQUEST/
041445      104     122     111 MSGB01: .ASCIZ  /DRIVE HAS BECOME NON-EXISTENT/
```

DATA HEADER (DT) MESSAGES

```
                                        .SBTTL  DATA HEADER (DT) MESSAGES

                                        .ASCIZ  /ERR PC  RMAS/
                                        .ASCIZ  /ERR PC  DRIVE   RMAS    RMDS    RMER1   RMMR2   RMER2/
                                        .ASCIZ  /RMCS1   ERR PC/
                                        .ASCIZ  /DRIVE   ERR PC/
                                        .ASCIZ  /TEST    ERR PC  TST PC  DRIVE   CYLNDR  TRACK   SECTOR/
                                        .ASCIZ  /GDCYL   GDTRK   GDSCTR  BDCYL   BDTRK   BDSCTR/
                                        .ASCIZ  /GDDAT   BDDAT   WRDCNT  GDADR   BDADR/
                                        .ASCIZ  /TEST    ERR PC  DRIVE   RMCS1   RMDS    RMER1   RMMR2   RMER2/
                                        .ASCIZ  /TEST    ERR PC  TST PC  DRIVE   CYLNDR  TRACK/
                                        .ASCIZ  /GDDAT   BDDAT   WRDCNT  SECTOR/
                                        .ASCIZ  /TEST    ERR PC  DRIVE   CYLNDR  RMCS1   RMCS2   RMDS/
                                        .ASCIZ  /RMER1   RMMR2   RMER2   RMDC    RMHR/
                                        .ASCIZ  /TEST    ERR PC  TST PC  DRIVE/
                                        .ASCIZ  /TEST    ERR PC  TST PC  DRIVE   RMCS1   RMCS2   RMDS/
                                        .ASCIZ  /RMER1   RMMR2   RMER2/
                                        .ASCIZ  /RMCS1   RMCS2   RMDS    RMHR    RMDC    RMDA/
                                        .ASCIZ  /RMER1   RMMR2   RMER2/
                                        .ASCIZ  /RMER1   RMMR2   RMER2   RMWC    RMBA    RMDB/
                                        .EVEN
```

DATA TABLE (DT)

```
                                        .SBTTL  DATA TABLE (DT)

042754  001132  001204          DT1:    .WORD   $ERRPC,$REG3
042760  001132  001200  001204  DT2:    .WORD   $ERRPC,$REG1,$REG3,RMERRS,RMERRS+2,RMERRS+6,RMERRS+4
042776  001212  001132  001200  DT5:    .WORD   $TMP0,$ERRPC,$REG1,$REG5,RMERRS,RMERRS+2,RMERRS+6,RMERRS+4
043016  001522  001132          DT10:   .WORD   RH.ADR,$ERRPC
043022  001202  001132          DT11:   .WORD   $REG2,$ERRPC
043026  001212  001132  001176  DT12:   .WORD   $TMP0,$ERRPC,$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS
043044  001366  001372  001370  DT12A:  .WORD   CYL.DS,TRK.DS,SEC.DS,CYL.RD,TRK.RD,SEC.RD
043060  001212  001132  001176  DT13:   .WORD   $TMP0,$ERRPC,$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS
043076  001140  001142  001206  DT13A:  .WORD   $GDDAT,$BDDAT,$REG4,$GDADR,$BDADR
043110  001212  001132  001352  DT17:   .WORD   $TMP0,$ERRPC,CHKDRV,RM.REG,RM.REG+12,RM.REG+14,RM.REG+40,RM.REG+42
043130  001212  001132  001176  DT21:   .WORD   $TMP0,$ERRPC,$REG0,CHKDRV,CYL.DS,TRK.DS
043144  001200  001142  001206  DT21A:  .WORD   $REG1,$BDDAT,$REG4,$REG1
043154  001212  001132  001352  DT23:   .WORD   $TMP0,$ERRPC,CHKDRV,CYL.DS,RM.REG,RM.REG+10,RM.REG+12
043172  036654  036700  036702  DT23A:  .WORD   RM.REG+14,RM.REG+40,RM.REG+42,RM.REG+34,RM.REG+36
043204  001212  001132  001176  DT41:   .WORD   $TMP0,$ERRPC,$REG0,CHKDRV
043214  001212  001132  001176  DT42:   .WORD   $TMP0,$ERRPC,$REG0,CHKDRV,RM.REG,RM.REG+10,RM.REG+12
043232  001212  001132  001176  DT43:   .WORD   $TMP0,$ERRPC,$REG0,CHKDRV,RM.REG,RM.REG+10,RM.REG+12
043250  036654  036700  036702  DT43A:  .WORD   RM.REG+14,RM.REG+40,RM.REG+42
043256  001212  001132  001176  DT44:   .WORD   $TMP0,$ERRPC,$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS
043274  036640  036650  036652  DT44A:  .WORD   RM.REG,RM.REG+10,RM.REG+12,RM.REG+36,RM.REG+34,RM.REG+06
043310  036654  036700  036702  DT44B:  .WORD   RM.REG+14,RM.REG+40,RM.REG+42
043316  001212  001132  001176  DT45:   .WORD   $TMP0,$ERRPC,$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS
043334  036640  036650  036652  DT45A:  .WORD   RM.REG,RM.REG+10,RM.REG+12,RM.REG+36,RM.REG+34,RM.REG+06
043350  036654  036700  036702  DT45B:  .WORD   RM.REG+14,RM.REG+40,RM.REG+42,RM.REG+2,RM.REG+4,RM.REG+22
```

DATA FORMAT (DF) TABLE

```
                                .SBTTL  DATA FORMAT (DF) TABLE

043364  000001          DF1:    .WORD   1               ;NUMBER OF DATA HEADERS
043366     002                  .BYTE   2               ;NUMBER OF WORDS IN DATA TABLE
043367     000                  .BYTE   0               ;ALL 3 NUMBERS ARE OCTAL

043370  000001          DF2:    .WORD   1
043372     007                  .BYTE   7
043373     000                  .BYTE   0

043374  000001          DF3:    .WORD   1
043376     004                  .BYTE   4
043377     000                  .BYTE   0

043400  000001          DF4:    .WORD   1
043402     005                  .BYTE   5
043403     000                  .BYTE   0

043404  000001          DF10:   .WORD   1
043406     002                  .BYTE   2
043407     000                  .BYTE   0

043410  000001          DF11:   .WORD   1
043412     002                  .BYTE   2
043413     000                  .BYTE   0

043414  000002          DF12:   .WORD   2               ;2 DH'S TO BE TYPED
043416     007                  .BYTE   7               ;7 DATA WORDS FOLLOW THE 1ST DH
043417     160                  .BYTE   160             ;WORDS 1-4 ARE OCTAL 5-7 ARE DECIMAL
043420  041732                  .WORD   DH12A           ;ADDRESS OF 2ND DH
043422     006                  .BYTE   6               ;6 DATA WORDS FOLLOW THE 2ND DH
043423     000                  .BYTE   0               ;ALL WORDS ARE OCTAL

043424  000002          DF13:   .WORD   2
043426     007                  .BYTE   7
043427     160                  .BYTE   160
043430  042011                  .WORD   DH13A
043432     005                  .BYTE   5
043433     004                  .BYTE   4               ;WORD 3 IS DECIMAL

043434  000000          DF14:   .WORD   0
043436     005                  .BYTE   5
043437     004                  .BYTE   4               ;WORD 3 IS DECIMAL

043440  000001          DF17:   .WORD   1
043442     010                  .BYTE   ^D8
043443     000                  .BYTE   0

043444  000002          DF21:   .WORD   2
043446     006                  .BYTE   6
043447     060                  .BYTE   60
043450  042232                  .WORD   DH21A
043452     004                  .BYTE   4
043453     014                  .BYTE   14

043454  000000          DF22:   .WORD   0
043456     004                  .BYTE   4
```

DATA FORMAT (DF) TABLE

```
     58 043457     014                          .BYTE   14
     59
     60 043460  000002                DF23:     .WORD   2
     61 043462     007                          .BYTE   7
     62 043463     010                          .BYTE   10              ;WORD 4 IS DECIMAL
     63 043464  042356                          .WORD   DH23A
     64 043466     005                          .BYTE   5
     65 043467     000                          .BYTE   0
     66
     67
     68 043470  000001                DF41:     .WORD   1
     69 043472     004                          .BYTE   4
     70 043473     000                          .BYTE   0
     71
     72 043474  000001                DF42:     .WORD   1
     73 043476     007                          .BYTE   7
     74 043477     000                          .BYTE   0
     75
     76 043500  000002                DF43:     .WORD   2
     77 043502     007                          .BYTE   7
     78 043503     000                          .BYTE   0
     79 043504  042546                          .WORD   DH43A
     80 043506     003                          .BYTE   3
     81 043507     000                          .BYTE   0
     82
     83 043510  000003                DF44:     .WORD   3
     84 043512     007                          .BYTE   7
     85 043513     160                          .BYTE   160
     86 043514  042574                          .WORD   DH44A
     87 043516     006                          .BYTE   6
     88 043517     000                          .BYTE   0
     89 043520  042650                          .WORD   DH44B
     90 043522     003                          .BYTE   3
     91 043523     000                          .BYTE   0
     92
     93 043524  000003                DF45:     .WORD   3
     94 043526     007                          .BYTE   7
     95 043527     160                          .BYTE   160
     96 043530  042574                          .WORD   DH44A
     97 043532     006                          .BYTE   6
     98 043533     000                          .BYTE   0
     99 043534  042676                          .WORD   DH45A
    100 043536     006                          .BYTE   6
    101 043537     000                          .BYTE   0
    102
    103                              .SBTTL  START OF READ/WRITE BUFFER
    104 043540                       BUFFER:
    105
    106          000200                .END    200
```

SYMBOL TABLE

```
ABASE = 000000        AT4    = 000020        CI3     032724        DH17    042057        DT13    043060
ACDW1 = 000000        AT5    = 000040        CI4     033032        DH2     041520        DT13A   043076
ACDW2 = 000000        AT6    = 000100        CI5     033366        DH21    042154        DT17    043110
ACPUOP= 000000        AT7    = 000200        CI8     033410        DH21A   042232        DT2     042760
ACTDRV  031326        AUNIT  = 000000        CKSWR = 104407        DH23    042271        DT21    043130
ACTSTR  031327        AUSWR  = 000000        CK.CHR  030644        DH23A   042356        DT21A   043144
ADDW0 = 000000        AVECT1= 000000        CK.DEC  030616        DH41    042423        DT23    043154
ADDW1 = 000000        AVECT2= 000000        CK.DIG  030720        DH42    042461        DT23A   043172
ADDW10= 000000        AVERGE  037520        CK.NUM  031104        DH43A   042546        DT41    043204
ADDW11= 000000        A16    = 000400        CK.OCT  030570        DH44A   042574        DT42    043214
ADDW12= 000000        A17    = 001000        CLKSTA  001342        DH44B   042650        DT43    043232
ADDW13= 000000        BADTMO  003176        CLOSE   030510        DH45A   042676        DT43A   043250
ADDW14= 000000        BAI    = 000010        CLR    = 000040        DISPLA  001156        DT44    043256
ADDW15= 000000        BASFLG  001466        CLRQUE  036134        DISPRE  000174        DT44A   043274
ADDW2 = 000000        BITS    001560        CNTCLR  023146        DLT    = 100000        DT44B   043310
ADDW3 = 000000        BIT0   = 000001        CNTRLC  001322        DMD    = 000001        DT45    043316
ADDW4 = 000000        BIT00  = 000001        COLON   037163        DORTI   025662        DT45A   043334
ADDW5 = 000000        BIT01  = 000002        COMMA   037160        DPB.A   036506        DT45B   043350
ADDW6 = 000000        BIT02  = 000004        CONT   = 000100        DPB.B   036530        DT5     042776
ADDW7 = 000000        BIT03  = 000010        COUNT   025726        DPB.C   036552        DVA    = 004000
ADDW8 = 000000        BIT04  = 000020        CPSAVE  016332        DPB.R   036616        DVC    = 000200
ADDW9 = 000000        BIT05  = 000040        CR     = 000015        DPE    = 000010        EBL    = 020000
ADEVCT= 000000        BIT06  = 000100        CRLF   = 000200        DPINT   031262        ECH    = 000100
ADEVM = 000000        BIT07  = 000200        CYINC   001510        DPR    = 000400        ECI    = 004000
AENV  = 000000        BIT08  = 000400        CYL.DS  001366        DPRQS   031272        ECRC   = 001000
AENVM = 000000        BIT09  = 001000        CYL.RD  001360        DRIVES  037122        EECC   = 000020
AFATAL= 000000        BIT1   = 000002        C.SWR   001314        DROP    037302        EMPTYQ  036166
AMADR1= 000000        BIT10  = 002000        DCK    = 100000        DRQ    = 004000        EMTVEC= 000030
AMADR2= 000000        BIT11  = 004000        DDISP = 177570        DRVACT  031232        EM1     040311
AMADR3= 000000        BIT12  = 010000        DECSEC  026632        DRVCLR= 000111        EM10    040455
AMADR4= 000000        BIT13  = 020000        DECSK   006330        DRVINT  031610        EM11    040533
AMAMS1= 000000        BIT14  = 040000        DELTA   001450        DRVMSK  001354        EM12    040570
AMAMS2= 000000        BIT15  = 100000        DFLT    002456        DRVQUE  036200        EM13    040615
AMAMS3= 000000        BIT2   = 000004        DF1     043364        DRVSEL  001330        EM17    040642
AMAMS4= 000000        BIT3   = 000010        DF10    043404        DRVSTA  031242        EM2     040363
AMSGAD= 000000        BIT4   = 000020        DF11    043410        DRVTYP  031252        EM20    040674
AMSGLG= 000000        BIT5   = 000040        DF12    043414        DRY    = 000200        EM23    040743
AMSGTY= 000000        BIT6   = 000100        DF13    043424        DSWR   = 177570        EM24    040772
AMTYP1= 000000        BIT7   = 000200        DF14    043434        DTADPB  036574        EM41    041032
AMTYP2= 000000        BIT8   = 000400        DF17    043440        DTE    = 010000        EM46    041046
AMTYP3= 000000        BIT9   = 001000        DF2     043370        DTG    = 000020        EM5     040421
AMTYP4= 000000        BLNKS1  040054        DF21    043444        DTO    = 020000        ERINDX  025062
AOE    = 001000       BLNKS2  040053        DF22    043454        DTUW    031354        ERMAX   001316
APASS = 000000        BLNKS3  040052        DF23    043460        DT00   = 000001        ERR    = 040000
APRIOR= 000000        BLNKS4  040051        DF3     043374        DT01   = 000002        ERRCN   001512
APTCSU= 000040        BPTVEC= 000014        DF4     043400        DT02   = 000004        ERROR  = 104000
APTENV= 000001        BSE    = 100000        DF41    043470        DT03   = 000010        ERRVEC= 000004
APTSIZ= 000200        BUFFER  043540        DF42    043474        DT04   = 000020        ERR.CT  001464
APTSPO= 000100        BUSADR  001324        DF43    043500        DT05   = 000040        ESRC   = 004000
ASWREG= 000000        BYPASS  001350        DF44    043510        DT06   = 000100        ES.SAV  036252
ATA    = 100000       CALL.A  024076        DF45    043524        DT07   = 000200        EXCEED  037313
ATABIT  031356        CALL.B  024244        DH1     041503        DT08   = 000400        EXIT.A  011410
ATESTN= 000000        CALL.C  024466        DH10    041605        DT1     042754        EXIT0   005702
AT0    = 000001       CALL.R  024710        DH11    041624        DT10    043016        EXIT1   006110
AT1    = 000002       CHANGE= 000001        DH12    041643        DT11    043022        EXIT10  010562
AT2    = 000004       CHKDRV  001352        DH12A   041732        DT12    043026        EXIT11  011424
AT3    = 000010       CI1     032572        DH13A   042011        DT12A   043044        EXIT12  012126
```

SYMBOL TABLE

```
EXIT13  012666          LC      002326          MSG.FT  036723          OPT     032312          RDY   = 000200
EXIT14  013436          LDCMD   024032          MSG.IC  036720          OR    = 000200          RD.ADR  035570
EXIT15  015140          LF    = 000012          MSG.IT  036731          PACK  = 000123          RD.RM   035552
EXIT2   006352          LKS     001546          MSG.LC  036715          PAR   = 000010          RD.RM1  035566
EXIT3   006572          LKV     001542          MSG.LS  036737          PAT     002344          RD.RM3  035624
EXIT4   007204          LODEV   037100          MSG.LT  036726          PDA   = 000400          RD.RM4  035630
EXIT5   007430          LODFLT  023516          MSG.PA  036742          PFECH   016754          RD.WRD  035572
EXIT6   007720          LODPRM  023734          MSG.R   036710          PFECH1  016764          READ  = 000171
EXIT7   010316          LOP.CK  025022          MSGOUS  037716          PFECH2  017046          READHD= 000173
FC      002324          LPB     001556          MSG11X  040056          PFECH3  017100          READIN= 000121
FER   = 000020          LPS     001554          MSG12X  040131          PFECH4  017110          RECAL = 000107
FMT16 = 010000          LPTAVL  001326          MSG13X  040173          PFTSTN  017114          RELEAS= 000113
FS      002340          LP.AVL  023176          MSG14X  040235          PGE   = 002000          RESREG= 104413
FT      002332          LS      002342          MUR   = 001000          PGM   = 001000          RESVEC= 000010
FWD     037604          LSC   = 004000          MWP   = 000010          PHA   = 000200          REV     037620
F1    = 000002          LSIT  = 00000           MXF   = 001000          PIP   = 020000          REX   = 010000
F2    = 000004          LST   = 00200           MXSEEK  037560          PIRQ  = 177772          RHVEC   001524
F3    = 000010          LSTPK   001374          MXSTAL  001462          PIRQVE= 000240          RH.ADR  001522
F4    = 000020          LT      002334          MXWNDW  031374          PKB     001536          RMADR   031366
F5    = 000040          MABOVE  037752          M.DPID  022652          PKC     001540          RMAS  = 000016
GETADR  036276          MBELOW  037723          M.DP40  022710          PKCS    001534          RMBA  = 000004
GETNUM  026750          MCLK  = 004000          M.DP41  022744          PKV     001530          RMCS1 = 000000
GETREG= 000141          MCPE  = 020000          M.DP42  022754          PLFS  = 002000          RMCS2 = 000010
GETREQ  036222          MDF   = 000100          M.DP44  023006          POPQUE  036234          RMDA  = 000006
GETSWR  026656          MEASUR  037634          M.DP50  023020          PRM     002320          RMDB  = 000022
GO    = 000001          MI    = 000004          NBA   = 100000          PRMLMT  002410          RMDC  = 000034
GTSWR = 104406          MOC   = 000400          NC1     002346          PRMMSG  002434          RMDS  = 000012
GTTST1  027244          MOH   = 020000          NC2     002350          PRMPT   002352          RMDT  = 000026
GTTST2  027330          MOL   = 010000          NED   = 010000          PRM0    002726          RMEC1 = 000044
GTTST3  027620          MPE   = 000400          NEM   = 004000          PRM1    002742          RMEC2 = 000046
GTTST4  027622          MRD   = 002000          NOASSY= 000001          PRM10   003106          RMERRS  031224
GTTST5  027772          MRMCS1  036466          NOCLOK  037166          PRM11   003124          RMER1 = 000014
GTTST6  030032          MRMVEC  036476          NODRVS  037351          PRM12   003136          RMER2 = 000042
GT.PRM  027106          MS    = 000040          NONE    037153          PRM13   003146          RMHR  = 000036
GT.PR1  027110          MSC   = 000002          NOTEST  037375          PRM14   003156          RMINIT  031376
GT.PR2  027240          MSDRIV  037273          NOTPRS  037036          PRM15   003166          RMLA  = 000020
HCE   = 000200          MSEN  = 010000          NOTRM   037063          PRM2    002764          RMMR1 = 000024
HCI   = 002000          MSER  = 000200          NOTSAF  037053          PRM3    003002          RMMR2 = 000040
HCRC  = 000400          MSGAVG  037710          OBCK  = 100000          PRM4    003020          RMOF  = 000032
HT    = 000011          MSGB01  041445          OBEN  = 040000          PRM5    003036          RMR   = 000004
IAE   = 002000          MSGB02  041375          OCC   = 100000          PRM6    003054          RMSN  = 000030
IBSAVE  016334          MSGB04  041355          OFD   = 000200          PRM7    003072          RMTMR   035034
IC      002330          MSGB05  041305          OFFSET= 000115          PR0   = 000000          RMVEC   031370
IE    = 000100          MSGB06  041241          OM    = 000001          PR1   = 000040          RMWC  = 000002
ILF   = 000001          MSGB08  041177          ONECYL  037453          PR2   = 000100          RM.REG  036640
ILR   = 000002          MSGB09  041140          OPE   = 020000          PR3   = 000140          RM80    032054
INCCYL  026602          MSGB14  041076          OPI   = 020000          PR4   = 000200          ROTATE  037422
INCEC   023030          MSGMAX  037702          OPNFLG  001336          PR5   = 000240          RPT     002322
INCSK   006266          MSGMIN  037674          OPNPCA  030236          PR6   = 000300          RSTART  005102
INCTRK  026552          MSGNOT  040036          OPNTST  030064          PR7   = 000340          RSTRT1  005132
IOTVEC= 000020          MSGSCH  040001          OPN.CT  030070          PS    = 177776          RTC   = 000117
IR    = 000100          MSGSEK  040021          OPN.N1  030432          PSEL  = 002000          RTURN   015406
ISR     033526          MSGTST  037265          OPN.N2  030436          PSW   = 177776          R6    =%000006
IT      002336          MSG.CS  036750          OPN.X1  030450          PWRVEC= 000024          R7    =%000007
ITEM41  002240          MSG.EQ  036746          OPN.X2  030454          QDRV    031302          SAVCSW  001320
IVC   = 010000          MSG.FC  036712          OPN.1   030074          RDCHR = 104410          SAVEFG  031330
LBC   = 002000          MSG.FS  036734          OPN.2   030116          RDLIN = 104411          SAVREG= 104412
```

SYMBOL TABLE

```
SC       034202
SCOPE  = 000004
SCTRWC= 177400
SC0    = 000100
SC1    = 000200
SC11     034554
SC12     034632
SC13     034674
SC2    = 000400
SC3      034252
SC4      034256
SC5      034270
SC6      034366
SC8      034540
SEARCH= 000131
SEC.DS   001370
SEC.RD   001364
SEEK   = 000105
SEEKFG   031332
SEKCNT   001446
SEKTMR   001444
SELDRV= 000145
SERIAL   037253
SETFOR= 000143
SETVEC   004356
SET.IE   036062
SHUT     021410
SKCNT    001502
SKI    = 040000
SKIP     034006
SKSIZ    001506
SKTIM    001472
SLASH    036773
SPTYP    026060
SP11A    001710
SP11B    001716
SP12     001724
SP13     001732
SP14     001740
SRCHWT   031324
SRCH00   025500
SRTDRV   004570
SRTINT   004300
SRVCLK   023504
SSE    = 000040
SSEI   = 001000
STACK  = 001100
STALL    025220
STALL0   001436
STALL1   001454
STALL2   001456
STALL3   001460
START    003324
START1   003270
START2   003312
START3   003260
START4   003302

STATBL   001746
STKLMT= 177774
ST0      035126
ST01     035156
ST02     035344
ST03     035414
ST05     035440
ST06     035446
ST07     035504
ST08     035534
ST09     035544
STRTMR   025664
STRT1A   003274
STRT2A   003316
ST.CLK   023240
ST.LCL   023450
ST.PCL   023406
SVADR    001440
SVRH70   035744
SVSTAT   001356
SWR      001154
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
TAB.XY= 001114
TAP    = 040000
TBITVE= 000014
TD       033570
TEST0    005612
TEST1    006074
TEST10   010504
TEST11   011102
TEST12   011636

TEST13   012340
TEST14   013100
TEST15   013644
TEST2    006246
TEST3    006510
TEST4    006744
TEST5    007342
TEST6    007566
TEST7    010056
TICKMS   001344
TICKUS   001346
TIMER    031334
TIM.DN   001414
TIM.PT   001432
TIM.UP   001376
TKVEC  = 000060
TPB      001552
TPS      001550
TPS50    011422
TPS60    011404
TPVEC  = 000064
TP50     011414
TP60     011376
TRAPVE= 000034
TRCKWC   001452
TRE    = 040000
TRK.DS   001372
TRK.RD   001362
TRNSWT   031322
TRTVEC= 000014
TSTNMS   001332
TST0     005442
TST1     005704
TST10    010320
TST11    010564
TST12    011426
TST13    012130
TST14    012670
TST15    013440
TST2     006112
TST3     006354
TST4     006574
TST5     007206
TST6     007432
TST7     007722
TYPDS  = 104405
TYPE   = 104401
TYPERR   016336
TYPOC  = 104402
TYPON  = 104404
TYPOS  = 104403
TYPTIM   026166
T11A     001640
T11B     001650
T12      001660
T13      001670
T14      001700

UNS    = 040000
UNSTAT   036777
UNTOFF   037015
UNTON    037026
UPE    = 020000
US1    = 000001
US2    = 000002
US4    = 000004
VERIFY   025302
VV     = 000100
WC     = 000040
WCE    = 040000
WCEFLG   001434
WCF    = 000040
WD     = 000010
WLE    = 004000
WRCKD  = 000151
WRCKHD= 000153
WRITE  = 000161
WRL    = 004000
WRTHD  = 000163
WRT.AD   035714
WRT.RM   035632
WRT.R1   035710
WRT.R3   035734
WRT.R4   035740
WRT.R5   035742
WRT.WD   035712
XXDP     001470
$APTHD   001100
$ATYC    015452
$ATY1    015426
$ATY3    015434
$ATY4    015444
$AUTOB   001150
$BASE    001310
$BDADR   001136
$BDDAT   001142
$BELL    001224
$BUF   = 000006
$CHARC   017446
$CKSWR   020434
$CMTAG   001114
$CM1   = 000006
$CM2   = 000014
$CM3   = 000006
$CM4   = 000003
$CNTLC   021346
$CNTLG   021360
$CNTLU   021353
$COMND= 000002
$CPUOP   001262
$CRLF    001231
$CYL   = 000012
$DBLK    020114
$DB2D    022246
$DECVL   022426

$DEVCT   001244
$DEVM    001312
$DIV     022604
$DOAGN   015376
$DTBL    020104
$ENDAD   015366
$ENDCT   015322
$ENULL   015402
$ENV     001254
$ENVM    001255
$EOP     015142
$EOPCT   015314
$ERFLG   001117
$ERMAX   001131
$ERROR   015674
$ERRPC   001132
$ERRTB   002000
$ERTTL   001126
$ESCAP   001222
$ETABL   001254
$ETEND   001314
$FATAL   001236
$FFLG    015672
$FILLC   001172
$FILLS   001171
$FMT   = 000001
$GDADR   001134
$GDDAT   001140
$GET42   015356
$GTSWR   020524
$HD    = 000000
$HIBTS   001100
$HINUM   022600
$ICNT    001120
$INTAG   001151
$ITEMB   001130
$LF      001232
$LFLG    015671
$LONUM   022602
$LPADR   001122
$LPERR   001124
$MADR1   001266
$MADR2   001272
$MADR3   001276
$MADR4   001302
$MAIL    001234
$MAMS1   001264
$MAMS2   001270
$MAMS3   001274
$MAMS4   001300
$MBADR   001102
$MFLG    015670
$MNEW    021376
$MSGAD   001250
$MSGLG   001252
$MSGTY   001234
$MSWR    021365
```

SYMBOL TABLE

```
$MTYP1  001265          $REG   = 000014        $SUPRS  022442          $TMP0   001212          $TYPEC  017330
$MTYP2  001271          $REGAD  001174          $SVLAD  021760          $TMP1   001214          $TYPEX  017450
$MTYP3  001275          $REG0   001176          $SVPC  = 000220          $TMP2   001216          $TYPOC  017476
$MTYP4  001301          $REG1   001200          $SWR   = 167000          $TN    = 000016          $TYPON  017512
$MXCNT  022030          $REG2   001202          $SWREG  001256          $TNPWR  022356          $TYPOS  017452
$NULL   001170          $REG3   001204          $SWRMK= 000000          $TPB    001166          $UNIT   001246
$NWTST= 000001          $REG4   001206          $TATUS= 000016          $TPFLG  001173          $UNITM  001110
$OCNT   017674          $REG5   001210          $TESTN  001240          $TPS    001164          $USWR   001260
$OMODE  017676          $RESRE  022070          $TIMES  001220          $TRAP   022126          $VECT1  001304
$OVER   022014          $RM80   037115          $TKB    001162          $TRAP2  022150          $VECT2  001306
$PASS   001242          $RTNAD  015400          $TKCNT  020124          $TRK   = 000011          $WCNT  = 000004
$PASTM  001106          $SAVRE  022032          $TKINT  020134          $TRP   = 000014          $XOFF  = 000023
$PSEL  = 000003         $SB2D   022212          $TKQEN= 020133          $TRPAD  022162          $XON   = 000021
$QUES   001230          $SCOPE  021432          $TKQIN  020126          $TSTM   001104          $XTSTR  021450
$RAND   022502          $SEC   = 000010         $TKQUU  020130          $TSTNM  001116          $$GET4= 000000
$RDCHR  020776          $SETUP= 000167          $TKQSR  020132          $TTYIN  021322          $OFILL  017675
$RDLIN  021066          $SSENB= 000020          $TKS    001160          $TYPDS  017700          .$X    = 001100
$RDSZ  = 000024         $STUP  = 177777         $TKSRV  020204          $TYPE   017116

. ABS.  043540      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  57856 WORDS  ( 226 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  71 PAGES
,A:CZRNGA/C=A:CZRNGA.DOC,CZRNGA,SYSMAC/M
```

CROSS REFERENCE TABLE (CREF V04.00)

```
$$GET4  15-1     15-1#
$OFILL  20-1     20-1#    20-1*    20-1*
$40CAT  17-1     23-1
$APTHD   5-12     5-12#
$ASTAT  16-1     16-1
$ATY1   16-1#
$ATY3   16-1#    19-1
$ATY4   16-1#    17-1
$ATYC   16-1     16-1#
$AUTOB   6-0#    10-37*   22-1     22-1     22-1
$BASE    6-0#
$BDADR   6-0#    45-11
$BDDAT   6-0#    45-11    45-14
$BELL    6-0#    10-41    17-1     17-1     17-1     22-1     22-1     22-1
$BUF    33-206   36-84    36-111
$CHARC  19-1     19-1#    19-1*    19-1*    19-1*
$CKSWR  22-1#    25-1     25-1
$CM1     6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0#     6-0#
         6-0#     6-0#     6-0#     6-0#     6-0#
$CM2     6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0      6-0#     6-0#
         6-0#     6-0#     6-0#     6-0#     6-0#
$CM3     6-0      6-0      6-0#
$CM4     6-0      6-0      6-0      6-0      6-0      6-0      6-0#     6-0#     6-0#     6-0#
$CMTAG   6-0#    10-32    10-32    10-32    10-32    10-32    10-32    10-32
$CNTLC  22-1     22-1     22-1     22-1     22-1#
$CNTLG  22-1     22-1#
$CNTLU  22-1     22-1     22-1#
$COMND  33-203   36-68    36-125
$CPUOP   6-0#
$CRLF    6-0#    10-64    10-102   10-147   11-49    11-69    12-46    12-68    14-583   17-1     17-1     17-1     18-42    18-58
        18-69    18-79    18-93    18-100   19-1     19-1     19-1     22-1     22-1     22-1     22-1     32-15    32-687   32-688
        32-703   32->07   39-16    39-19
$CYL    33-209
$DB2D   14-432   14-583   14-583   26-1     27-1#
$DBLK   21-1     21-1     21-1#
$DEFVL  27-1     27-1#
$DEVCT   6-0#
$DEVM    6-0#
$DIV    13-70    13-335   30-24#   32-759
$DOAGN  15-1     15-1     15-1#
$DTBL   21-1     21-1#
$ENDAD   5-9     10-37    15-1#    17-1
$ENDCT  10-32    11-44*   11-61*   11-70    15-1#
$ENULL  15-1     15-1#
$ENV     6-0#    10-37    16-1     16-1     17-1     19-1
$ENVM    6-0#    10-32    16-1     19-1     19-1
$EOP    12-37    14-388   15-1#    22-7     32-23    32-460   39-51
$EOPCT  10-32*   11-70*   12-27*   15-1     15-1#    32-22*
$ERFLG   6-0#    17-1     17-1     17-1*    23-1     23-1     23-1     23-1     23-1     23-1*
$ERMAX   6-0#    10-32*   12-69*   13-44*   13-79*   13-109*  13-141*  13-170*  13-239*  13-274*  13-316*  13-380*  14-23*   14-140*
        14-221*  14-305*  14-388*  23-1     23-1     23-1     23-1*
$ERROR  10-32    17-1#
$ERRPC   6-0#    17-1     17-1     17-1     17-1*    17-1*    18-111   45-3     45-4     45-5     45-6     45-7     45-8     45-10
        45-12    45-13    45-15    45-17    45-18    45-19    45-21    45-24
$ERRTB   8-0#    18-37    18-38
$ERTTL   6-0#    15-1     15-1     15-1*    17-1     17-1     17-1*
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
$OMODE  20-1     20-1#    20-1*    20-1*    20-1*    20-1*
$OVER   23-1     23-1     23-1     23-1#
$PASS    6-0#    10-32*   15-1     15-1     15-1*    15-1*
$PASTM   5-12#
$PSEL   33-204
$QUES    6-0#    17-1     17-1     19-1     19-1     22-1     22-1     22-1     22-1
$R2A    25-1
$RAND   13-70    13-329   29-1#    32-531
$RDCHR  22-1#    25-1     25-1
$RDDEC  25-1
$RDLIN  22-1#    25-1     25-1
$RDOCT  25-1
$RDSZ   22-1     22-1#
$REG    33-210
$REG0    6-0#    18-14*   45-8     45-10    45-13    45-17    45-18    45-19    45-21    45-24
$REG1    6-0#    18-15*   45-4     45-5     45-14    45-14
$REG2    6-0#    18-16*   45-7
$REG3    6-0#    18-17*   45-3     45-4
$REG4    6-0#    18-18*   45-11    45-14
$REG5    6-0#    18-19*   45-5
$REGAD   6-0#    10-39
$RESRE  24-1#    25-1
$RHEXT  33-167   33-197   38-90
$RM80   10-135   41-24#
$RTNAD  15-1#
$SAVRE  24-1#    25-1     25-1
$SB2D   14-428   26-1#    32-695   32-700   32-736   32-741   32-743   32-746   32-751   32-753   32-763   32-766   32->37
$SCOPE  10-32    23-1#
$SEC    33-207
$SETUP   4-315    4-315    4-315    4-315    4-315    4-315    4-315#   4-315#   4-315#   4-315#   4-315#   4-315#   4-315#   10-32
        10-32    10-32    10-32    10-32    10-32    10-32    10-32    10-32    10-32    10-32    10-32    10-37    10-37    10-37
        15-1     15-1     17-1     17-1     17-1     17-1     22-1     22-1     22-1     22-1     22-1     23-1
$SSENB  32-256*  32-297*  32-560   33-212   36-124
$STUP    4-315    4-315    4-315    4-315    4-315    4-315    4-315#   4-315#   4-315#   4-315#   4-315#   4-315#   4-315#   4-315#
         4-315#   4-315#   4-315#   4-315#
$SUPRS  14-429   14-433   14-583   14-583   28-1#    32-696   32-701   32-736   32-741   32-743   32-746   32-751   32-753   32-763
        32-766   32->37
$SVLAD  23-1     23-1#
$SVPC    5-9      5-9#
$SWR     4-30#    4-40     4-41     4-41     4-41     4-41     4-41     4-41     4-41     4-41     5-196#   6-0      6-0      6-0
        10-32    10-32    10-32    10-32    10-32    13-44    13-79    13-109   13-141   13-170   13-239   13-274   13-316   13-380
        14-23    14-140   14-221   14-305   14-388   15-1     15-1     15-1     15-1     15-1     17-1     17-1     17-1     17-1
        17-1     17-1     17-1     17-1     17-1     17-1     17-1     23-1     23-1     23-1     23-1     23-1     23-1     23-1
        23-1     23-1     23-1     23-1     23-1     23-1     23-1     23-1     23-1     23-1     23-1     23-1
$SWREG   6-0#    10-32
$SWRMK  23-1
$TATUS  33-211   36-67*
$TESTN   6-0#    13-44*   13-45    13-79*   13-80    13-109*  13-110   13-141*  13-142   13-170*  13-171   13-239*  13-240   13-274*
        13-275   13-316*  13-317   13-380*  13-381   14-23*   14-24    14-140*  14-141   14-221*  14-222   14-305*  14-306   14-388*
        14-389   18-24    18-27    23-1*
$TIMES   6-0#    10-32*   13-44*   13-79*   13-109*  13-141*  13-170*  13-239*  13-274*  13-316*  13-380*  14-23*   14-140*  14-221*
        14-305*  14-388*  15-1*    23-1     23-1     23-1     23-1*    23-1*
$TKB     6-0#    19-1     19-1     22-1     22-1     22-1     22-1     22-1     22-1     22-1
$TKCNT  22-1     22-1     22-1#    22-1*    22-1*    22-1*
$TKINT  10-79    15-4     22-1     22-1     22-1#
$TKQEN  22-1     22-1     22-1#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
$TKQIN  22-1    22-1    22-1#   22-1*   22-1*   22-1*   22-1*
$TKQOU  22-1    22-1    22-1#   22-1*   22-1*   22-1*
$TKQSR  22-1    22-1    22-1    22-1#
$TKS     6-0#   19-1    19-1    22-1    22-1    22-1    22-1    22-1    22-1*   22-1*   22-1*   22-1*   22-1*   22-1*
$TKSRV  22-1    22-1#
$TMP0    6-0#   18-11*  45-5    45-8    45-10   45-12   45-13   45-15   45-17   45-18   45-19   45-21   45-24
$TMP1    6-0#
$TMP2    6-0#
$TN      4-31#   4-40    13-44   13-44   13-44   13-44   13-44   13-44   13-44   13-44   13-44   13-44   13-44#  13-45
        13-50   13-70   13-79   13-79   13-79   13-79   13-79   13-79   13-79   13-79   13-79   13-79   13-79#  13-80
        13-88   13-96   13-109  13-109  13-109  13-109  13-109  13-109  13-109  13-109  13-109  13-109  13-109# 13-110
        13-114  13-132  13-141  13-141  13-141  13-141  13-141  13-141  13-141  13-141  13-141  13-141  13-141# 13-142
        13-146  13-160  13-170  13-170  13-170  13-170  13-170  13-170  13-170  13-170  13-170  13-170  13-170# 13-171
        13-175  13-226  13-239  13-239  13-239  13-239  13-239  13-239  13-239  13-239  13-239  13-239  13-239# 13-240
        13-244  13-258  13-263  13-274  13-274  13-274  13-274  13-274  13-274  13-274  13-274  13-274  13-274  13-274#
        13-275  13-279  13-288  13-305  13-316  13-316  13-316  13-316  13-316  13-316  13-316  13-316  13-316  13-316
        13-316# 13-317  13-320  13-363  13-367  13-380  13-380  13-380  13-380  13-380  13-380  13-380  13-380  13-380
        13-380  13-380# 13-381  13-389  13-403  14-23   14-23   14-23   14-23   14-23   14-23   14-23   14-23   14-23
        14-23   14-23#  14-24   14-27   14-33   14-60   14-111  14-129  14-140  14-140  14-140  14-140  14-140  14-140
        14-140  14-140  14-140  14-140  14-140# 14-141  14-144  14-150  14-200  14-205  14-221  14-221  14-221  14-221
        14-221  14-221  14-221  14-221  14-221  14-221  14-221# 14-222  14-225  14-282  14-287  14-305  14-305  14-305
        14-305  14-305  14-305  14-305  14-305  14-305  14-305  14-305# 14-306  14-309  14-315  14-370  14-375  14-388
        14-388  14-388  14-388  14-388  14-388  14-388  14-388  14-388  14-388# 14-389  14-392  14-398  14-450  14-498
        14-549  14-595  32-<69  32-<96  32-=89
$TNPWR  27-1    27-1    27-1#
$TPB     6-0#   10-44*  17-1*   17-1*   19-1    19-1    19-1*
$TPFLG   6-0#   19-1    19-1    19-1
$TPS     6-0#   10-43*  17-1*   17-1*   19-1    19-1    19-1
$TRAP   10-32   25-1#
$TRAP2  25-1    25-1#
$TRK    33-208
$TRP    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1
        25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1
        25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1
        25-1    25-1    25-1#   25-1#   25-1#   25-1#   25-1#   25-1#   25-1#   25-1#   25-1#   25-1#   25-1#   25-1#
$TRPAD  25-1    25-1#
$TSTM    5-12#
$TSTNM   6-0#   13-44*  13-79*  13-109* 13-141* 13-170* 13-239* 13-274* 13-316* 13-380* 14-23*  14-140* 14-221* 14-305*
        14-388* 15-1*   17-1    17-1    17-1    18-11   23-1    23-1    23-1    23-1    23-1    23-1*   32-184  32-767
$TTYIN  22-1    22-1    22-1    22-1    22-1    22-1#
$TYPBN  25-1
$TYPDS  21-1#   25-1    25-1
$TYPE   16-1    19-1#   25-1    25-1
$TYPEC  19-1    19-1    19-1    19-1#   22-1
$TYPEX  19-1    19-1    19-1#
$TYPOC  20-1#   25-1    25-1
$TYPON  20-1    20-1#   25-1
$TYPOS  20-1#   25-1
$UNIT    6-0#
$UNITM   5-12#
$USWR    6-0#
$VECT1   6-0#
$VECT2   6-0#
$WCNT   33-205  36-82   36-105  36-108
$XOFF   19-1    19-1
$XON    19-1    19-1    22-1
```

CROSS REFERENCE TABLE (CREF V04.00)

```
$XTSTR  23-1#
.$ASTA  16-1      16-1
.$X      5-12      5-12#
A16      4-86#
A17      4-87#
ABASE    6-0       6-0
ACDW1    6-0
ACDW2    6-0
ACPUOP   6-0       6-0
ACTDRV  33-118#  35-13*   35-57*   36-3*    36-11*   37-6
ACTSTR  33-124#  37-8*    37-23*
ADDW0    6-0
ADDW1    6-0
ADDW10   6-0
ADDW11   6-0
ADDW12   6-0
ADDW13   6-0
ADDW14   6-0
ADDW15   6-0
ADDW2    6-0
ADDW3    6-0
ADDW4    6-0
ADDW5    6-0
ADDW6    6-0
ADDW7    6-0
ADDW8    6-0
ADDW9    6-0
ADEVCT   6-0       6-0
ADEVM    6-0       6-0
AENV     6-0       6-0
AENVM    6-0       6-0
AFATAL   6-0       6-0
AMADR1   6-0       6-0
AMADR2   6-0       6-0
AMADR3   6-0       6-0
AMADR4   6-0       6-0
AMAMS1   6-0       6-0
AMAMS2   6-0       6-0
AMAMS3   6-0       6-0
AMAMS4   6-0       6-0
AMSGAD   6-0       6-0
AMSGLG   6-0       6-0
AMSGTY   6-0       6-0
AMTYP1   6-0       6-0
AMTYP2   6-0       6-0
AMTYP3   6-0       6-0
AMTYP4   6-0       6-0
AOE      4-160#
APASS    6-0       6-0
APRIOR   6-0
APTCSU  16-1#     19-1
APTENV  16-1      16-1#     17-1      19-1
APTSIZ  10-32     16-1#
APTSPO  16-1      16-1#     19-1
ASWREG   6-0       6-0
ATO      4-212#
```

CROSS REFERENCE TABLE (CREF V04.00)

AT1	4-213#													
AT2	4-214#													
AT3	4-215#													
AT4	4-216#													
AT5	4-217#													
AT6	4-218#													
AT7	4-219#													
ATA	4-147#													
ATABIT	11-31	32-10	32-21	32-<23	32-<29	33-160#	34-99	35-96	35-100	35-167	36-224	36-238	36-246	36-268
	37-78													
ATESTN	6-0	6-0												
AUNIT	6-0	6-0												
AUSWR	6-0	6-0												
AVECT1	6-0	6-0												
AVECT2	6-0	6-0												
AVERGE	7-0	41-40#												
BADTMO	10-3#	10-34												
BAI	4-104#													
BASFLG	7-0#	13-81*												
BIT0	4-80#													
BIT00	4-80	4-80#	7-0	7-0	11-8	12-39	17-1	17-1	23-1	23-1	32-146			
BIT01	4-80	4-80#	7-0	7-0	32-457	32-485	35-49	35-251						
BIT02	4-80	4-80#	7-0	7-0	32-457	32-488								
BIT03	4-80	4-80#	7-0	7-0										
BIT04	4-80	4-80#	7-0	7-0	32-491	36-221								
BIT05	4-80	4-80#	7-0	7-0	32-494	36-219	37-54							
BIT06	4-80	4-80#	7-0	7-0	32-497	34-106	35-40	35-111	36-53	38-116				
BIT07	4-80	4-80#	7-0	7-0	34-106	35-221	36-23	36-219	36-221	36-242	38-77			
BIT08	4-80	4-80#	7-0	32-158	32-485	34-106	37-66							
BIT09	4-80	4-80#	7-0	17-1	17-1	23-1	23-1	32-500	35-145	37-52				
BIT1	4-80#													
BIT10	4-80#	7-0	17-1	32-457	32-?65									
BIT11	4-80#	7-0	23-1	34-81	35-72	35-88	36-269							
BIT12	4-80#	7-0	32-457	34-76	34-94	34-106	35-51	35-84	38-19	38-54	38-117			
BIT13	4-80#	7-0	17-1	32-457	36-193									
BIT14	4-80#	7-0	14-465	14-513	23-1	32-457	32-491	35-46	35-81	36-285	37-103	38-114		
BIT15	4-80#	7-0	32-633	32-634	35-46	35-49	35-51	35-81	35-84	35-251	36-53	36-219	36-221	36-285
	37-52	37-66	37-103	37-110										
BIT2	4-80#	37-110												
BIT3	4-80#													
BIT4	4-80#													
BIT5	4-80#													
BIT6	4-80#	35-74												
BIT7	4-80#	36-45												
BIT8	4-80#	36-45												
BIT9	4-80#													
BITS	7-0#	13-44	13-79	13-109	13-141	13-170	13-239	13-274	13-316	13-380	14-23	14-140	14-221	14-305
	14-388	32-<80	32-=14	32-=43	32->03	32-?43								
BLNKS1	41-58#													
BLNKS2	10-130	18-63	18-89	18-96	18-103	32->32	41-57#							
BLNKS3	41-56#													
BLNKS4	10-104	41-55#												
BPTVEC	4-80#													
BSE	4-281#	32-237	32-270	32-311	32-506									
BUFFER	32-554*	32-556	32-558	32-563	32-565	32-566	32-567	32-632	32-669	40-8	40-33	40-58	40-83	40-108
	46-104#													

CROSS REFERENCE TABLE (CREF V04.00)

```
BUSADR   7-0#    10-15*   10-18*   10-22*   10-25*   39-13    39-15*
BYPASS   7-0#    12-37*   13-70*   13-92*   13-117*  13-149*  13-182*  13-247*  13-282*  13-324*  13-392*  14-54*   14-161*  14-240*
        14-326*  14-408*  14-452*  14-500*
C.SWR    7-0#    11-7     11-10    12-39    13-182   14-58    14-109   32-93    32-103   32-113   32-146   32-209   32-284   32-325
        32-529   32-:55   32-:59   32-:60*
CALL.A   12-43   12-45    13-70    13-340   14-54    14-161   14-240   14-326   14-408   14-453   14-501   32-225#  32-228   32-575
CALL.B   13-70   13-95    13-121   13-128   13-152   13-156   13-187   13-193   13-208   13-214   13-252   13-254   13-292   13-294
        13-296   13-298   13-300   13-302   13-360   13-395   32-255#  32-259
CALL.C   13-94   13-394   32-296#  32-300
CALL.R   14-473  14-521   32-242   32-275   32-316   32-432#  32-435
CHANGE   4-8#
CHKDRV   7-0#    12-13*   12-17*   12-22    15-1     32-35*   32-36    32-188   45-8     45-10    45-12    45-13    45-15    45-17
        45-18    45-19    45-21    45-24
CI1      35-101   35-131#
CI3      35-103   35-154#
CI4      35-92    35-172#
CI5      35-150   35-168   35-234#
CI8      35-36    35-110   35-141   35-143   35-144   35-146   35-148   35-157   35-164   35-166   35-178   35-183   35-187   35-189
        35-194   35-198   35-200   35-210   35-219   35-230   35-242#  36-25    36-44    36-47    36-50    36-103   36-117   36-119
        36-121   36-123   36-126   36-155   36-232   37-95    38-89
CK.CHR   32-;85   32-<20   32->41   32-@19#  32-@56   32-@63   32-@68   32-A13
CK.DEC   32-?97#  32-@27
CK.DIG   32-<22   32->55   32-@50#
CK.NUM   32-;87   32-@96#
CK.OCT   32-<57   32-<61   32-<84   32-<88   32-?79#  32-@98   32-A08
CKSWR    17-1     17-1     23-1     25-1#
CLKSTA    7-0#    11-71    14-25    14-142   14-223   14-307   14-390   32-78*   32-80*   32-86*
CLOSE    32-?32   32-?37   32-?49#
CLR       4-108#  32-34    32-599
CLRQUE   34-17    37-75    38-130#
CNTCLR   11-3     12-30    14-84    14-93    14-103   14-105   14-177   14-195   14-197   14-254   14-266   14-277   14-279   14-340
        14-352   14-365   14-367   14-467   14-515   14-544   14-593   32-33#
CNTRLC    7-0#    10-19*   10-26*   10-98    11-4
COLON    41-28#
COMMA    11-65    12-49    14-583   32-19    32-740   32-750   32-765   41-27#
CONT      4-178#
COUNT    14-95    14-179   14-257   14-269   14-343   14-355   32-648#
CPSAVE   17-1     17-1     17-1     17-1     17-1     17-1#    17-1*    17-1*    18-111   23-1     23-1     23-1     23-1*    23-1*
CR        4-80#   19-1     19-1
CRLF      4-80#   10-6     10-37    10-37    10-58    10-69    10-75    14-585   15-1     15-1     15-1     19-1     19-1     32-;53
        32-<17   39-53    39-54    41-14    41-17    41-25    41-29    41-31    41-32    41-34    41-35    41-35    41-36    41-36
        41-38    41-39    41-39    41-40    41-40    41-41    41-42    41-43    41-44    41-46    41-47    41-48
CYINC     7-0#    14-418*  14-419   14-436   14-488   14-541*
CYL.DS    7-0#    32-233*  32-264*  32-305*  32-440*  32-571*  32-611*  45-8     45-9     45-10    45-13    45-15    45-21    45-24
CYL.RD    7-0#    32-565*  45-9
DCK       4-166#
DDISP     4-80#    6-[      10-32
DECSEC   32-820#
DECSK    13-127#  13-131
DELTA     7-0#
DF1       8-14    46-3#
DF10      8-75    46-19#
DF11      8-86    46-23#
DF12      8-100   46-27#
DF13      8-114    8-137   46-34#
DF14      8-123    8-146   46-41#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
EM1       8-11    42-3#
EM10      8-72    42-6#
EM11      8-83    42-7#
EM12      8-97    42-8#
EM13      8-111    8-134    8-179    42-9#
EM17      8-154   42-10#
EM2       8-22    42-4#
EM20      8-165   42-11#
EM23      8-201   42-12#
EM24      8-214   42-13#
EM41      8-242    8-253    8-266    8-282    8-298    42-14#
EM46      8-314   42-15#
EM5       8-47    42-5#
EMPTYQ   35-258   36-54    36-223   36-231   36-287   37-111   38-148#
EMTVEC    4-80#   10-32*   10-32*
ERINDX   32-233   32-264   32-305   32-440   32-479#  32-611
ERMAX     7-0#    32-13
ERR       4-146#  14-82    14-91    14-175   14-252   14-264   14-338   14-350
ERR.CT    7-0#    12-69
ERRCN     7-0#    10-84*   10-87*   10-87*   10-87*   10-87*   10-87*   10-87*   10-87*   32-12*   32-13
ERROR     4-80#
ERRVEC    4-80#   10-32    10-32*   10-32*   10-34*   10-35*   17-1     17-1*    17-1*    23-1     23-1     23-1*    23-1*    23-1*
          23-1*   23-1*    32-47*   32-48*   32-53    32-53*   32-73    39-36    39-37*   39-39*   39-45*
ES.SAV   36-158   36-196   36-198   38-205#
ESRC      4-183#
EXCEED   32-20    41-34#
EXIT.A   14-128   14-128#
EXIT0    13-70    13-70#
EXIT1    13-92    13-96#
EXIT10   13-392   13-403#
EXIT11   14-29    14-54    14-54    14-54    14-128   14-129#
EXIT12   14-146   14-161   14-161   14-161   14-205#
EXIT13   14-240   14-240   14-240   14-240   14-287#
EXIT14   14-311   14-326   14-326   14-326   14-375#
EXIT15   14-394   14-408   14-408   14-408   14-452   14-500   14-583   14-595#
EXIT2    13-117   13-132#
EXIT3    13-149   13-160#
EXIT4    13-182   13-226#
EXIT5    13-247   13-260   13-263#
EXIT6    13-282   13-290   13-305#
EXIT7    13-324   13-365   13-367#
F1        4-128#
F2        4-129#
F3        4-130#
F4        4-131#
F5        4-132#
FC        9-7#    13-70    13-70    13-70    13-86    13-118   13-130   13-150   13-183   13-192   13-224   13-248   13-261   13-283
         13-326   13-327   13-334   13-387   13-388   13-399   14-54    14-54    14-161   14-163   14-188   14-240   14-259   14-326
         14-345   14-408   14-437   14-535   14-538
FER       4-155#
FMT16     4-260#
FS        9-13#   13-47    13-82    13-112   13-144   13-173   13-242   13-277   13-383   13-384   14-54
FT        9-10#   13-48    13-84    13-113   13-145   13-174   13-243   13-278   13-318   13-366   13-385   13-386   14-54
FWD      41-42#
GETADR   10-82    39-13#
GETNUM   32-;56   32-;78#  39-23    39-30
```

CROSS REFERENCE TABLE (CREF V04.00)

```
GETREG    4-307#
GETREQ    35-69     35-248    36-206    36-284    37-51     37-65     37-100    37-107    38-178#
GETSWR    10-90     32-;51#
GNS        5-1       5-1      10-6      10-37     10-58     10-69     10-75     15-1      15-1      15-1      15-1      18-52     18-66     25-1
          25-1      25-1      25-1      25-1      25-1      25-1      25-1      25-1      25-1      25-1      25-1      25-1      25-1      25-1
          25-1      25-1      25-1      25-1      25-1      25-1      25-1      32-;53    32-<17    32-<33    32->05
GO         4-127#
GT.PR1    32-<16#   32-<20    32-<20    32-<20    32-<20    32-<20    32-<22    32-<22
GT.PR2    32-<30#   32-<41    32-=69    32-=79    32-?42
GT.PRM    11-12     32-<15#
GTSWR     10-37     25-1#
GTTST1    32-<28    32-<33#   32-<72    32-<85    32-<99    32-=02    32-=29    32-=48    32-=57    32-=64    32-=74    32-=94
GTTST2    32-<47#   32-=52
GTTST3    32-<50    32-<55    32-=22#
GTTST4    32-<58    32-<82    32-=19    32-=23#
GTTST5    32-=24    32-=59#
GTTST6    32-=61    32-=71#
HCE        4-158#   32-268    32-309
HCI        4-258#
HCRC       4-159#
HT         4-80#    19-1      19-1
IAE        4-161#
IBSAVE    17-1      17-1      17-1      17-1      17-1      17-1#     17-1*     17-1*     17-1*
IC         9-9#     13-122    13-129    13-153    13-202    13-223    13-255    13-256    13-303    13-396    13-400    32-807
IE         4-84#
ILF        4-151#
ILR        4-152#
INCCYL    32-805#
INCEC     18-41     32-6#
INCSK     13-120#   13-124
INCTRK    32-788#
IOTVEC     4-80#    10-32*    10-32*
IR         4-109#
ISR       14-108    14-199    14-281    14-369    14-445    14-472    14-493    14-520    14-546    32-597    34-32     36-3#
IT         9-12#    32-790
ITEM41     8-234#   18-37
IVC        4-278#
LBC        4-276#
LC         9-8#     13-49     13-70     13-70     13-87     13-123    13-125    13-158    13-203    13-205    13-213    13-249    13-257    13-284
          13-327    13-332    13-397    13-401    14-183    14-247    14-333    14-440    14-485    14-487    14-537    32-805    32-808    32-810
LDCMD     12-36     32-209#
LF         4-80#    15-1      19-1      19-1      41-38     41-39     41-40     41-41     41-44
LKS        7-0#     32-85     32-117*
LKV        7-0#     32-115*   32-116*
LODEV     10-127    41-23#
LODFLT    11-11     11-20     32-130#
LODPRM    13-44     13-79     13-109    13-141    13-170    13-239    13-274    13-316    13-380    14-23     14-140    14-221    14-305    14-388
          32-182#
LOP.CK    32-244    32-277    32-318    32-443    32-453#
LP.AVL    10-88     32-48#
LPB        7-0#     17-1
LPS        7-0#     17-1      32-49
LPTAVL     7-0#     17-1      32-46*    32-50*
LS         9-14#    13-83
LSC        4-277#
LSIT       4-171#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
LST       4-142#
LSTRK     7-0#      32-197*
LT        9-11#     13-85     13-362    32-788    32-791    32-793
M.DP40   31-25#     31-46
M.DP41   31-30      31-37#
M.DP42   31-36      31-41#
M.DP44   31-48      31-53#
M.DP50   31-23      31-57#
M.DPID   14-583     30-31     31-14#
MABOVE   32-752     41-51#
MBELOW   32-742     41-50#
MCLK      4-204#
MCPE      4-89#
MDF       4-199#
MEASUR   14-583     41-44#
MI        4-195#
MOC       4-201#
MOH       4-236#
MOL       4-144#
MPE       4-111#
MRD       4-203#
MRMCS1   39-21      39-53#
MRMVEC   39-28      39-54#
MS        4-198#
MSC       4-194#
MSDRIV   12-47      32-16     41-32#
MSEN      4-205#
MSER      4-200#
MSG.CS   32-;54     41-14#
MSG.EQ   32->36     41-13#
MSG.FC    9-41      41-4#
MSG.FS    9-47      41-10#
MSG.FT    9-44      41-7#
MSG.IC    9-43      41-6#
MSG.IT    9-46      41-9#
MSG.LC    9-42      41-5#
MSG.LS    9-48      41-11#
MSG.LT    9-45      41-8#
MSG.PA   41-12#
MSG.R     9-40      41-3#
MSGOUS   14-583     32-697    32-702    32-737    32-744    32-747    32-754    32-764    41-49#
MSG11X    7-0        7-0      41-59#
MSG12X    7-0       41-60#
MSG13X    7-0       41-61#
MSG14X    7-0       41-62#
MSGAVG   14-583     32-755    41-48#
MSGB01    7-0       43-10#
MSGB02    7-0       43-9#
MSGB04    7-0       43-8#
MSGB05    7-0       43-7#
MSGB06    7-0       43-6#
MSGB08    7-0       43-5#
MSGB09    7-0       43-4#
MSGB14    7-0       43-3#
MSGMAX   32-698     32-745    41-47#
MSGMIN   32-694     32-735    41-46#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
MSGNOT  32-777  41-54#
MSGSCH  32-771  41-52#
MSGSEK  14-583  32-769  41-53#
MSGTST  13-45   13-80   13-110  13-142  13-171  13-240  13-275  13-317  13-381  14-24   14-141  14-222  14-306  14-389
        41-31#
MUR      4-202#
MWP      4-196#
MXF      4-112#
MXSEEK   7-0    41-41#
MXSTAL   7-0#   13-199  13-220
MXWNDW  33-173# 35-159
NBA      4-238#
NC1      9-17#  14-436* 14-437* 14-438  14-449  14-483* 14-489* 14-490* 14-508  14-531  14-532* 14-533  14-535
NC2      9-18#  14-438* 14-439* 14-440  14-460  14-483  14-484* 14-485  14-487* 14-488* 14-489  14-497  14-531*
NED      4-115#
NEM      4-114#
NOASSY   4-9#    4-44    4-57    9-182   11-36   32-139  32-331  32-826  32->28  32->59
NOCLOK  11-75   41-29#
NODRVS  12-10   41-35#
NONE    11-48   41-26#
NOTEST  11-17   41-36#
NOTPRS  10-114  41-20#
NOTRM   10-111  41-22#
NOTSAF  10-120  41-21#
OBCK     4-208#
OBEN     4-207#
OCC      4-189#
OFD      4-256#
OFFSET   4-296#
OM       4-137#
ONECYL   7-0    41-39#
OPE      4-279#
OPI      4-164#
OPN.1   32-=88# 32->04  32-?33
OPN.2   32-=86  32-=92  32-=96#
OPN.CT  32-=85# 32-=88* 32-=91  32-=96  32-=98  32->06
OPN.N1  32->55  32-?30#
OPN.N2  32->47  32-?31#
OPN.X1  32->55  32-?35#
OPN.X2  32->52  32-?36#
OPNFLG   7-0#   32-<44* 32-<45* 32-=37* 32-=41* 32-=43* 32-=53  32-=55  32-=65  32-=67  32-=75  32-=77  32->03
OPNPRM  32->14  32->20# 32->24
OPNTST  32-=54  32-=56  32-=66  32-=68  32-=76  32-=78  32-=83#
OPT     35-34   35-67#  36-38   36-247  36-290
OR       4-110#
PACK     4-299#
PAR      4-154#
PAT      9-15#  32-195
PDA      4-180#
PERFEX  36-69
PFECH   18-29   18-107#
PFECH1  18-107  18-108#
PFECH2  18-107  18-109#
PFECH3  18-107  18-111#
PFECH4  18-107  18-112#
PFTSTN  18-27*  18-111  18-115#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
PGE       4-113#
PGM       4-141#
PHA       4-179#
PIP       4-145#
PIRQ      4-80#
PIRQVE    4-80#
PKB       7-0#     14-77*   14-86*   14-169*  14-246*  14-258*  14-332*  14-344*  14-459*  14-507*  32-107*
PKC       7-0#     14-476   14-524   32-652   32-654   32-655   32-659   32-661   32-662   32-666   32-671
PKCS      7-0#     14-78*   14-81*   14-88*   14-90*   14-99*   14-172*  14-174*  14-191*  14-249*  14-251*  14-261*  14-263*  14-273*
         14-335*   14-337*  14-347*  14-349*  14-361*  14-455*  14-462*  14-464*  14-503*  14-510*  14-512*  14-589*  32-79    32-108*
PKV       7-0#     14-75*   14-167*  14-244*  14-330*  14-456*  14-504*  32-105*  32-106*
PLFS      4-182#
POPQUE   35-80     35-131   35-220   36-241   38-190#
PR0       4-80#
PR1       4-80#
PR2       4-80#
PR3       4-80#
PR4       4-80#
PR5       4-80#
PR6       4-80#    10-35
PR7       4-80#
PRM       9-4#     32-187   32-190*  32->10   32->20   32-?50
PRM0      9-21      9-71#   32-134
PRM1      9-24      9-79#
PRM10     9-24      9-143#
PRM11     9-24      9-152#
PRM12     9-24      9-159#
PRM13     9-24      9-165#
PRM14     9-24      9-171#
PRM15     9-24      9-177#
PRM2      9-24      9-90#
PRM3      9-24      9-99#
PRM4      9-24      9-108#
PRM5      9-24      9-117#
PRM6      9-24      9-126#
PRM7      9-24      9-135#
PRMLMT    9-28#    13-352   32-148*  32-149*  32-152*  32-153*  32-169   32-171   32-197   32-823   32->55
PRMMSG    9-40#    32->33
PRMPT     9-21#    32-155   32-186   32->08   32-?51
PS        4-80      4-80#   10-95*   12-31*   14-100*  14-192*  14-274*  14-362*  14-590*  32-596*  34-15    34-16*   34-45*   34-53*
          35-11    35-12*   35-58*   35-68    35-114*
PSEL      4-88#
PSW       4-80#
PWRVEC    4-80#
QDRV     33-95#    38-131   38-149*  38-163   38-165*  38-179   38-191   38-192*
R6        4-80#    10-32    10-32*   10-32*
R7        4-80#
RD.ADR   38-11*    38-12*   38-14#
RD.RM    34-83     34-96    34-100   35-144   35-187   35-198   35-208   36-25    36-44    36-47    36-50    36-103   36-121   36-155
         36-208    36-212   37-41    37-86    38-10#   38-42    38-81
RD.RM1   38-13#
RD.RM3   38-20     38-23#
RD.RM4   38-22     38-25#
RD.WRD   38-15#    38-16
RDCHR    22-1      25-1#
RDLIN    25-1#     32-;83   32-<18   32-<34   32->39
```

CROSS REFERENCE TABLE (CREF V04.00)

```
RDY      4-85#
READ     4-305#
READHD   4-306#  32-211  32-212  32-279  32-320
READIN   4-298#
RECAL    4-293#  12-44   12-44   13-46   13-46   32-574  32-602  40-105
RELEAS   4-295#
RESREG   14-84   14-93   14-103  14-177  14-195  14-254  14-266  14-277  14-340  14-352  14-365  14-467  14-515  14-583
         14-593  18-105  25-1#   27-1    30-34   32-635  32-<30  32-=93  32-?41  32-?67  34-54   35-52   35-55   35-115
         35-262  36-10   37-22   38-100  38-139
RESVEC   4-80#
REV      7-0     7-0     7-0     41-43#
REX      4-184#
RH.ADR   7-0#    39-20   39-38   39-40   45-6
RHVEC    7-0#
RM.REG   12-52   14-469  14-517  32-235  32-237  32-240  32-266  32-268  32-270  32-273  32-307  32-309  32-311  32-314
         32-504  32-506  32-820  40-15   40-40   40-65   40-90   40-115  40-130#  45-12   45-12   45-12   45-12   45-12
         45-15   45-15   45-15   45-16   45-16   45-16   45-16   45-16   45-18   45-18   45-18   45-19   45-19   45-19
         45-20   45-20   45-20   45-22   45-22   45-22   45-22   45-22   45-22   45-23   45-23   45-23   45-25   45-25
         45-25   45-25   45-25   45-25   45-26   45-26   45-26   45-26   45-26   45-26
RM80     32-226  32-257  32-298  32-433  32-604  35-11#
RMADR    13-324  32-7    32-33   32-598  33-165#  34-34   35-18   35-134  35-154  35-172  36-5    37-39   38-11   38-17
         38-48   38-52   38-70   38-111  39-41
RMAS     33-184#  34-99*  36-153  36-246*  36-268*  37-77
RMBA     33-179#  36-117
RMCS1    14-248*  14-260*  14-334*  14-346*  14-461*  14-509*  33-177#  34-75*  34-81   34-93   35-72   35-87   35-148  35-166
         35-209  35-219  35-230  36-25   36-126  36-155  36-216  36-226  37-86   38-42
RMCS2    13-342*  32-8    32-34*  32-36*  32-599*  33-181#  34-35*  34-74*  34-76   35-29*  35-135*  35-155*  35-173*  36-24*
         36-207*  36-252*  36-267*  37-40*  37-54*  38-18   38-53   38-71*  38-77   38-112*  38-117
RMDA     14-54*  32-614*  32-820  33-180#  35-141  35-164  35-178
RMDB     33-186#  38-75
RMDC     14-54*  14-170*  14-247*  14-259*  14-333*  14-345*  14-460*  14-508*  32-615*  33-191#  35-143  35-157  35-183  35-194
RMDS     14-82   14-91   14-175  14-252  14-264  14-338  14-350  14-465  14-513  33-182#  34-96   35-74   36-208  36-253
         37-41
RMDT     33-188#  34-83
RMEC1    33-195#
RMEC2    33-196#  38-85
RMER1    32-235  32-266  32-268  32-307  32-309  32-504  33-183#  34-100  36-44   36-212  36-254
RMER2    14-469  14-517  32-237  32-240  32-270  32-273  32-311  32-314  32-506  33-194#  36-47   36-255
RMERRS   33-43#  34-18   36-193  36-253*  36-254*  36-255*  45-4    45-4    45-4    45-4    45-5    45-5    45-5    45-5
RMHR     33-192#
RMINIT   10-93   34-14#
RMLA     13-343  33-185#
RMMR1    33-187#
RMMR2    33-193#
RMOF     33-190#  34-95   35-144  35-146  35-187  35-189  35-198  35-200  36-50   36-121  36-123
RMR      4-153#
RMSN     33-189#
RMTMR    32-121  37-6#
RMVEC    14-76*  14-108*  14-168*  14-199*  14-245*  14-281*  14-331*  14-369*  14-445*  14-457*  14-472*  14-493*  14-505*  14-520*
         14-546*  32-597*  33-166#  34-31   34-33   35-12
RMWC     33-178#  36-103  36-119
ROTATE   7-0     7-0     41-38#
RPT      9-6#    13-44   13-79   13-109  13-141  13-170  13-239  13-274  13-316  13-380  14-23   14-140  14-221  14-305
         14-388  32->37  32->56*  32-?30*  32-?35*
RSTART   12-5#   12-28   15-6
RSTRT1   11-76   12-13#  12-19
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
SW6       4-80#
SW7       4-80#
SW8       4-80#
SW9       4-80#    32-453   32-577
SWR       6-0#     10-32    10-32    10-32*   10-32*   10-32*   10-37    13-45    13-80    13-110   13-142   13-171   13-240   13-275
         13-317    13-381   14-24    14-141   14-222   14-306   14-389   14-424   17-1     17-1     17-1     17-1     17-1     17-1
         22-1      22-1     22-1*    23-1     23-1     23-1     23-1     23-1*    23-1*    23-1*    23-1*    32-453   32-577   32-685
         32-724    32-:51
SWREG     5-1#     10-32    10-37    22-1     22-1     22-1
T11A      7-0#     14-71    14-71    14-128
T11B      7-0#     14-71    14-71    14-128
T12       7-0#     14-161   14-204
T13       7-0#     14-240   14-286
T14       7-0#     14-326   14-374
TAB.XY    5-13#     6-0
TAP       4-237#
TBITVE    4-80#
TD       36-8      36-16#
TEST0    13-44     13-70    13-70#
TEST1    13-79     13-92    13-92#
TEST10   13-380    13-392#
TEST11   14-23     14-71#
TEST12   14-140    14-161#
TEST13   14-221    14-240#
TEST14   14-305    14-326#
TEST15   14-388    14-408#
TEST2    13-109    13-117#
TEST3    13-141    13-149#
TEST4    13-170    13-182   13-182#
TEST5    13-239    13-247#
TEST6    13-274    13-282#
TEST7    13-316    13-324   13-324#
TICKMS    7-0#     32-95*   32-98*   32-120
TICKUS    7-0#     32-96*   32-99*
TIM.DN    7-0#     14-356*  14-357*  32-634*  32-651
TIM.PT    7-0#     32-630   32-669   32-671*  32-672*
TIM.UP    7-0#     32-628   32-633*  32-648   32-730
TIMER    33-145#   35-108*  35-235*  35-252*  36-19*   36-204*  36-265*  36-272*  37-12    37-14*   37-68*   37-93*   37-99*   37-105*
TKVEC     4-80#    22-1*    22-1*
TP50     14-71*    14-128#
TP60     14-71*    14-128#
TPB       7-0#     10-44    17-1
TPS       7-0#     10-43    17-1
TPS50    14-71*    14-128#
TPS60    14-71*    14-128#
TPVEC     4-80#
TRAPVE    4-80#    10-32*   10-32*
TRCKWC    7-0#     32-150*  32-154*
TRE       4-90#
TRK.DS    7-0#     32-233*  32-264*  32-305*  32-440*  32-570*  32-611*  45-8     45-9     45-10    45-13    45-21    45-24
TRK.RD    7-0#     32-566*  45-9
TRNSWT   33-104#   35-132*  35-245   35-257*  36-21    36-22*   36-66*   37-48    37-62    37-74*
TRTVEC    4-80#
TST0     13-44#
TST1     13-44     13-79#
TST10    13-316    13-380#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
$$CMRE    5-198#  6-0     6-0     6-0     6-0     6-0     6-0
$$CMTM    5-198#  6-0     6-0     6-0
$$ESCA    4-80#
$$NEWT    4-80#   13-44   13-79   13-109  13-141  13-170  13-239  13-274  13-316  13-380  14-23   14-140  14-221  14-305
          14-388
$$SET     25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1    25-1#
$$SETM    10-32   10-32#
$$SKIP    4-80#
.$ACT1    4-35#   5-9
.$APTB    4-36#   6-0     6-0#
.$APTH    4-36#   5-12
.$APTY    4-36#   16-1
.$CATC    4-33#   5-1
.$CMTA    4-33#   5-198
.$DB2D    4-35#   27-1
.$EOP     4-33#   15-1
.$ERRO    4-33#   17-1
.$RAND    4-35#   29-1
.$RDDE    4-34#
.$RDOC    4-34#
.$READ    4-34#   22-1
.$SAVE    4-35#   24-1
.$SB2D    4-35#   26-1
.$SCOP    4-35#   23-1
.$SUPR    4-35#   28-1
.$TRAP    4-33#   25-1
.$TYPD    4-34#   21-1
.$TYPE    4-33#   19-1
.$TYPO    4-34#   20-1
.EQUAT    4-33#   4-80
.HEADE    4-34#   4-40
.SETUP    4-34#   4-315
.SWRHI    4-33#   4-41
.SWRLO    4-41#   4-42    4-43
CKCHR     9-589#  32-;85   32-<20  32->41  32-a56  32-a63  32-a68  32-A13
CKDIG     9-599#  32-<22   32->55
CKNUM     9-609#  32-;87
COMMEN    4-80#
COMND     9-530#  12-44   13-46   13-319  14-54   14-161  14-240  14-326  14-408  14-447  14-495
DO        9-537#  12-43   12-45   13-70   13-70   13-94   13-95   13-121  13-128  13-152  13-156  13-187  13-193  13-208
          13-214  13-252  13-254  13-292  13-294  13-296  13-298  13-300  13-302  13-340  13-360  13-394  13-395  14-54
          14-161  14-240  14-326  14-408  14-453  14-501  32-575
DODTA     9-550#
ENDCOM    4-80#
ENDPAS    14-597# 15-1
ER.NDX    9-554#  32-233  32-264  32-305  32-440  32-611
ERRCAL    33-1#   36-158  36-196  36-198
ERREND    9-582#  17-1
ERROR     4-80#   12-25   14-85   14-94   14-104  14-178  14-196  14-255  14-267  14-278  14-341  14-353  14-366  14-468
          14-516  14-594  23-1    32-233  32-233  32-233  32-233  32-239  32-264  32-264  32-264  32-264  32-272  32-305
          32-305  32-305  32-305  32-313  32-440  32-440  32-440  32-440  32-573  32-611  32-611  32-611  32-611  32-611
          39-47
ERRTYP    9-573#  17-1
ESCAPE    4-80#   14-54   14-161  14-240  14-326  14-408  32-232  32-263  32-304  32-439  32-572  32-610
EXIT      4-11#   13-70   13-96   13-132  13-160  13-226  13-263  13-305  13-367  13-403  14-129  14-205  14-287  14-375
          14-595
```

CROSS REFERENCE TABLE (CREF V04.00)