RP04/05/06

HEAD ALIGN VERIFICATION

MD-11-DZRJC-A

EP-DZRJC-A-DL-A

FICHE 1 OF 1

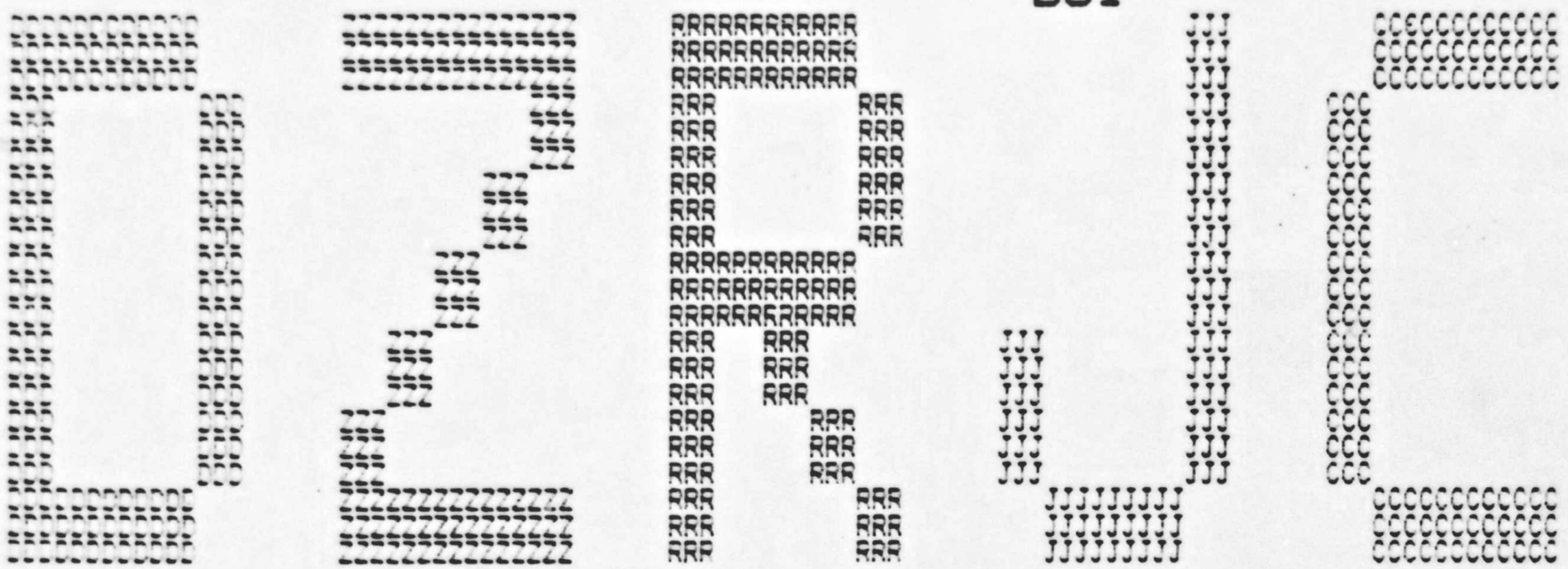
NOV 1976

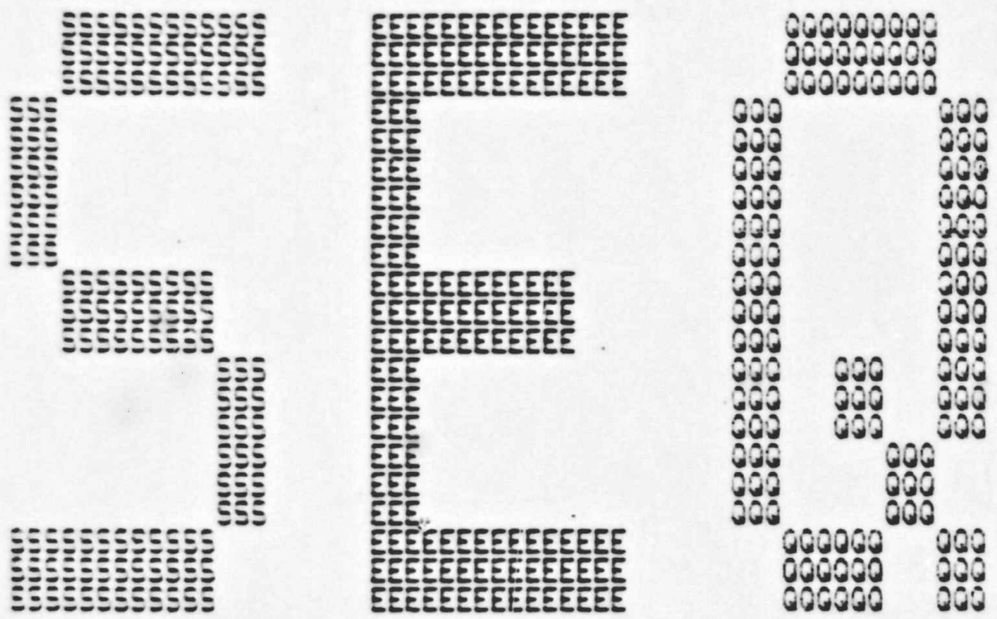
digital

MADE IN USA

DZRJC
SEQ

B01





LPTSPL Version 101(2111) Running on: MTA140
 START User WELLINGHAM,AI400,3722) Job DZRJC Seq. 6671 Date 11-May-76 16:33:58 Monitor RV242A KI10 SYS#514 *START*
Request created: 11-May-76 16:23:57
File: RP58:DZRJC.SEG<055>[401,3722] Created: 03-May-76 10:15:07 Printed: 11-May-76 16:34:26
QUEUE Switches: /PRINT:ARROW /FILE:ASCII /COPIES:1 /SPACING:1 /LIMIT:2000 /FORMS:NORMAL

```
        .REM    %
```

I D E N T I F I C A T I O N

PRODUCT CODE: MAINDEC-11-DZRJC-A-D

PRODUCT NAME: RP04/5/6 HEAD ALIGNMENT VERIFICATION PROGRAM

DATE CREATED: MAY 1976

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: C. HESS

CONTENTS

1. ABSTRACT

THE RP04/5/6 HEAD ALIGNMENT VERIFICATION PROGRAM CHECKS RP04, RP05, OR RP06 DISK DRIVE HEAD ALLIGNMENT OR ALLOWS THE OPERATOR TO ALIGN HEADS WITH THE APPROPRIATE DRIVE'S TEST BOX. THE PROGRAM ALSO CONTAINS A UTILITY ROUTINE WHICH PERFORMS 5,000 RANDOM SEEKS WITHOUT DATA TRANSFERS. THE RANDOM SEEK UTILITY ALLOWS THE OPERATOR TO EXERCISE THE DRIVE WITH THE ALIGNMENT PACK IN PLACE AND THEN RE-VERIFY HEAD ALIGNMENT.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
8K MEMORY
TELETYPE
PROGRAM LOAD DEVICE
KW11-L OR KW11-P CLOCK
RH11 OR RH70 WITH RP04'S, RP05'S, OR RP06'S
ALIGNMENT PACK ('CE' PACK)
'DDU' OR 'DEDU' FOR RP04'S
'PERCH' FOR RP05/6'S

2.2 PRELIMINARY PROGRAMS

THE RP04/5/6 DISK SUBSYSTEM MUST BE OPERATION AND ALL OTHER PROGRAMS IN THE SET FOR THE SUBSYSTEM MUST RUN SUCESSFULLY.

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 ASSOCIATED LOADER. THE PROGRAM MAY NOT BE INCLUDED IN A 'XXDP' CHAIN.

4. STARTING PROCEDURES

4.1 STARTING ADDRESSES

THE PROGRAM IS STARTED FROM LOCATION 200(8) IF THE ADDRESS OF THE RH11 OR RH70 WILL NOT BE CHANGED.

THE PROGRAM IS STARTED FROM LOCATION 204(8) IF THE ADDRESS

OF THE RH11 OR RH70 IS TO BE CHANGED FROM THE PRELOADED VALUE. (SEE SECTION 4.4)

4.2 PROGRAM & OPERATION ACTION

1. LOAD THE PROGRAM INTO MEMORY (SEE SECTION 3)

2. LOAD ADDRESS 200 OR 204.

3. SET THE SWITCHES AND PRESS 'START'

4. PLACE AN ALIGNMENT DISK PACK ON THE DRIVE TO BE CHECKED AND SET THE DRIVE IN WRITE PROTECT MODE.

5. CYCLE UP THE DRIVE TO BE CHECKED. ALLOW SUFFICIENT TIME FOR THE ALIGNMENT PACK TO REACH OPERATING TEMPERATURE AND TO STABILIZE. REFER TO APPLICABLE MAINTENANCE PROCEDURES FOR THE REQUIRED AMOUNT OF TIME.

6. SELECT THE DRIVE TO BE CHECKED IN RESPONSE TO THE TYPEOUT ON THE TELETYPE.

7. THE PROGRAM WILL ASK FOR THE MODE OF OPERATION: ENTER AN 'A' IF THE PROGRAM IS TO BE USED FOR HEAD ALIGNMENT; ENTER A 'V' IF HEAD ALIGNMENT IS TO BE CHECKED; OR ENTER AN 'E' IF THE RANDOM SEEK UTILITY IS TO BE RUN.

8. IF AN 'A' IS ENTERED IN RESPONSE TO THE PROGRAM MODE MESSAGE, THE PROGRAM WILL POSITION THE DRIVE TO THE ALIGNMENT CYLINDER: RP04/5 - CYLINDER 245; RP06 - CYLINDER 496. THE PROGRAM WILL THEN ASK FOR A HEAD ADDRESS. THE HEAD ENTERED WILL BE SELECTED AND THE PROGRAM WILL REQUEST ANOTHER HEAD. UNTIL A NEW HEAD ADDRESS IS ENTERED, THE LAST ENTERED HEAD ADDRESS WILL REMAIN SELECTED. THE ALIGNMENT SEQUENCE MAY BE TERMINATED BY TYPING A 'CONTROL C'; THE PROGRAM WILL RETURN TO THE DRIVE SELECTION ROUTINE.

9. IF THE PROGRAM IS BEING RUN IN 'VERIFICATION' MODE AND SW<02> IS SET, THE PROGRAM WILL ASK FOR HEAD AND CYLINDER ADDRESSES. (HEAD AND CYLINDER ADDRESSES ARE ENTERED IN DECIMAL.) WITH SW<02>, ONLY THE ALIGNMENT OF THE SPECIFIED HEAD WILL BE CHECKED.

10. WHEN THE OPERATOR HAS COMPLETED THE PRESENT DRIVE, CYCLE THE DRIVE DOWN, AND TRANSFER THE TEST BOX AND ALIGNMENT PACK TO THE NEXT DRIVE TO BE CHECKED. WHEN THE ALIGNMENT PACK HAS STABILIZED, THE NEW DRIVE CAN BE SELECTED. IT IS NOT NECESSARY TO RESTART THE PROGRAM.

4.3 DRIVE SELECTION

ENTER A DRIVE NUMBER IN RESPONSE TO THE 'ENTER DRIVE NUMBER:' TYPEOUT FROM THE PROGRAM. ONLY VALID DRIVE NUMBERS (0 - 7) WILL BE ACCEPTED BY THE PROGRAM. NOTE THAT THE DRIVE SETUP

(TEST BOX CONNECTION AND ALIGNMENT PACK STABILIZATION) MUST HAVE
BEEN COMPLETED BEFORE A DRIVE IS SELECTED.

4.4 RH11 - RH70 UNIBUS ADDRESSES

THE PROGRAM ASSUMES THAT THE RH11 OR RH70 ADDRESSES START AT
176700 AND THAT THE VECTOR ADDRESS IS 254. THESE ADDRESSES MAY
BE CHANGED WHEN THE PROGRAM IS STARTED FROM LOCATION 204. ENTER
THE RH11/RH70 ADDRESS IN RESPONSE TO THE REQUEST FROM THE PROGRAM.

4.5 OTHER UNIBUS ADDRESSES

LOC	TAG	CONTENTS	FUNCTION
1144	$TKS	177560	TTY KEYBOARD STATUS REGISTER
1146	$TKB	177562	TTY KEYBOARD BUFFER REGISTER
1150	$TPS	177564	TTY PRINTER STATUS REGISTER
1152	$TPB	177566	TTY PRINTER BUFFER REGISTER
1244	PKV	104	KW11-P CLOCK VECTOR ADDRESS
1246	PKCS	172540	KW11-P CONTROL REGISTER
1250	PKB	172542	KW11-P COUNT SET REGISTER
1252	PKC	172544	KW11-P COUNTER REGISTER
1254	LKV	100	KW11-L CLOCK VECTOR ADDRESS
1256	LKS	177546	KW11-L STATUS REGISTER

E. SWITCH REGISTER SETTINGS

SW<15>=1...HALT ON ERROR
SW<13>=1...INHIBIT ERROR TYPEOUTS
SW<09>=1...LOOP ON ERROR
SW<02>=1...CHECK ALIGNMENT OF THE SPECIFIED HEAD
SW<01>=1...LOOP ON THE CURRENT HEAD
SW<00>=1...TYPEOUT ALL TRACK CENTER VALUES

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34)
THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS
NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE
'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE
SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD
ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL
RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM
IS AT A HIGHER PRIORITY PROCESSING AN RP04/5/6 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 SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER
IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND
'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS

DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

5. ERRORS

5.1 TEST ERRORS

RP04/5:

THE PROGRAM CHECKS THAT EACH HEAD IS WITHIN + OR - 150 MICRO-INCHES FROM THE TRACK CENTERLINE FOR CYLINDER 245 AND IS WITHIN + OR - 350 MICRO-INCHES FOR CYLINDERS 4 AND 400. IF A HEAD IS FOUND THAT IS NOT WITHIN THESE TOLERANCES, AN ERROR MESSAGE IS TYPED. THE ERROR MESSAGE IDENTIFIES THE CYLINDER, HEAD, AND THE ACTUAL POSITION OF THE HEAD RELATIVE TO THE CENTERLINE. (IF SW<00>=1, THE ACTUAL ALIGNMENT OF EACH HEAD WILL BE TYPED OUT; HEADS OUT OF TOLERANCE WILL NOT BE IDENTIFIED.)

RP06:

THE PROGRAM CHECKS THAT EACH HEAD IS WITHIN + OR - 75 MICRO-INCHES FROM THE TRACK CENTERLINE FOR CYLINDER 496 AND IS WITHIN + OR - 175 MICRO-INCHES FOR CYLINDERS 8 AND 800. IF A HEAD IS FOUND THAT IS NOT WITHIN THESE TOLERANCES, AN ERROR MESSAGE IS TYPED. THE ERROR MESSAGE IDENTIFIES THE CYLINDER, HEAD, AND THE ACTUAL POSITION OF THE HEAD RELATIVE TO THE CENTERLINE. (IF SW<00>=1, THE ACTUAL ALIGNMENT OF EACH HEAD WILL BE TYPED OUT; HEADS OUT OF TOLERANCE WILL NOT BE IDENTIFIED.)

6.2 ENTRY ERRORS

THE PROGRAM WILL NOT ACCEPT DRIVE NUMBERS GREATER THAN 7. IF AN INVALID DRIVE NUMBER IS ENTERED IN RESPONSE TO THE DRIVE NUMBER REQUEST, THE PROGRAM WILL TYPE A '?'; THE OPERATOR MAY ENTER A VALID NUMBER.

THE PROGRAM WILL NOT ACCEPT HEAD ADDRESSES GREATER THAN 18 (DECIMAL) OR CYLINDER ADDRESSES OTHER THAN 245(10) FOR RP04/5'S OR 496(10) FOR RP06'S. IF AN INVALID CYLINDER OR HEAD ADDRESS IS ENTERED, THE PROGRAM WILL REJECT THE ENTRY AND RETURN TO THE DRIVE SELECTION ROUTINE.

THE PROGRAM WILL NOT ALLOW A DRIVE TO BE ASSIGNED IF IT IS NOT 'WRITE PROTECTED'. IF THIS IS ATTEMPTED, THE OPERATOR WILL BE NOTIFIED.

DRIVES ASSIGNED ARE CHECKED TO ENSURE THAT THE DRIVE ASSIGNED IS PRESENT, ONLINE, AND IS AN RP04, RP05, OR RP06. THE PROGRAM WILL REJECT ASSIGNMENTS FOR DRIVES WHICH ARE NOT PRESENT OR ARE NOT ONLINE.

6.3 SUBSYSTEM/DEVICE ERROR MESSAGES

1. 'RH11 INTERRUPT OCCURRED (RPAS=0) - AN INTERRUPT OCCURRED, BUT NOTHING ON THE MASSBUS IS INDICATING AN ATTENTION.

2. 'UNEXPECTED ATTENTION OCCURRED' - THE INDICATED DRIVE INTERRUPTED, BUT NO INTERRUPT WAS EXPECTED FROM THE INDICATED DRIVE.

3. 'MASSBUS PARITY ERROR (MCPE=1)' - A CONTROL BUS PARITY ERROR WAS DETECTED BY THE RH11 WHEN THE INDICATED REGISTER WAS READ.

4. 'MASSBUS PARITY ERROR (PAR=1)' - A CONTROL BUS PARITY ERROR OCCURRED WHEN THE INDICATED REGISTER WAS WRITTEN.

5. 'ADDRESS PLUG CHANGE BIT SET' - THE PROGRAM FOUND THE 'OPE' BIT SET FOR THE INDICATED DRIVE.

6. 'RH11 DIDN'T RESPOND TO ADDRESSING' - THE PROGRAM ACCESSED THE RH11 AT THE INDICATED ADDRESS AND RECEIVED NO RESPONSE.

7. 'DRIVE OR DATA ERROR' - THE INDICATED DRIVE COMPLETED AN OPERATION WITH THE INDICATED ERROR CONDITIONS SET.

8. 'DRIVE UNSAFE ERROR' - THE INDICATED DRIVE SIGNALED AN UNSAFE CONDITION.

9. 'HEAD OUT OF ALIGNMENT' - THE INDICATED HEAD OF NOT WITHIN TOLERANCE.

10. 'HEAD TOO FAR OUT OF ALIGNMENT' - THE INDICATED HEAD IS TOO FAR OUT OF ALIGNMENT FOR THE PROGRAM TO FIND THE TRACK CENTERLINE FOR THAT HEAD.

11. 'SOFTWARE TIMEOUT' - THE INDICATED DID NOT COMPLETE THE OPERATION WITH 1 SECOND.

12. 'UNCORRECTABLE MASSBUS PARITY ERROR' - THE PROGRAM ATTEMPTED TO PERFORM AN OPERATION AND DETECTED 3 SUCESSIVE MASSBUS PARITY ERRORS OR THE PROGRAM ATTEMPTED TO CLEAR A 'PAR' ERROR IN THE DRIVE AND A PARITY ERROR OCCURRED.

13. 'DRIVE IS UNAVAILABLE' - THE INDICATED DRIVE HAS GONE OFFLINE OR HAS BECOME NON-EXISITENT.

7. RESTRICTIONS

BECAUSE AN ALIGNMENT DISK PACK IS USED ON DRIVES TESTED BY THIS

PROGRAM, NO DATA TRANSFER OPERATIONS ARE PERFORMED.

8. PROGRAM DESCRIPTION

8.1 VERIFICATION MODE

8.1.1 RP04/5 OPERATIONS

THE PROGRAM CHECKS HEAD ALIGNMENT AT CYLINDER 245, HEADS 0 - 18, AT CYLINDERS 400 AND 4, HEADS 0 AND 18, AND REVERFIES ALIGNMENT AT CYLINDER 245, HEADS 0 - 18. THE OPERATOR WILL BE NOTIFIED IF ANY HEAD IS OUT OF ALIGNMENT BY MORE THAN THE SPECIFIED AMOUNT.

HEAD ALIGNMENT IS CHECKED IN THE FOLLOWING MANNER:

1. OFFSET THE POSITIONER TO +1200 MICRO-INCHES.
2. STORE THE SIGN CHANGE BIT.
3. MOVE THE POSITIONER IN THE OPPOSITE DIRECTION IN 25 MICRO-INCH INCREMENTS UNTIL THE SIGN CHANGE BIT CHANGES VALUE. STORE THE OFFSET VALUE.
4. OFFSET THE POSITIONER TO -1200 MICRO-INCHES AND REPEAT STEPS 2 AND 3 ABOVE.
5. AVERAGE THE TWO SIGN CHANGE OFFSET VALUES AND REPORT IF THE SELECTED HEAD IS MISALIGNED BY MORE THAN + OR - 150 MICRO-INCHES FOR CYLINDER 245 OR + OR - 350 MICRO-INCHES FOR CYLINDERS 4 AND 400.

REPEAT THE ABOVE SEQUENCE FOR ALL HEADS AT CYLINDER 245 AND FOR HEADS 0 AND 18 AT CYLINDERS 4 AND 400.

THE OPERATOR MAY TERMINATE THE OPERATION AT ANY TIME BY TYPING A 'CONTROL C'.

8.1.2 RP06 OPERATIONS

THE PROGRAM CHECKS HEAD ALIGNMENT AT CYLINDER 496, HEADS 0 - 18, AT CYLINDERS 800 AND 8, HEADS 0, 1, 10, 17, AND 18, AND RE-VERFIES ALIGNMENT AT CYLINDER 496, HEADS 0 - 18. THE OPERATOR WILL BE NOTIFIED IF A HEAD IS OUT OF ALIGNMENT BY MORE THAN THE SPECIFIED AMOUNT.

HEAD ALIGNMENT IS CHECKED IN THE FOLLOWING MANNER:

1. OFFSET THE POSITIONER TO +600 MICRO-INCHES.
2. STORE THE SIGN CHANGE BIT.
3. MOVE THE POSITIONER IN THE OPPOSITE DIRECTION IN 25 MICRO-INCH

INCREMENTS UNTIL THE SIGN CHANGE BIT CHANGES VALUE. STORE THE OFFSET VALUE.

4. OFFSET THE POSITIONER TO -600 MICRO-INCHES AND REPEAT STEPS 2 AND 3 ABOVE.

5. AVERAGE THE TWO SIGN CHANGE OFFSET VALUES AND REPORT IF THE SELECTED HEAD IS MISALIGNED BY MORE THAN + OR - 75 MICRO-INCHES FOR CYLINDER 496 OR + OR - 175 MICRO-INCHES FOR CYLINDERS 8 AND 800.

REPEAT THE ABOVE SEQUENCE FOR ALL HEADS AT CYLINDER 496 AND FOR HEADS 0, 1, 10, 17, 18 AT CYLINDERS 8 AND 800.

THE OPERATOR MAY TERMINATE THE OPERATION AT ANY TIME BY TYPING A 'CONTROL C'.

8.2 ALIGNMENT MODE

THE PROGRAM MAY ALSO BE USED TO PROVIDE HEAD SELECTION FOR DDU CONTROLLED HEAD ALIGNMENT. WHEN THIS MODE IS SELECTED, THE PROGRAM WILL PERFORM NO ALIGNMENT CHECKING - ONLY THE REQUESTED HEAD IS SELECTED IN THE DCL. THE ACTUAL ALIGNMENT MUST BE PERFORMED USING THE ALIGNMENT METER ON THE DDU, DEDU, OR PERCH. WHEN USED IN THE ALIGNMENT MODE, THE PROGRAM PROVIDES HEAD SELECTION WHICH IS NOT PRESENT IN THE DDU.

THE OPERATOR MAY TERMINATE THE OPERATION AT AY TIME BY TYPING A 'CONTROL C'.

8.3 RANDOM SEEK UTILITY

THE PROGRAM CONTAINS A UTILITY ROUTINE WHICH PERFORMS 5,000 (10) RANDOM SEEK OPERATIONS ON THE DRIVE BEING CHECKED. THIS UTILITY ALLOWS THE OPERATOR TO EXERCISE THE HEAD ASSEMBLY ON THE DRIVE AFTER HEAD ALIGNMENT HAS BEEN PERFORMED.

THE UTILITY ROUTINE IS NORMALLY USED AFTER HEADS HAVE BEEN ALIGNED: THE RANDOM SEEK UTILITY IS CALLED AND HEAD ALIGNMENT IS RE-VERIFIED AT THE COMPLETION OF THE RANDOM SEEKS. USE OF THIS ROUTINE DOES NOT REQUIRE CYCLING DOWN THE DRIVE AND REPLACING THE ALIGNMENT PACK WITH A SCRATCH PACK FOR THE HEAD SHAKE DOWN.

THE OPERATOR MAY TERMINATE THE OPERATION AT ANY TIME BY TYPING A 'CONTROL C'.

9. PROGRAM LISTING

```
DZRJCA.DOC

        %
        .END

 ERRORS DETECTED:  0

 *,DZRJCA/SOL=DZRJCA.DOC
 RUN-TIME:  1 2 0 SECONDS
 CORE USED:  3K
```

DZRJC.009 TABLE OF CONTENTS

```
      1
```

```
                        .TITLE  MD-11-DZRIC-A, RP04/5/6 HEAD ALIGNMENT PROGRAM
                        ;*COPYRIGHT (C) 1976
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY C. HESS
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C1),MAR 24, 1976.
                        ;*
000001                  $TN=1
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*         9            LOOP ON ERROR
                        ;*         2            CHECK ONLY THE SPECIFIED HEAD
                        ;*         1            LOOP ON THE CURRENT HEAD
                        ;*         0            TYPE ALL TRACK CENTER VALUES
                        .SBTTL  BASIC DEFINITIONS

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

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

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

                        ;*PRIORITY LEVEL DEFINITIONS
```

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

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0

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

```
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

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

                        .SBTTL  RH11 REGISTERS

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

                        ;CONTROL AND STATUS REGISTER 1 (RPCS1)

000100                  IE=     100             ;INTERRUPT ENABLE (BIT #6)
000200                  RDY=    200             ;READY (BIT #7)
000400                  A16=    400             ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
001000                  A17=    1000            ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
002000                  PSEL=   2000            ;PORT SELECT (BIT #10)
020000                  MCPE=   20000           ;MASSBUSS PARITY ERROR (BIT #13)
040000                  TRE=    40000           ;TRANSFER ERROR (BIT #14)
                        ;SC=    100000          ;SPECIAL CONDITION (BIT #15)

                        ;WORD COUNT REGISTER (RPWC)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        ;BUS ADDRESS REGISTER (RPBA)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        ;CONTROL AND STATUS REGISTER 2 (RPCS2)

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)
000020                  PAT=    20              ;MASSBUS PARITY TEST (BIT #4)
```

```
                        CLR=    40              ;CLEAR (BIT #5)
                        IR=     100             ;INPUT READY (BIT #6)
                        OR=     200             ;OUTPUT READY (BIT #7)
                        MPE=    400             ;MASS BUS PARITY ERROR (BIT #8)
                        MXF=    1000            ;MISSED TRANSFER ERROR (BIT #9)
                        PGE=    2000            ;PROGRAM ERROR (BIT #10)
                        NEM=    4000            ;NON EXISTENT MEMORY (BIT #11)
                        NED=    10000           ;NON EXISTENT DRIVE (BIT #12)
                        UPE=    20000           ;UNIBUS PARITY ERROR (BIT #13)
                        WCE=    40000           ;WRITE CHECK ERROR (BIT #14)
                        DLT=    100000          ;DATA LATE (BIT #15)

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


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

                        .SBTTL  RP04/5/6 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 (RPDS1) (#01)

                        ;DF5=   1               DRIVE FORWARD 5"/SEC.  (BIT #0)
000002                  DFF20=  2               ;DRIVE FORWARD 20"/SEC.  (BIT #1)
000004                  DIGB=   4               ;DRIVE TO INNER GUARD BAND (BIT #2)
000010                  GRV=    10              ;GO REVERSE (BIT #3)
000020                  DL64=   20              ;DIFFERENCE LESS THAN 64 (BIT #4)
000040                  DE1=    40              ;DIFFERENCE EQUALS 1 (BIT #5)
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 (RPER1) (#02)

000001                  ILF=    1               ;ILLEGAL FUNCTION (BIT #0)
```

```
                        ;ILLEGAL REGISTER (BIT #1)
                        ;REGISTER MODIFICATION REFUSED (BIT #2)
                        ;PARITY ERROR (BIT #3)
                        ;FORMAT ERROR (BIT #4)
                        ;WRITE CLOCK FAIL (BIT #5)
                        ;ECC HARD ERROR (BIT #6)
                        ;HEADER COMPARE ERROR (BIT #7)
                        ;HEADER CRC ERROR (BIT #8)
                        ;ADDRESS OVERFLOW ERROR (BIT #9)
                        ;INVALID ADDRESS ERROR (BIT #10)
                        ;WRITE LOCK ERROR (BIT #11)
                        ;DRIVE TIMING ERROR (BIT #12)
                        ;OPERATION INCOMPLETE (BIT #13)
                        ;DRIVE UNSAFE (BIT #14)
                        ;DATA CHECK ERROR (BIT 15)

                        ;MAINTAINABILITY REGISTER (RPMR)(#03)

                        ;DIAGINOSTIC MODE (BIT #0)
                        ;MAINTAINABILITY CLOCK (BIT #1)
                        ;MAINTAINABILITY INDEX (BIT #2)
                        ;MAINTAINABILITY SECTOR CLOCK (BIT #3)
                        ;MAINTAINABILITY READ (BIT #4)
                        ;MAINTAINABILITY WRITE (BIT #5)
                        ;MAINTAINABILITY SYNC DETECTED (BIT #7)

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

                        ;DEVICE 0 (BIT #0)
                        ;DEVICE 1 (BIT #1)
                        ;DEVICE 2 (BIT #2)
                        ;DEVICE 3 (BIT #3)
                        ;DEVICE 4 (BIT #4)
                        ;DEVICE 5 (BIT #5)
                        ;DEVICE 6 (BIT #6)
                        ;DEVICE 7 (BIT #7)

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

                        ;DRIVE TYPE REGISTER (RPDT) (#06)

                        ;DRIVE TYPE NUMBER BIT 1
                        ;DRIVE TYPE NUMBER BIT 2
                        ;DRIVE TYPE NUMBER BIT 3
                        ;DRIVE TYPE NUMBER BIT 4
                        ;DRIVE TYPE NUMBER BIT 5
                        ;DRIVE TYPE NUMBER BIT 6
                        ;DRIVE TYPE NUMBER BIT 7
                        ;DRIVE TYPE NUMBER BIT 8
                        ;DRIVE TYPE NUMBER BIT 9
                        ;DRIVE REQUEST REQUIRED (BIT #11)
                        ;MOVING HEAD (BIT #13)
                        ;TAPE DRIVE (BIT #14)
```

```
100000                  NBA=    100000                  ;NOT BLOCK ADDRESSED (BIT #15)

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

000001                  EXT1=   1                       ;EXTENSION 1 (BIT #0)
000002                  EXT2=   2                       ;EXTENSION 2 (BIT #1)
000004                  EXT4=   4                       ;EXTENSION 3 (BIT #2)
000010                  EXT10=  10                      ;EXTENSION 4 (BIT #3)
000020                  EXT20=  20                      ;EXTENSION 5 (BIT #4)
000040                  EXT40=  40                      ;EXTENSION 6 (BIT #5)
000100                  SC1=    100                     ;SECTOR COUNT FIELD 0 (BIT #6)
000200                  SC2=    200                     ;SECTOR COUNT FIELD 1 (BIT #7)
                        ;SC4=   400                     ;SECTOR COUNT FIELD 2 (BIT #8)
001000                  SC10=   1000                    ;SECTOR COUNT FIELD 3 (BIT #9)
002000                  SC20=   2000                    ;SECTOR COUNT FIELD 4 (BIT #10)
004000                  TRK1=   4000                    ;TRACK FIELD 1 (BIT #11)
010000                  TRK2=   10000                   ;TRACK FIELD 2 (BIT #12)
020000                  TRK4=   20000                   ;TRACK FIELD 3 (BIT #13)
040000                  TRK10=  40000                   ;TRACK FIELD 4 (BIT #14)
100000                  TRK20=  100000                  ;TRACK FIELD 5 (BIT #15)

                        ;RP04 ERROR REGISTER #2 (RPER2) (#10)

000001                  WCU=    1                       ;WRITE CURRENT UNSAFE (BIT #0)
000002                  CSF=    2                       ;CURRENT SINK FAILURE (BIT #1)
000004                  WSU=    4                       ;WRITE SELECT UNSAFE (BIT #2)
000010                  CSU=    10                      ;CURRENT SWITCH UNSAFE (BIT #3)
000020                  MSE=    20                      ;MOTOR SEQUENCE ERROR (BIT #4)
000040                  TDF=    40                      ;TRANSITIONS DETECTOR FAILURE (BIT #5)
000100                  TUF=    100                     ;TRANSITIONS UNSAFE (BIT #6)
000200                  FEN=    200                     ;FAILSAFE ENABLED (BIT #7)
000400                  WRU=    400                     ;WRITE READY UNSAFE (BIT #8)
001000                  MHS=    1000                    ;MULTIPLE HEAD SELECT (BIT #9)
002000                  NHS=    2000                    ;NO HEAD SELECTION (BIT #10)
004000                  IXE=    4000                    ;INDEX ERROR (BIT #11)
010000                  VU30=   10000                   ;30VOLT UNSAFE (BIT #12)
020000                  PLU=    20000                   ;PLO UNSAFE (BIT #13)
100000                  ACU=    100000                  ;AC UNSAFE (BIT #15)

                        ;RP05/6 ERROR REGISTER #02 (RPER2) (#10)

000001                  WCU=    1                       ;WRITE CURRENT UNSAFE (BIT #0)
000002                  CSF=    2                       ;CURRENT SINK FAILURE (BIT #1)
000004                  WSU=    4                       ;WRITE SELECT UNSAFE (BIT #2)
000010                  CSU=    10                      ;CURRENT SWITCH UNSAFE (BIT #3)
000020                  RAW=    20                      ;READ AND WRITE (BIT #4)
000040                  TDF=    40                      ;TRANSITIONS DETECTOR FAILURE (BIT #5)
000100                  TUF=    100                     ;TRANSITIONS UNSAFE (BIT #6)
000200                  ABS=    200                     ;ABNORMAL STOP (BIT #7)
000400                  WRU=    400                     ;WRITE READY UNSAFE (BIT #8)
001000                  MHS=    1000                    ;MUTLTIPLE HEAD SELECT (BIT #9)
002000                  NHS=    2000                    ;NO HEAD SELECTION (BIT #10)
004000                  IXE=    4000                    ;INDEX ERROR (BIT #11)
020000                  PLU=    20000                   ;PLO UNSAFE (BIT #12)
```

```
                                ;OFFSET REGISTER (RPOF) (#11)

000001                          OF25=   1               ;OFFSET 25 MICRO INCHES (BIT #0)
000002                          OF50=   2               ;OFFSET 50 MICRO INCHES (BIT #1)
000004                          OF100=  4               ;OFFSET 100 MICRO INCHES (BIT #2)
000010                          OF200=  10              ;OFFSET 200 MICRO INCHES (BIT #3)
000020                          OF400=  20              ;OFFSET 400 MICRO INCHES (BIT #4)
000040                          OF800=  40              ;OFFSET 800 MICRO INCHES (BIT #5)
000200                          OFREV=  200             ;OFFSET NEGATIVE (REVERSE) (BIT #5)
002000                          HCI=    2000            ;HEADER COMPARE INHIBIT (BIT #10)
004000                          ECI=    4000            ;ERROR CORRECTION CODE INHIBIT (BIT #11)
010000                          FMT22=  10000           ;FORMAT BIT (BIT #12

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

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

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

                                ;RP04 ERROR REGISTER #03 (RPER3) (#15)

000001                          PSU=    1               ;PACK SPEED UNSAFE (BIT #0)
000002                          VUF=    2               ;VELOCITY UNSAFE (BIT #1)
000010                          UWR=    10              ;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
000040                          ACL=    40              ;AC LOW (BIT #5)
000100                          DCL=    100             ;DC LOW (BIT #6)
040000                          SKI=    40000           ;SEEK INCOMPLETE (BIT #14)
100000                          OCYL=   100000          ;OFF CYLINDER (BIT #15)

                                ;RP05/6 ERROR REGISTER #03 (RPER3) (#15)

000001                          DCU=    1               ;DC UNSAFE (BIT #0)
000002                          WAO=    2               ;WRITE AND OFFSET (BIT #1)
000040                          ACL=    40              ;AC LOW (BIT #5)
000100                          DCL=    100             ;DC LOW (BIT #6)
020000                          OPE=    20000           ;OPERATOR PLUG ERROR (BIT #13)
040000                          SKI=    40000           ;SEEK INCOMPLETE (BIT #14)
100000                          OCYL=   100000          ;OFF CYLINDER ERROR (BIT #15)

                                ;ECC POSITION REGISTER (RPEC1) (#16)
                                ;(EACH BIT IS CALLED BY BIT NUMBER)

                                ;ECC PATTERN REGISTER (RPEC2) (#17)
                                ;(EACH BIT IS CALLED BY BIT NUMBER)

                                .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  PROGRAM STARTING ADDRESS = 200
                    000200      .=200
000200  000137  001460          JMP     START1      ;START THE PROGRAM, USE THE DEFAULT RH11 ADDRESS
000204  000137  001450          JMP     START       ;START THE PROGRAM, CHANGE THE RH11 ADDRESS
```

```
                                .SBTTL  COMMON TAGS

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

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134  000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001161  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001162  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
                        000015  CR      =       15
                        000012  LF      =       12
001164  000000          DRVSEL: .WORD   0               ;ADDRESS OF SELECTED DRIVE
001166  000000          DRIVE:  .WORD   0               ;DRIVE NUM STORAGE FOR DRIVER ERR CALLS
001170  000000          ATTN:   .WORD   0               ;ATTENTION REGISTER STORAGE FOR ERROR MESSAGES
001172  000000          $HEAD:  .WORD   0               ;HEAD ADDRESS STORED FOR ERROR MESSAGES
001174  000000          TOPLN:  .WORD   0               ;'TYPE THE HEADER' INDICATOR
001176  000000          OFFDIR: .WORD   0               ;NEG OFFSET PASS IND
001200  000000          BITIND: .WORD   0               ;SIGN CHANGE BIT STORED INDICATOR
001202  000000          SINCNG: .WORD   0               ;SIGN CHANGE BIT STORAGE
001204  000000          PLUS:   .WORD   0               ;STORE THE POS SIGN CHANGE CODE
001206  000000          INREG:  .WORD   0               ;CONTAINS THE VALUE READ FROM THE REGISTER
001210  000000          TOLER:  .WORD   0               ;ALIGNMENT TOLERANCE FOR PRESENT CYLINDER
```

```
                                MAXCYL: .WORD   0               ;CONTAINS THE MAXIMUM CYLINDER ADDRESS
                                                                ;OF DRIVE UNDER TEST
                                MAXPOS: .WORD   0               ;CONTAINS MAXIMUM POSITIVE OFFSET
                                MAXNEG: .WORD   0               ;CONTAINS MAXIMUM NEGATIVE OFFSET
                                MAXOFF: .WORD   0               ;CONTAINS NUMBER OF OFFSET POSITIONS
                                OUTTOL: .WORD   0               ;CONTAINS TOLERANCE FOR INNER AND OUTER CYLINDERS
                                MIDTOL: .WORD   0               ;CONTAINS TOLERANCE FOR THE ALIGNMENT CYLINDER
                                INCYL:  .WORD   0               ;INNER CYLINDER ADDRESS
                                MIDCYL: .WORD   0               ;ALIGNMENT CYLINDER ADDRESS
                                OUTCYL: .WORD   0               ;OUTER CYLINDER ADDRESS
                                TABLE1: .WORD   0               ;CONTAINS HEAD TABLE ADDRESS FOR 'INCYL' & 'OUTCYL'
                                TABLE2: .WORD   0               ;CONTAINS ADDRESS OF OFFSET CODE TABLE
                                SEKCNT: .WORD   0               ;CONTAINS SEEK COUNT
                                CHGADR: .WORD   0               ;CHANGE RH11 BUS ADDRESS FLAG
                                PKV:    .WORD   104,106         ;KW11-P VECTOR ADDRESS
                                PKCS:   .WORD   172540          ;KW11-P CONTROL AND STATUS REG.
                                PKB:    .WORD   172542          ;KW11-P COUNT SET BUFFER
                                PKC:    .WORD   172544          ;KW11-P COUNTER
                                LKV:    .WORD   100,102         ;KW11-L VECTOR ADDRESS
                                LKS:    .WORD   177546          ;KW11-L STATUS REGISTER

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

                                .SBTTL  RH11/RPO4/5/6 ADDRESS & VECTOR LOCATIONS

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

                                $RPADR: .WORD   176700          ;RH11/RPO4/5/6 UNIBUS ADDRESS
                                $RPVEC: .WORD   254             ;RH11/RPO4/5/6 VECTOR ADDRESS
```

```
                                .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

001270                          $ERRTB:

                                ;ERROR 1

001270  020516                          EM1             ;RH11 INTERRUPT OCCURRED (RHAS=0)
001272  021247                          DH1
001274  021636                          DT1
001276  021750                          DF1

                                ;ERROR 2

001300  020557                          EM2             ;UNEXPECTED ATTENTION OCCURRED
001302  021254                          DH2
001304  021642                          DT2
001306  021751                          DF2

                                ;ERROR 3

001310  020615                          EM3             ;MASSBUS PARITY ERROR (MCPE=1)
001312  021331                          DH3
001314  021660                          DT3
001316  021757                          DF3

                                ;ERROR 4

001320  020653                          EM4             ;MASSBUS PARITY ERROR (PAR=1)
001322  021357                          DH4
001324  021670                          DT4
001326  021762                          DF4

                                ;ERROR 5

001330  020710                          EM5             ;ADDRESS PLUG CHANGE BIT SET
001332  021254                          DH2
001334  021642                          DT2
001336  021751                          DF2

                                ;ERROR 6
```

```
001340  020744                          EM6             ;RH11 DIDN'T RESPOND TO ADDRESSING
001342  021416                          DH6
001344  021702                          DT6
001346  021750                          DF1

                                ;ERROR 7

001350  000000                          0               ;UNUSED
001352  000000                          0
001354  000000                          0
001356  000000                          0

                                ;ERROR 10

001360  021006                          EM10            ;DRIVE OR DATA ERROR
001362  021431                          DH10
001364  021706                          DT10
001366  021766                          DF10

                                ;ERROR 11

001370  021032                          EM11            ;DRIVE UNSAFE ERROR
001372  021431                          DH10
001374  021706                          DT10
001376  021766                          DF10

                                ;ERROR 12

001400  021055                          EM12            ;HEAD OUT OF ALIGNMENT
001402  021517                          DH12
001404  021726                          DT12
001406  021775                          DF12

                                ;ERROR 13

001410  021103                          EM13            ;HEAD TOO FAR OUT OF ALIGNMENT
001412  021610                          DH13
001414  021736                          DT13
001416  022000                          DF13

                                ;ERROR 14

001420  021141                          EM14            ;SOFTWARE TIMEOUT
001422  021431                          DH10
001424  021706                          DT10
001426  021766                          DF10

                                ;ERROR 15

001430  021162                          EM15            ;UNCORRECTABLE MASSBUS PARITY ERROR
001432  000000                          0
001434  000000                          0
001436  000000                          0
```

```
                                ;ERROR 16

001440  021225                          EM16            ;DRIVE IS UNAVAILABLE
001442  021627                          DH16
001444  021744                          DT16
001446  021750                          DF1

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

                                .SBTTL  RP04/5/6 DRIVER COMMANDS

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

                        000101  RNOP=    101             ;NO OPERATION
                        000103  UNLOAD=  103             ;UNLOAD
                        000105  SEEK=    105             ;SEEK
                        000107  RECAL=   107             ;RECALIBRATE
                        000111  DRVCLR=  111             ;DRIVE CLEAR
                        000113  REL=     113             ;RELEASE
                        000115  OFFSET=  115             ;OFFSET
                        000117  RTC=     117             ;RETURN TO CENTER LINE
                        000121  PRESET=  121             ;READ IN PRESET
                        000123  ACK=     123             ;PACK ACKNOWLEDGE
                        000131  SEARCH=  131             ;SEARCH
                        000141  GETREG=  141             ;GET REGISTERS
                        000143  SETFMT=  143             ;SET FORMAT (& ECI OR HCI)
                        000145  SELDRV=  145             ;SELECT DRIVE
                        000151  WCHKD=   151             ;WRITE CHECK DATA
                        000153  WCHKHD=  153             ;WRITE CHECK HEADER & DATA
                        000161  WRTDAT=  161             ;WRITE DATA
                        000163  WRTHD=   163             ;WRITE HEADER & DATA
                        000171  RDDAT=   171             ;READ DATA
                        000173  RDHD=    173             ;READ HEADER & DATA


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

                                .SBTTL  PROGRAM START AND INITIALIZATION ROUTINES

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

001450  012737  177777  001242  START:  MOV     #-1,CHGADR              ;SET CHANGE RH11 ADDRESS FLAG
001456  000402                          BR      START2          ;FINISH
001460  005037  001242          START1: CLR     CHGADR          ;CLEAR RH11 CHANGE ADDRESS FLAG
001464  000005                  START2: RESET                   ;CLEAR THE BUS
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001466  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001472  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001474  022706  001140                  CMP     #SWR,R6 ;;DONE?
001500  001374                          BNE     .-6             ;;LOOP BACK IF NO
001502  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
001506  012737  004412  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
```

```
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001514  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001522  012737  007452  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001530  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
001536  012737  176543  007132          MOV     #176543,$HINUM  ;;PRIME THE RANDOM NUMBER GENERATOR
001544  012737  123456  007134          MOV     #123456,$LONUM  ;;BOTH HIGH AND LOW WORDS
001552  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001556  012737  001612  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001564  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
001572  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001600  022777  177777  177332          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001606  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
001610  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
001612  012716  001620          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001616  000002                          RTI
001620  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001626  012737  000174  001142          MOV     #DISPREG,DISPLAY
001634  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

001640  005227  177777                  INC     #-1             ;FIRST START ?
001644  001006                          BNE     1$              ;BR IF NOT
001646  104401  016546                  TYPE    ,TITLE          ;TYPE PROGRAM TITLE
001652  104401  016654                  TYPE    ,DDU            ;TYPE 'DDU' SETUP MESSAGE
001656  104401  001161                  TYPE,   $CRLF           ;CR-LF
001662  004737  005604          1$:     JSR     PC,$TKINT       ;TURN ON THE TTY KEYBOARD INTERRUPT
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
001666  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
001672  001006                          BNE     67$             ;;BRANCH IF YES
001674  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
001702  001005                          BNE     68$             ;;BRANCH IF NO
001704  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
001706  000403                          BR      68$
001710  112737  000001  001134  67$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
001716                          68$:
001716  005227  177777                  INC     #-1             ;FIRST START ?
001722  001010                          BNE     INIT            ;BR IF NOT
001724  004737  022002                  JSR     PC,BUSADR       ;CHECK THE RH11 ADDRESS
001730  013737  001264  007704          MOV     $RPADR,RPADR    ;MOVE RH11 ADDRESS TO DRIVER
001736  013737  001266  007706          MOV     $RPVEC,RPVEC    ;VECTOR ADDRRESS


                                ;GET THE DRIVE NUMBER FROM THE OPERATOR AND SETUP THE VARIABLE LOCATION
                                ;FOR RP04/5 OR RP06'S

001744  012716  001100          INIT:   MOV     #STACK,(SP)     ;SETUP THE STACK ADDRESS
001750  005037  177776                  CLR     PS              ;SET PROCESSOR PRIORITY TO ZERO
001754  005037  001174                  CLR     TOPLN           ;CLEAR HEADER INDICATOR
001760  004737  005604                  JSR     PC,$TKINT       ;INITIALIZE THE TTY KEYBOARD
001764  104401  017555                  TYPE    ,ENTERD         ;ASK FOR DRIVE ASSIGNMENT
001770  004737  004270                  JSR     PC,GETNUM       ;GET DRIVE NUMBER
001774  000410                          BR      1$              ;'CR' ENTERED
001776  011637  001164                  MOV     (SP),DRVSEL     ;DRIVE NUMBER
```

```
                                                MOVB    (SP)+,DPB       ;DRIVE NUMBER TO PAR BLOCK
                                                CMP     DRVSEL,#8.      ;CHECK DRIVE NUMBER
                                                BLO     2$              ;BR IF DRIVE NUMBER OK
                                        1$:     TYPE    ,$QUES          ;TYPE QUESTION MARK
                                                BR      INIT            ;TRY AGAIN
                                        2$:     JSR     PC,ST.CLK       ;INITIALIZE THE CLOCK
                                                JSR     PC,RPINIT       ;INITIALIZE THE DRIVER
                                                MOV     #-1,SEEKFG      ;SET SEEK ALLOW FLAG
                                                MOV     #-1,SAVEFG      ;SAVE DRIVE REGISTERS
                                                MOVB    DPB,R0          ;USE DRIVE NUMBER AS AN INDEX
                                                TSTB    DRVSTA(R0)      ;SEE IF DRIVE AVAILABLE
                                                BGT     3$              ;BRANCH IF IT IS
                                                TYPE    ,NODRV          ;'DRIVE NOT AVAILABLE'
                                                JMP     INIT            ;TRY IT AGAIN
                                        3$:     BITB    #BIT00!BIT01,DRVTYP(R0) ;RP04/5 ?
                                                BEQ     4$              ;BR IF NOT
                                                MOV     #350.,OUTTOL    ;SETUP 4 & 400 TOLERANCES
                                                MOV     #75.,MIDTOL     ;TOLERANCE AT CYLINDER 245
                                                MOV     #400.,INCYL     ;INNER ALIGNMENT CYLINDER
                                                MOV     #4,OUTCYL       ;OUTER ALIGNMENT CYLINDER
                                                MOV     #245.,MIDCYL    ;ALIGNMENT CYLINDER
                                                MOV     #HEAD45,TABLE1  ;HEAD ADDRESS TABLE
                                                MOV     #OFTBL4,TABLE2  ;OFFSET TABLE ADDRESS
                                                MOV     #411.,MAXCYL    ;LAST CYLINDER + 1
                                                MOV     #-1200.,MAXNEG  ;MAXIMUM NEGATIVE OFFSET VALUE
                                                MOV     #1200.,MAXPOS   ;MAXIMUM POSITIVE OFFSET VALUE
                                                MOV     #97.,MAXOFF     ;NUMBER OF OFFSET POSITIONS
                                                BR      5$              ;CONTINUE WITH THE SETUP
                                        4$:     MOV     #175.,OUTTOL    ;TOLERANCE AT CYLINDERS 8 & 800
                                                MOV     #75.,MIDTOL     ;TOLERANCE AT ALIGNMENT CYLINDER
                                                MOV     #800.,INCYL     ;INNER ALIGNMENT CYLINDER
                                                MOV     #8.,OUTCYL      ;OUTER ALIGNMENT CYLINDER
                                                MOV     #496.,MIDCYL    ;ALIGNMENT CYLINDER
                                                MOV     #HEAD6,TABLE1   ;HEAD ADDRESS TABLE
                                                MOV     #OFTBL6,TABLE2  ;OFFSET CODE TABLE
                                                MOV     #815.,MAXCYL    ;MAXIMUM CYLINDER ADDRESS + 1
                                                MOV     #-600.,MAXNEG   ;MAXIMUM NEGATIVE OFFSET VALUE
                                                MOV     #600.,MAXPOS    ;MAXIMUM POSITIVE OFFSET VALUE
                                                MOV     #49.,MAXOFF     ;NUMBER OF OFFSET POSITIONS
                                        5$:     JSR     R5,DRIVER       ;RECALIBRATE
                                                .WORD   RECAL           ;DRIVER CALL
                                                BIT     #BIT11,REG+RPDS1  ;SEE IF WRITE LOCK SET
                                                BNE     6$              ;BR IF WRITE LOCKED
                                                TYPE    ,WLOCK          ;REPORT NOT WRITE LOCKED
                                                BR      INIT            ;TRY IT AGAIN
                                        6$:     TYPE    ,MODE           ;SEE IF ALIGN, VERIFY, OR EXERCISE
                                                RDCHR                   ;GET THE ENTRY
                                                MOV     (SP)+,10$       ;SAVE THE ENTRY
                                                CMPB    #'V,10$         ;WAS A 'V' ENTERED ?
                                                BEQ     9$              ;BR IF IT WAS
                                                CMP     #'A,10$         ;WAS AN 'A' ENTERED ?
                                                BNE     7$              ;BR IF NOT
                                                JMP     DDUALN          ;HEADS TO BE ALIGNED WITH DDU
                                        7$:     CMP     #'E,10$         ;SEE IF EXERCISE
```

```
744  002400  001004                        BNE     8$              ;BR IF 'E' NOT ENTERED
745  002402  000137  003640                JMP     RANDOM          ;GO TO EXERCISE ROUTINE
746  002406  104401  002442                TYPE    .10$            ;ECHO THE INVALID CHARACTER
747  002412  104401  001160        8$:     TYPE    #$QUES          ;TYPE A QUESTION MARK
748  002416  000137  001744                JMP     INIT            ;GO ALL THE WAY BACK TO DRIVE ENTRY
749  002422  104401  020025        9$:     TYPE    .VERIFY         ;TYPE 'VERIFY'
750  002426  032777  000004  176504        BIT     #SW2,@SWR       ;SWITCH 2 SET ?
751  002434  001504                        BEQ     CYLDER          ;BR IF NOT SET - CHECK ALL HEADS
752  002436  000137  002444                JMP     ONEHD           ;CHECK ONLY THE SPECIFIED HEAD
753  002442  000000        10$:    .WORD   0               ;OPERATOR ENTRY GOES HERE
754
755                          ;;*****************************************************************
756
757                          .SBTTL  SETUP TO CHECK ONLY THE SPECIFIED HEAD
758
759                          ;;*****************************************************************
760
761  002444  104401  020365        ONEHD:  TYPE    #ENTCYL         ;ASK FOR CYLINDER ADDRESS
762  002450  004737  004270                JSR     PC,GETNUM       ;GET THE CYLINDER ADDRESS
763  002454  000427                        BR      1$              ;'CR' ENTERED
764  002456  012637  016132                MOV     (SP)+,DPB+CYL   ;MOVE IT TO PARAMETER BLOCK
765  002462  013737  001222  001210        MOV     OUTTOL,TOLER    ;SET INITIAL ALIGNMENT TOLERANCE
766  002470  023737  001232  016132        CMP     OUTCYL,DPB+CYL  ;SEE IF OUTER CYLINDER
767  002476  001432                        BEQ     4$              ;BR IF IT IS
768  002500  023737  001226  016132        CMP     INCYL,DPB+CYL   ;SEE IF INNER CYLINDER
769  002506  001426                        BEQ     4$              ;BR IF IT IS
770  002510  013737  001224  001210        MOV     MIDTOL,TOLER    ;CHANGE TOLERANCE
771  002516  023737  001230  016132        CMP     MIDCYL,DPB+CYL  ;SEE IF ALIGNMENT CYLINDER
772  002524  001414                        BEQ     3$              ;BR IF YES
773  002526  104401  020447                TYPE    #BADCYL         ;REPORT INVALID CYLINDER
774  002532  000744                        BR      ONEHD           ;TRY IT AGAIN
775  002534  023727  001212  000633  1$:    CMP     MAXCYL,#411.    ;RP04/5 ?
776  002542  001403                        BEQ     2$              ;BR IF YES
777  002544  104401  020166                TYPE    #CYL496         ;CYLINDER 496 DEFAULT MESSAGE
778  002550  000402                        BR      3$              ;CONTINUE
779  002552  104401  020122        2$:     TYPE    #CYL245         ;CYLINDER 245 DEFAULT MESSAGE
780  002556  013737  001230  016132  3$:    MOV     MIDCYL,DPB+CYL  ;CYLINDER ADDRESS
781  002564  104401  020346        4$:     TYPE    #ENTHD          ;ASK FOR THE HEAD ADDRESS
782  002570  004737  004270                JSR     PC,GETNUM       ;GET THE HEAD ADDRESS
783  002574  000411                        BR      5$              ;'CR' ENTERED
784  002576  012637  001172                MOV     (SP)+,$HEAD     ;SAVE THE HEAD ADDRESS
785  002602  022737  000022  001172        CMP     #18.,$HEAD      ;ADDRESS VALID ?
786  002610  002007                        BGE     6$              ;BR IF YES
787  002612  104401  020420                TYPE    #BADTRK         ;INVALID HEAD ADDRESS MESSAGE
788  002616  000762                        BR      4$              ;TRY AGAIN
789  002620  104401  020064        5$:     TYPE    #HDZERO         ;HEAD DEFAULT MESSAGE
790  002624  005037  001172                CLR     $HEAD           ;CLEAR HEAD STORAGE
791  002630  004537  003736        6$:     JSR     R5,DRIVER       ;SEEK TO THE SPECIFIED CYLINDER
792  002634  000105                        .WORD   SEEK            ;SEEK CODE
793  002636  004737  003032                JSR     PC,CHECK        ;CHECK THE HEAD
794  002642  000137  001744                JMP     INIT            ;GET THE NEXT DRIVE/HEAD
795
796                          ;;*****************************************************************
797
```

```
                                .SBTTL  MAIN ROUTINE - CHECK ALL HEADS AT ALL ALIGHMENT CYLNDERS
                                ;;*****************************************************************
002646  013737  001230  016132  CYLDER: MOV     MIDCYL,DPB+CYL  ;START AT THE ALIGNMENT CYLINDER
002654  013737  001224  001210          MOV     MIDTOL,TOLER    ;ALIGNMENT CHECK TOLERANCE
002662  004537  003736                  JSR     R5,DRIVER       ;SEEK TO THE ALIGNMENT CYLINER
002666  000105                          .WORD   SEEK            ;DRIVER SEEK CODE
002670  012704  016210                  MOV     #HEAD1,R4       ;ADDRESS OF ALIGNMENT HEAD SELECTION TABLE
002674  004737  003014                  JSR     PC,1$           ;CHECK ALIGNMENT, ALL HEADS, AT THE ALIGNMENT CYLINDER
002700  013737  001226  016132          MOV     INCYL,DPB+CYL   ;CHANGE TO INNER CYLINDER
002706  013737  001222  001210          MOV     OUTTOL,TOLER    ;CHANGE TOLERANCE VALUE
002714  004537  003736                  JSR     R5,DRIVER       ;SEEK TO INNER CYLINDER
002720  000105                          .WORD   SEEK            ;DRIVER SEEK CODE
002722  013704  001234                  MOV     TABLE1,R4       ;CHANGE HEAD TABLE ADDRESS
002726  004737  003014                  JSR     PC,1$           ;CHECK ALIGNMENT AT INNER CYLINDER
002732  013737  001232  016132          MOV     OUTCYL,DPB+CYL  ;CHANGE CYLINDER TO OUTER CYLINDER
002740  004537  003736                  JSR     R5,DRIVER       ;SEEK TO OUTER CYLINDER
002744  000105                          .WORD   SEEK            ;DRIVER SEEK CODE
002746  004737  003014                  JSR     PC,1$           ;CHECK ALIGNMENT AT OUTER CYLINDER
002752  013737  001230  016132          MOV     MIDCYL,DPB+CYL  ;GO BACK TO THE ALIGNMENT CYLINDER
002760  013737  001224  001210          MOV     MIDTOL,TOLER    ;CHANGE ALIGNMENT TOLERANCE
002766  004537  003736                  JSR     R5,DRIVER       ;SEEK
002772  000105                          .WORD   SEEK            ;DRIVER SEEK CODE
002774  012704  016210                  MOV     #HEAD1,R4       ;ADDRESS OF 0 - 19 HEAD ADDRESS TABLE
003000  004737  003014                  JSR     PC,1$           ;RECHECK ALIGNMENT AT ALIGNMENT CYLINDER
003004  104401  020502                  TYPE    ,ENDMSG         ;TYPE 'DONE'
003010  000137  001744                  JMP     $INIT           ;GET NEXT DRIVE OR HEAD
003014  012437  001172          1$:     MOV     (R4)+,$HEAD     ;MOVE HEAD ADDRESS FROM TABLE
003020  100403                          BMI     2$              ;BR IF END OF TABLE
003022  004737  003032                  JSR     PC,CHECK        ;CHECK THE ALIGNMENT
003026  000772                          BR      1$              ;GET NEXT HEAD VALUE
003030  000207                  2$:     RTS     PC              ;RETURN

                                ;;*****************************************************************
                                .SBTTL  ROUTINE TO FIND THE TRACK CENTER
                                ;;*****************************************************************
003032  005037  001176  CHECK:  CLR     OFFDIR          ;CLEAR THE OFFSET DIRECTION INDICATOR
003036  005037  001200          CLR     BITIND          ;CLEAR THE SIGN BIT INDICATOR
003042  013702  001214          MOV     MAXPOS,R2       ;MAXIMUM POSITIVE OFFSET VALUE
003046  013703  001220          MOV     MAXOFF,R3       ;MAXIMUM NUMBER OF OFFSETS
003052  013701  001236          MOV     TABLE2,R1       ;OFFSET CODE TABLE ADDRESS
003056  013746  001172          MOV     $HEAD,-(SP)     ;HEAD VALUE TO STACK
003062  000316                  SWAB    (SP)            ;SWITCH FOR RPDA LOAD
003064  004037  015012          JSR     R0,WRT.RP       ;LOAD RPDA WITH HEAD ADDRESS
003070  000006                  RPDA                    ;REGISTER ADDRESS
003072  001744                  INIT                    ;UNCORRECTABLE PARITY ERROR RETURN
003074  112137  016121  1$:     MOVB    (R1)+,DPB+CODE  ;OFFSET VALUE TO PARAMETER BLOCK
003100  162702  000001          SUB     #25.,R2         ;DECREMENT OFFSET VALUE
003104  003404                  BR      3$              ;BYPASS REVERSE DIRECTION SETUP
003106  114137  016121  2$:     MOVB    -(R1),DPB+CODE  ;REVERSE OFFSET VALUE TO PAR BLOCK
```

```
                003112  062702  000001          ADD     #25.,R2         ;INCREMENT OFFSET VALUE
                003116  004537  003736    3$:   JSR     R5,DRIVER       ;OFFSET THE DRIVE
                003122  000115              .WORD   OFFSET          ;OFFSET OP CODE
                003124  005737  001200          TST     BITIND          ;INITIAL SIGN VALUE STORED ?
                003130  001011              BNE     4$              ;BR IF STORED
                003132  005137  001200          COM     BITIND          ;SET THE SIGN BIT INDICATOR
                003136  042737  077777  016172  BIC     #^CBIT15,REG+RPOF  ;LEAVE THE SIGN BIT
                003144  013737  016172  001202  MOV     REG+RPOF,SINCNG ;STORE THE SIGN CHANGE BIT
                003152  000407              BR      5$
                003154  042737  077777  016172  4$:   BIC     #^CBIT15,REG+RPOF  ;KEEP ONLY THE SIGN VALUE
                003162  023737  016172  001202      CMP     REG+RPOF,SINCNG ;SIGN CHANGED ?
                003170  001044              BNE     8$              ;BRANCH IF IT DID
                003172  005303              5$:   DEC     R3              ;DECREMENT THE OFFSET ATTEMPT COUNTER
                003174  001036              BNE     7$              ;BRANCH IF OFFSETS NOT COMPLETED
                003176  032777  000001  175734  BIT     #SWD,@SWR       ;CHECK SWR0
                003204  001002              BNE     6$              ;BR IF SET
                003206  104013              ERROR   13              ;REPORT OUT OF LIMIT CONDITION
                003210  000534              BR      13$             ;SEE IF LOOP ON ERROR
                003212  005737  001174    6$:   TST     TOPLN           ;TYPE THE HEADING LINE ?
                003216  001005              BNE     17$             ;BR IF NOT
                003220  104401  017667          TYPE    .HEADER         ;TYPE THE HEADER
                003224  012737  177777  001174  MOV     #-1,TOPLN       ;SET THE BYPASS SWITCH
                003232                    17$:
                003232  013746  001172          MOV     $HEAD,-(SP)     ;;SAVE $HEAD FOR TYPEOUT
                003236  104405              TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                003240  013746  016132          MOV     DPB+CYL,-(SP)   ;;SAVE DPB+CYL FOR TYPEOUT
                003244  104405              TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                003246  104401  017643          TYPE    .TOLRG          ;HEAD OUT OF LIMIT
                003252  104401  001161          TYPE    .$CRLF          ;CR-LF
                003256  032777  100000  175654  BIT     #SW15,@SWR      ;HALT IF ERROR ?
                003264  001506              BEQ     13$             ;BR IF NOT
                003266  000000              HALT                    ;HEAD JUST TYPED OUT OF TOLERANCE
                003270  000504              BR      13$             ;CHECK FOR LOOP
                003272  005737  001176    7$:   TST     OFFDIR          ;SEE WHICH DIRECTION
                003276  001676              BEQ     1$              ;BRANCH IF FORWARD
                003300  000702              BR      2$              ;REVERSE DIRECTION
                003302  005737  001176    8$:   TST     OFFDIR          ;FINISHED WITH REVERSE ?
                003306  001017              BNE     9$              ;BRANCH IF YES
                003310  010237  001204          MOV     R2,PLUS         ;SAVE THE SIGN CHANGE VALUE
                003314  013702  001216          MOV     MAXNEG,R2       ;MAXIMUM NEGATIVE OFFSET VALUE
                003320  013703  001220          MOV     MAXOFF,R3       ;NUMBER OF OFFSETS
                003324  013701  001220          MOV     MAXOFF,R1       ;NUMBER OF OFFSETS
                003330  063701  001236          ADD     TABLE2,R1       ;BEGINNING OF REVERSE DIRECTION VALUES
                003334  005137  001176          COM     OFFDIR          ;SET REVERSE DIRECTION INDICATOR
                003340  005037  001200          CLR     BITIND          ;RESET SIGN BIT INDICATOR
                003344  000660              BR      2$              ;START REVERSE OFFSETS
                003346  063702  001204    9$:   ADD     PLUS,R2         ;ADD THE TWO SIGN CHANGE VALUES
                003352  006202              ASR     R2              ;'DIVIDE' BY 2
                003354  010237  001204          MOV     R2,PLUS         ;SAVE THE RESULT
                003360  005702              TST     R2              ;SEE IF VALUE NEGATIVE
                003362  100001              BPL     10$             ;BRANCH IF POSITIVE
                003364  005402              NEG     R2              ;RECOMPLEMENT THE VALUE
                003366  032777  000001  175644  10$:   BIT     #SW0,@SWR       ;SEE IF SWR0 SET
                003374  001005              BNE     11$             ;BRANCH IF IT IS
```

```
003376  023702  001210                    CMP     TOLER,R2        ;SEE IF TRACK CENTERLINE WITHIN  TOLERANCE
003402  002043                            BGE     14$             ;BRANCH IF VALUE OK
003404  104012                            ERROR   12              ;REPORT THE ERROR
003406  000435                            BR      13$             ;CHECK FOR LOOP ON ERROR
003410  010246                    11$:    MOV     R2,-(SP)        ;SAVE TRACK CENTER VALUE
003412  005737  001174                    TST     TOPLN           ;TYPE HEADER ?
003416  001005                            BNE     12$             ;BR IF NOT
003420  104401  017667                    TYPE    ,HEADER         ;TYPE THE HEADER
003424  012737  177777  001174            MOV     #-1,TOPLN       ;CLEAR THE INDICATOR
003432                            12$:
003432  013746  001172                    MOV     $HEAD,-(SP)     ;;SAVE $HEAD FOR TYPEOUT
003436  104405                            TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
003440  013746  016132                    MOV     DPB+CYL,-(SP)   ;;SAVE DPB+CYL FOR TYPEOUT
003444  104405                            TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
003446  013746  001204                    MOV     PLUS,-(SP)      ;;SAVE PLUS FOR TYPEOUT
003452  104405                            TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
003454  104401  001161                    TYPE,   $CRLF           ;CR-LF
003460  023726  001210                    CMP     TOLER,(SP)+     ;SEE IF HEAD IN TOLERANCE
003464  002012                            BGE     14$             ;BR IF IN TOLERANCE
003466  032777  100000  175444            BIT     #SW15,@SWR      ;HALT IF ERROR ?
003474  001406                            BEQ     14$             ;BR IF NOT
003476  000000                            HALT                    ;HEAD JUST TYPED OUT OF TOLERANCE.
003500  000404                            BR      14$             ;CHECK FOR SWITCH 1
003502  032777  001000  175430    13$:    BIT     #SW9,@SWR       ;SEE IF SWITCH 9 SET
003510  001004                            BNE     15$             ;BR IF SET
003512  032777  000002  175420    14$:    BIT     #SW1,@SWR       ;SWITCH 1 SET ?
003520  001402                            BEQ     16$             ;NOT SET
003522  000137  003032            15$:    JMP     CHECK           ;DO THE HEAD AGAIN
003526  000207                    16$:    RTS     PC              ;RETURN

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

                                .SBTTL  ROUTINE TO SELECT HEAD FOR DDU CONTROLLED HEAD ALIGNMENT

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

003530  104401  020037            DDUALN: TYPE    ,ALIGN          ;TYPE 'ALIGN'
003534  022737  000633  001212            CMP     #411.,MAXCYL    ;RP04/5 ?
003542  001403                            BEQ     1$              ;BR IF YES
003544  104401  020300                    TYPE    ,ALN496         ;'POSITIONED AT 496' MESSAGE
003550  000402                            BR      2$              ;CONTINUE
003552  104401  020232            1$:     TYPE    ,ALN245         ;'POSITIONED AT 245'
003556  013737  001230  016132    2$:     MOV     MIDCYL,DPB+CYL  ;LOAD CYLINDER ADDRESS
003564  004537  003736                    JSR     R5,DRIVER       ;SEEK TO ALIGNMENT CYLINDER
003570  000105                            .WORD   SEEK            ;DRIVER CODE FOR SEEK
003572  104401  020346            3$:     TYPE    ,ENTHD          ;ASK FOR HEAD NUMBER
003576  004737  004270                    JSR     PC,GETNUM       ;READ THE KEYBOARD
003602  000403                            BR      4$              ;'CR' ENTERED
003604  021627  000022                    CMP     (SP),#18.       ;ENTRY CAN'T BE GREATER THAN 18
003610  101403                            BLOS    5$              ;BR IF LESS OR EQUAL
003612  104401  001160            4$:     TYPE    ,$QUES          ;TYPE A QUESTION MARK
003616  000755                            BR      1$              ;TRY AGAIN
003620  011637  001172            5$:     MOV     (SP),$HEAD      ;HEAD NUMBER FOR TYPEOUT
003624  000316                            SWAB    (SP)            ;PUT HEAD NUMBER INTO UPPER BYTE
```

```
                003626  004037  015012                  JSR     R0,WRT.RP       ;LOAD RPDA
                003632  000006                          RPDA                    ;REGISTER ADDRESS
                003634  001744                          INIT                    ;UNCORRECTABLE PARITY ERROR RETURN
                003636  000755                          BR      3$              ;WAIT FOR NEXT HEAD ENTRY

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

                                        .SBTTL  ROUTINE TO PERFORM 5,000 RANDOM SEEKS

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

                003640  104401  020050  RANDOM: TYPE    ,EXER           ;TYPE 'EXERCISE'
                003644  012737  011610  001240        MOV     #5000.,SEKCNT   ;NUMBER OF SEEKS
                003652  004737  007034  1$:     JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
                003656  013746  007132          MOV     $HINUM,-(SP)    ;PUT UPPER RANDOM NUMBER ON THE STACK
                003662  005046                  CLR     -(SP)           ;UPPER DIVIDEND
                003664  013746  001212          MOV     MAXCYL,-(SP)    ;PUT DIVISOR ON STACK
                003670  004737  007136          JSR     PC,$DIV         ;GET THE REMAINDER (RANDOM CYLINDER ADDRESS)
                003674  021637  016132          CMP     (SP),DPB+CYL    ;SEE IF SAME RANDOM CYLINDER AS LAST
                003700  001003                  BNE     2$              ;BR IF DIFFERENT CYLINDER
                003702  062706  000004          ADD     #4,SP           ;ADJUST THE STACK POINTER
                003706  000761                  BR      1$              ;TRY AGAIN
                003710  012637  016132  2$:     MOV     (SP)+,DPB+CYL   ;PUT CYL ADDR IN THE DPB
                003714  005726                  TST     (SP)+           ;CORRECT THE STACK POINTER
                003716  004537  003736          JSR     R5,DRIVER       ;DO THE SEEK
                003722  000105                  .WORD   SEEK            ;SEEK CODE
                003724  005337  001240          DEC     SEKCNT          ;DECREMENT THE SEEK COUNT
                003730  001350                  BNE     1$              ;BR IF NOT DONE
                003732  000137  001744          JMP     INIT            ;RETURN TO COMMAND ROUTINE

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

                                        .SBTTL  COMMON ENTRY TO THE RP04/5/6 DRIVER

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

                003736  012537  016122  DRIVER: MOV     (R5)+,DPB+COMND ;COMMAND CODE
                003742  004037  010472  1$:     JSR     R0,RP04         ;DRIVER ENTRY
                003746  016120                  DPB                     ;DATA PARAMETER BLOCK ADDRESS
                003750  000774                  BR      1$              ;REQUEST NOT ACCEPTED
                003752  005737  016136  2$:     TST     DPB+STATUS      ;SEE IF ORDER COMPLETE
                003756  001775                  BEQ     2$              ;BR IF NOT COMPLETE
                003760  100401                  BMI     4$              ;BRANCH IF ERROR
                003762  000205          3$:     RTS     R5              ;RETURN
                003764  032737  000200  016136  4$:     BIT     #BIT7,DPB+STATUS  ;DID THE ORDER TERMINATE ?
                003772  001020                  BNE     5$              ;BRANCH IF IT DID
                003774  032737  060006  016136          BIT     #BIT14!BIT13!BIT02!BIT01,DPB+STATUS  ;DRIVE OFFLINE OR
                                                                        ;NON-EXISTENT ?
                004002  001024                  BNE     6$              ;BR IF IT IS
                004004  032737  010000  016136          BIT     #BIT12,DPB+STATUS  ;DRIVE UNSAFE ?
                004012  001022                  BNE     7$              ;BR IF UNSAFE
                004014  032737  006000  016136          BIT     #BIT11!BIT10,DPB+STATUS  ;UNCORRECTABLE PARITY ERROR ?
                004022  001020                  BNE     8$              ;BR IF YES
                004024  032737  001400  016136          BIT     #BIT09!BIT08,DPB+STATUS  ;TIMEOUT ON THIS DRIVE ?
```

```
1014  004032  001016                        BNE     9$              ;BR IF YES
1015  004034  104010                5$:     ERROR   10              ;REPORT DRIVE OR DATA ERROR
1016  004036  032777  001000  175074        BIT     #SW09,@SWR      ;LOOP ON ERROR ?
1017  004044  001746                        BEQ     3$              ;BR IF NOT
1018  004046  162705  000006                SUB     #6,R5           ;CORRECT R5 FOR LOOP
1019  004052  000743                        BR      3$              ;RETURN
1020  004054  104016                6$:     ERROR   16              ;REPORT DRIVE UNAVAILABLE
1021  004056  000405                        BR      10$             ;CHECK FOR LOOP
1022  004060  104011                7$:     ERROR   11              ;REPORT PERSISTENT UNSAFE
1023  004062  000403                        BR      10$             ;CHECK FOR LOOP
1024  004064  104015                8$:     ERROR   15              ;UNCORRECTABLE PARITY ERROR
1025  004066  000401                        BR      10$             ;CHECK FOR LOOP
1026  004070  104014                9$:     ERROR   14              ;SOFTWARE TIMEOUT
1027  004072  032777  001000  175040  10$:  BIT     #SW09,@SWR;LOOP ON ERROR ?
1028  004100  001403                        BEQ     11$             ;BR IF NOT
1029  004102  162705  000006                SUB     #6,R5           ;CORRECT R5 FOR LOOP
1030  004106  000725                        BR      3$              ;RETURN
1031  004110  000137  001744          11$:  JMP     INIT            ;RETURN TO USER
1032
1033                                  ;;*****************************************************************
1034
1035                                  .SBTTL  SUBROUTINES
1036
1037                                  ;;*****************************************************************
1038
1039                                  ;THIS ROUTINE WILL DETERMINE IF THERE IS A CLOCK ON THE SYSTEM
1040                                  ;AND IF THERE IS IT WILL SETUP THE VECTOR AND START THE CLOCK.
1041                                  ;CALL
1042                                  ;       JSR     PC,@#ST.CLK
1043                                  ;       RETURN
1044
1045  004114  010146                ST.CLK: MOV     R1,-(SP)        ;SAVE R1
1046  004116  012701  000006                MOV     #ERRVEC+2,R1    ;SAVE AND SETUP TIMEOUT VECTOR
1047  004122  011146                        MOV     (R1),-(SP)
1048  004124  005011                        CLR     (R1)            ;LEVEL 0
1049  004126  014146                        MOV     -(R1),-(SP)
1050  004130  012711  004146                MOV     #1$,(R1)        ;GO TO 1$ ON TIMEOUT
1051  004134  005777  175110                TST     @PKCS           ;IS THERE A KW11-P?
1052  004140  004737  004200                JSR     PC,ST.PCLK      ;START THE KW11-P
1053  004144  000411                        BR      3$              ;GO TO EXIT
1054  004146  022626                1$:     CMP     (SP)+,(SP)+     ;CLEAN UP THE STACK
1055  004150  012711  004166                MOV     #2$,(R1)        ;IF TIMEOUT GO TO 2$
1056  004154  005777  175102                TST     @LKS            ;IS THERE A KW11-L?
1057  004160  004737  004232                JSR     PC,ST.LCLK      ;START THE KW11-L
1058  004164  000401                        BR      3$              ;EXIT
1059  004166  022626                2$:     CMP     (SP)+,(SP)+     ;CLEAN UP THE STACK
1060  004170  012621                3$:     MOV     (SP)+,(R1)+     ;RESTORE THE TIMEOUT VECTOR
1061  004172  012621                        MOV     (SP)+,(R1)+
1062  004174  012601                        MOV     (SP)+,R1        ;RESTORE R1
1063  004176  000207                        RTS     PC              ;RETURN
1064
1065  004200  012777  004256  175036  ST.PCLK: MOV   #SRVCLK,@PKV   ;SETUP THE KW11-P VECTOR
1066  004206  012777  000300  175032          MOV   #300,@PKV+2
1067  004214  012777  000001  175030          MOV   #1,@PKB         ;COUNT ONE TICK
```

```
1068  004222  012777  000115  175020          MOV     #115,@PKCS      ;"INT.EN." ,COUNT DOWN", "MODE 1 (REPEAT)",
1069                                                                  ;"LINE FREQ", AND "RUN"
1070  004230  000207                          RTS     PC              ;RETURN
1071
1072  004232  012777  004256  175016  ST.LCLK: MOV    #SRVCLK,@LKV    ;SETUP THE KW11-L VECTOR
1073  004240  012777  000300  175012          MOV     #300,@LKV+2
1074  004246  012777  000100  175006          MOV     #100,@LKS       ;START THE KW11-L
1075  004254  000207                          RTS     PC              ;RETURN
1076
1077  004256  012746  000020          SRVCLK: MOV     #16.,-(SP)      ;TIME PER TICK IN MILLISECONDS
1078  004262  004737  014110                  JSR     PC,@#RPTMR      ;COUNT THE ELASPED TIME
1079  004266  000002                          RTI                     ;RETURN AFTER INTERRUPT
1080
1081                                  ;ROUTINE TO GET DECIMAL KEYBOARD ENTRIES
1082                                  ;CALL
1083                                  ;       JSR     PC,GETNUM
1084                                  ;       RETURN1                 ;'CR' ENTERED
1085                                  ;       RETURN2                 ;NUMBER ENTERED, CONVERTED NUMBER ON THE STACK
1086
1087  004270  011646          GETNUM: MOV     (SP),-(SP)      ;PROVIDE SPACE FOR FIRST CHAR
1088  004272  104411          1$:     RDLIN                   ;READ AN ASCIZ LINE
1089  004274  012600                  MOV     (SP)+,R0        ;ADDRESS OF 1ST CHAR
1090  004276  010037  004376          MOV     R0,6$           ;SAVE IN CASE OF BAD INPUT
1091  004302  105710                  TSTB    (R0)            ;FIRST CHARACTER A 'CR' ?
1092  004304  001440                  BEQ     7$              ;BR IF IT IS
1093  004306  005046                  CLR     -(SP)           ;CLEAR DATA WORD
1094  004310  112001          2$:     MOVB    (R0)+,R1        ;PICKUP THIS CHARACTER
1095  004312  001421                  BEQ     4$              ;GET OUT IF ZERO
1096  004314  122701  000060          CMPB    #'0,R1          ;MAKE SURE THAT THIS CHARACTER
1097  004320  003016                  BGT     4$              ;IS A DIGIT BETWEEN 0 & 9
1098  004322  122701  000071          CMPB    #'9,R1
1099  004326  002420                  BLT     5$
1100  004330  006316                  ASL     (SP)            ;*2
1101  004332  011646                  MOV     (SP),-(SP)      ;SAVE FOR LATER
1102  004334  006316                  ASL     (SP)            ;*4
1103  004336  006316                  ASL     (SP)            ;*8.
1104  004340  062616                  ADD     (SP)+,(SP)      ;*10.
1105  004342  102412                  BVS     5$              ;OVERFLOW ISN'T ALLOWED
1106  004344  042701  000060          BIC     #60,R1          ;CLEAR THE UPPER BITS
1107  004350  060116                  ADD     R1,(SP)         ;ADD THE NUMBER
1108  004352  102406                  BVS     5$              ;OVERFLOW ISN'T ALLOWED
1109  004354  000755                  BR      2$              ;LOOP
1110  004356  012666  000002  4$:     MOV     (SP)+,2(SP)     ;SAVE THE RESULT
1111  004362  062716  000002          ADD     #2,(SP)         ;INCREMENT THE RETURN ADDRESS
1112  004366  000410                  BR      8$              ;EXIT
1113  004370  005726          5$:     TST     (SP)+           ;CLEAN PARTIAL NUMBER FROM STACK
1114  004372  105010                  CLRB    (R0)            ;SET A TERMINATOR
1115  004374  104401                  TYPE                    ;TYPE THE INPUT UP TO BAD CHAR
1116  004376  000000          6$:     .WORD   0               ;POINTER GOES HERE
1117  004400  104401  001160          TYPE    $QUES           ;'?' 'CR' 'LF'
1118  004404  000732                  BR      1$              ;TRY AGAIN
1119  004406  012616          7$:     MOV     (SP)+,(SP)      ;RESTORE THE PC
1120  004410  000207          8$:     RTS     PC              ;RETURN
1121
```

```
                                        ;;*****************************************************************
                                        .SBTTL  MACRO ROUTINES

                                        .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 $ERRTYP ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1         HALT ON ERROR
                                        ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

004412                                  $ERROR:
004412  104407                                 CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
004414  010137  001166                         MOV     R1,DRIVE
004420  010337  001170                         MOV     R3,ATTN
004424  105237  001103                 7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
004430  001775                               BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
004432  013777  001102  174502               MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
004440  005237  001112                       INC     $ERTTL          ;;INC THE ERROR COUNT
004444  011637  001116                       MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
004450  162737  000002  001116               SUB     #2,$ERRPC
004456  117737  174434  001114               MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
004464  032777  020000  174446               BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
004472  001004                               BNE     20$             ;;SKIP TYPEOUTS
004474  004737  004520                       JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
004500  104401  001161                       TYPE    ,$CRLF
004504                                  20$:
004504  005777  174430                 2$:     TST     @SWR            ;;HALT ON ERROR
004510  100002                               BPL     3$              ;;SKIP IF CONTINUE
004512  000000                               HALT                    ;;HALT ON ERROR!
004514  104407                               CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
004516                                  3$:
004516  000002                               RTI                     ;;RETURN
                                        .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

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

004520                                  $ERRTYP:
004520  104401  001161                         TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
004524  010046                               MOV     R0,-(SP)        ;;SAVE R0
004526  005000                               CLR     R0              ;;PICKUP THE ITEM INDEX
004530  153700  001114                       BISB    @#$ITEMB,R0
004534  001004                               BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                        ;;TYPE THE PC OF THE ERROR
004536  013746  001116                       MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                        ;;ERROR ADDRESS
```

```
1176  004542  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1177  004544  000445                        BR      10$             ;;GET OUT
1178  004546  005300                1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
1179  004550  006300                        ASL     R0              ;;      WORK FOR THE ERROR TABLE
1180  004552  006300                        ASL     R0
1181  004554  006300                        ASL     R0
1182  004556  062700  001270              ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
1183  004562  012037  004572              MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
1184  004566  001404                      BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
1185  004570  104401                      TYPE                    ;;TYPE THE "ERROR MESSAGE"
1186  004572  000000                2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
1187  004574  104401  001161              TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
1188  004600  012037  004610        3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
1189  004604  001404                      BEQ     5$              ;;SKIP TYPEOUT IF 0
1190  004606  104401                      TYPE                    ;;TYPE THE "DATA HEADER"
1191  004610  000000                4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
1192  004612  104401  001161              TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
1193  004616  010146                5$:     MOV     R1,-(SP)        ;;SAVE R1
1194  004620  012001                        MOV     (R0)+,R1        ;;PICKUP "DATA TABLE" POINTER
1195  004622  001415                        BEQ     9$              ;;BR IF NO DATA TO BE TYPED
1196  004624  012000                        MOV     (R0)+,R0        ;;PICKUP "DATA FORMAT" POINTER
1197  004626  105720                6$:     TSTB    (R0)+           ;;"OCTAL" OR "DECIMAL"
1198  004630  001003                        BNE     7$              ;;BR IF DECIMAL
1199  004632  013146                        MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
1200  004634  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1201  004636  000402                        BR      8$
1202  004640                        7$:
1203  004640  013146                        MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
1204  004642  104405                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
1205  004644  005711                8$:     TST     (R1)            ;;IS THERE ANOTHER NUMBER?
1206  004646  001403                        BEQ     9$              ;;BR IF NO
1207  004650  104401  004670              TYPE    ,11$            ;;TYPE TWO(2) SPACES
1208  004654  000764                        BR      6$              ;;LOOP
1209
1210  004656  012601                9$:     MOV     (SP)+,R1        ;;RESTORE R1
1211  004660  012600                10$:    MOV     (SP)+,R0        ;;RESTORE R0
1212  004662  104401  001161              TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
1213  004666  000207                      RTS     PC              ;;RETURN
1214  004670  020040      000       11$:    .ASCIZ  /  /            ;;TWO(2) SPACES
1215          004674                        .EVEN
1216                                .SBTTL  TYPE ROUTINE
1217
1218                                ;;******************************************************************
1219                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1220                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1221                                ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1222                                ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1223                                ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1224                                ;*
1225                                ;*CALL:
1226                                ;*1) USING A TRAP INSTRUCTION
1227                                ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1228                                ;*OR
1229                                ;*      TYPE
```

```
1230                                    ;*      MESADR
1231                                    ;*
1232
1233  004674  105737  001157    $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
1234  004700  100002                    BPL     1$              ;;BR IF YES
1235  004702  000000                    HALT                    ;;HALT HERE IF NO TERMINAL
1236  004704  000407                    BR      3$              ;;LEAVE
1237  004706  010046            1$:     MOV     R0,-(SP)        ;;SAVE R0
1238  004710  017600  000002            MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
1239  004714  112046            2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1240  004716  001005                    BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
1241  004720  005726                    TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
1242  004722  012600            60$:    MOV     (SP)+,R0        ;;RESTORE R0
1243  004724  062716  000002    3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
1244  004730  000002                    RTI                     ;;RETURN
1245  004732  122716  000011    4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
1246  004736  001430                    BEQ     8$
1247  004740  122716  000200            CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
1248  004744  001006                    BNE     5$
1249  004746  005726                    TST     (SP)+           ;;POP  <CR><LF> EQUIV
1250  004750  104401                    TYPE                    ;;TYPE A CR AND LF
1251  004752  001161                    $CRLF
1252  004754  105037  005110            CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
1253  004760  000755                    BR      2$              ;;GET NEXT CHARACTER
1254  004762  004737  005044    5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
1255  004766  123726  001156    6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
1256  004772  001350                    BNE     2$              ;;IF NO GO GET NEXT CHAR.
1257  004774  013746  001154            MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1258                                                            ;;AND THE NULL CHAR.
1259  005000  105366  000001    7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
1260  005004  002770                    BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
1261  005006  004737  005044            JSR     PC,$TYPEC       ;;GO TYPE A NULL
1262  005012  105337  005110            DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
1263  005016  000770                    BR      7$              ;;LOOP
1264
1265                                    ;HORIZONTAL TAB PROCESSOR
1266
1267  005020  112716  000040    8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
1268  005024  004737  005044    9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
1269  005030  132737  000007  005110    BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
1270  005036  001372                    BNE     9$              ;;TAB STOP
1271  005040  005726                    TST     (SP)+           ;;POP SPACE OFF STACK
1272  005042  000724                    BR      2$              ;;GET NEXT CHARACTER
1273  005044  105777  174100    $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
1274  005050  100375                    BPL     $TYPEC
1275  005052  116677  000002  174072    MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1276  005060  122766  000015  000002    CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
1277  005066  001003                    BNE     1$              ;;BRANCH IF NO
1278  005070  105037  005110            CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1279  005074  000406                    BR      $TYPEX          ;;EXIT
1280  005076  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
1281  005104  001402                    BEQ     $TYPEX          ;;BRANCH IF YES
1282  005106  105227                    INCB    (PC)+           ;;COUNT THE CHARACTER
1283  005110  000000    $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
```

```
1284  005112  000207                    $TYPEX: RTS     PC
1285
1286                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1287
1288                                    ;;*****************************************************************
1289                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1290                                    ;*OCTAL (ASCII) NUMBER AND TYPE IT.
1291                                    ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1292                                    ;*CALL:
1293                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1294                                    ;*      TYPOS                   ;;CALL FOR TYPEOUT
1295                                    ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1296                                    ;*      .BYTE   M               ;;M=1 OR 0
1297                                    ;*                                      ;;1=TYPE LEADING ZEROS
1298                                    ;*                                      ;;0=SUPPRESS LEADING ZEROS
1299                                    ;*
1300                                    ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1301                                    ;*$TYPOS OR $TYPOC
1302                                    ;*CALL:
1303                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1304                                    ;*      TYPON                   ;;CALL FOR TYPEOUT
1305                                    ;*
1306                                    ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1307                                    ;*CALL:
1308                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1309                                    ;*      TYPOC                   ;;CALL FOR TYPEOUT
1310
1311  005114  017646  000000            $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
1312  005120  116637  000001  005337            MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
1313  005126  112637  005341                    MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
1314  005132  062716  000002                    ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
1315  005136  000406                            BR      $TYPON
1316  005140  112737  000001  005337    $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
1317  005146  112737  000006  005341            MOVB    #6.$OMODE+1     ;;SET FOR SIX(6) DIGITS
1318  005154  112737  000005  005336    $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
1319  005162  010346                            MOV     R3,-(SP)        ;;SAVE R3
1320  005164  010446                            MOV     R4,-(SP)        ;;SAVE R4
1321  005166  010546                            MOV     R5,-(SP)        ;;SAVE R5
1322  005170  113704  005341                    MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
1323  005174  005404                            NEG     R4
1324  005176  062704  000006                    ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
1325  005202  110437  005340                    MOVB    R4,$OMODE       ;;SAVE IT FOR USE
1326  005206  113704  005337                    MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
1327  005212  016605  000012                    MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
1328  005216  005003                            CLR     R3              ;;CLEAR THE OUTPUT WORD
1329  005220  006105                    1$:     ROL     R5              ;;ROTATE MSB INTO "C"
1330  005222  000404                            BR      3$              ;;GO DO MSB
1331  005224  006105                    2$:     ROL     R5              ;;FORM THIS DIGIT
1332  005226  006105                            ROL     R5
1333  005230  006105                            ROL     R5
1334  005232  010503                            MOV     R5,R3
1335  005234  006103                    3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
1336  005236  105337  005340                    DECB    $OMODE          ;;TYPE THIS DIGIT?
1337  005242  100016                            BPL     7$              ;;BR IF NO
```

```
 1338  005244  042703  177770            BIC     #177770,R3      ;;GET RID OF JUNK
 1339  005250  001002                    BNE     4$              ;;TEST FOR 0
 1340  005252  005704                    TST     R4              ;;SUPPRESS THIS 0?
 1341  005254  001403                    BEQ     5$              ;;BR IF YES
 1342  005256  005204            4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
 1343  005260  052703  000060            BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
 1344  005264  052703  000040    5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
 1345  005270  110337  005334            MOVB    R3,8$           ;;SAVE FOR TYPING
 1346  005274  104401  005334            TYPE    ,8$             ;;GO TYPE THIS DIGIT
 1347  005300  105337  005336    7$:     DECB    $OCNT           ;;COUNT BY 1
 1348  005304  003347                    BGT     2$              ;;BR IF MORE TO DO
 1349  005306  002402                    BLT     6$              ;;BR IF DONE
 1350  005310  005204                    INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
 1351  005312  000744                    BR      2$              ;;GO DO THE LAST DIGIT
 1352  005314  012605            6$:     MOV     (SP)+,R5        ;;RESTORE R5
 1353  005316  012604                    MOV     (SP)+,R4        ;;RESTORE R4
 1354  005320  012603                    MOV     (SP)+,R3        ;;RESTORE R3
 1355  005322  016666  000002  000004    MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
 1356  005330  012616                    MOV     (SP)+,(SP)
 1357  005332  000002                    RTI                     ;;RETURN
 1358  005334  000000            8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
 1359  005335  000000                    .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
 1360  005336  000000            $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
 1361  005337  000000            $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
 1362  005340  000000            $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
 1363                            .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 1364
 1365                            ;;**************************************************************
 1366                            ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
 1367                            ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
 1368                            ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
 1369                            ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
 1370                            ;*REPLACED WITH SPACES.
 1371                            ;*CALL:
 1372                            ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
 1373                            ;*      TYPDS                   ;;GO TO THE ROUTINE
 1374
 1375  005342                    $TYPDS:
 1376  005342  010046                    MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 1377  005344  010146                    MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 1378  005346  010246                    MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 1379  005350  010346                    MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 1380  005352  010546                    MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 1381  005354  012746  020200            MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
 1382  005360  016605  000020            MOV     20(SP),R5       ;;GET THE INPUT NUMBER
 1383  005364  100004                    BPL     1$              ;;BR IF INPUT IS POS.
 1384  005366  005405                    NEG     R5              ;;MAKE THE BINARY NUMBER POS.
 1385  005370  112766  000055  000001    MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
 1386  005376  005000            1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
 1387  005400  012703  005556            MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
 1388  005404  112723  000040            MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
 1389  005410  005002            2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
 1390  005412  016001  005546            MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
 1391  005416  160105            3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
```

```
                                                BLT     4$              ;;BR IF DONE
                                                INC     R2              ;;INCREASE THE BCD DIGIT BY 1
                                                BR      3$
                                        4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
                                                TST     R2              ;;CHECK IF BCD DIGIT=0
                                                BNE     5$              ;;FALL THROUGH IF 0
                                                TSTB    (SP)            ;;STILL DOING LEADING 0'S?
                                                BMI     7$              ;;BR IF YES
                                        5$:     ASLB    (SP)            ;;MSD?
                                                BCC     6$              ;;BR IF NO
                                                MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
                                        6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
                                        7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
                                                MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
                                                TST     (R0)+           ;;JUST INCREMENTING
                                                CMP     R0,#10          ;;CHECK THE TABLE INDEX
                                                BLT     2$              ;;GO DO THE NEXT DIGIT
                                                BGT     9$              ;;GO TO EXIT
                                                MOV     R5,R2           ;;GET THE LSD
                                                BR      6$              ;;GO CHANGE TO ASCII
                                        8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
                                                BPL     9$              ;;BR IF NO
                                                MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
                                        9$:     CLRB    (R3)            ;;SET THE TERMINATOR
                                                MOV     (SP)+,R5        ;;POP STACK INTO R5
                                                MOV     (SP)+,R3        ;;POP STACK INTO R3
                                                MOV     (SP)+,R2        ;;POP STACK INTO R2
                                                MOV     (SP)+,R1        ;;POP STACK INTO R1
                                                MOV     (SP)+,R0        ;;POP STACK INTO R0
                                                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
                                                MOV     2(SP),4(SP)     ;;ADJUST THE STACK
                                                MOV     (SP)+,(SP)
                                                RTI                     ;;RETURN TO USER
                                        $DTBL:  10000.
                                                1000.
                                                100.
                                                10.
                                        $DBLK:  .BLKW   4
                                        .SBTTL  TTY INPUT ROUTINE

                                        ;;******************************************************************
                                        .ENABL  LSB
                                        $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
                                        $TKQIN: .WORD   0               ;;INPUT POINTER
                                        $TKQOUT: .WORD  0               ;;OUTPUT POINTER
                                        $TKQSRT: .BLKB  7               ;;TTY KEYBOARD QUEUE
                                        $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
                                        ;
005604  005037  005566                  $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
005610  012737  005574  005570                  MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
005616  013737  005570  005572                  MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
005624  012737  005654  000060                  MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
005632  012737  000200  000062                  MOV     #200,@#TKVEC+2  ;;"BR" LEVEL 4
005640  005777  173302                          TST     @$TKB           ;;CLEAR DONE FLAG
005644  012777  000100  173272                  MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
005652  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 (START1)
                                        ;
005654  117746  173266                  $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
005660  042716  177600                          BIC     #↑C177,(SP)     ;;STRIP THE JUNK
005664  021627  000003                          CMP     (SP),#3         ;;IS IT A CONTROL C?
005670  001007                                  BNE     1$              ;;BRANCH IF NO
005672  104401  006773                          TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (↑C)
005676  004737  005604                          JSR     PC,$TKINT       ;;INIT THE KEYBOARD
005702  005726                                  TST     (SP)+           ;;CLEAN UP STACK
005704  000137  001460                          JMP     START1          ;;CONTROL C RESTART
005710  021627  000007                  1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
005714  001004                                  BNE     2$              ;;BRANCH IF NO
005716  022737  000176  001140                  CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
005724  001500                                  BEQ     6$              ;;GO TO SWR CHANGE

005726                                  2$:
005726  022737  000007  005566                  CMP     #7,$TKCNT       ;;IS THE QUEUE FULL?
005734  001004                                  BNE     3$              ;;BRANCH IF NO
005736  104401  006767                          TYPE    ,$BELL          ;;RING THE TTY BELL
005742  005726                                  TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
005744  000451                                  BR      5$              ;;EXIT
005746  021627  000023                  3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
005752  001021                                  BNE     32$             ;;BRANCH IF NO
005754  005077  173164                          CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
005760  005726                                  TST     (SP)+           ;;CLEAN CHAR OFF STACK
005762  105777  173156                  31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
005766  100375                                  BPL     31$             ;;LOOP UNTIL ITS THERE
005770  117746  173152                          MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
005774  042716  177600                          BIC     #↑C177,(SP)     ;;MAKE IT 7-BIT ASCII
006000  022627  000021                          CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
006004  001366                                  BNE     31$             ;;BRANCH IF NO
006006  012777  000100  173130                  MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
006014  000002                                  RTI                     ;;RETURN
006016  005237  005566                  32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
006022  021627  000140                          CMP     (SP),#140       ;;IS IT UPPER CASE?
006026  002405                                  BLT     4$              ;;BRANCH IF YES
006030  021627  000175                          CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
```

```
 1500  006034  003002                        BGT     4$                      ;;BRANCH IF YES
 1501  006036  042716  000040                BIC     #40,(SP)                ;;MAKE IT UPPER CASE
 1502  006042  112677  177522          4$:   MOVB    (SP)+,@$TKQIN           ;;AND PUT IT IN QUEUE
 1503  006046  005237  005570                INC     $TKQIN                  ;;UPDATE THE POINTER
 1504  006052  023727  005570  005603        CMP     $TKQIN,#$TKQEND         ;;GO OFF THE END?
 1505  006060  001003                        BNE     5$                      ;;BRANCH IF NO
 1506  006062  012737  005574  005570        MOV     #$TKQSRT,$TKQIN         ;;RESET THE POINTER
 1507  006070  000002                  5$:   RTI                             ;;RETURN
 1508
 1509                                        ;;**********************************************************************
 1510                                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
 1511                                        ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
 1512                                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
 1513                                        ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
 1514  006072  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR              ;;IS THE SOFT-SWR SELECTED
 1515  006100  001124                        BNE     15$                     ;;EXIT IF NOT
 1516  006102  105777  173036                TSTB    @$TKS                   ;;IS A CHAR WAITING?
 1517  006106  100121                        BPL     15$                     ;;IF NOT, EXIT
 1518  006110  117746  173032                MOVB    @$TKB,-(SP)             ;;YES
 1519  006114  042716  177600                BIC     #^C177,(SP)             ;;MAKE IT  7-BIT ASCII
 1520  006120  021627  000007                CMP     (SP),#7                 ;;IS IT A CONTROL-G?
 1521  006124  001300                        BNE     2$                      ;;IF NOT, PUT IT IN THE TTY QUEUE
 1522                                                                        ;;AND EXIT
 1523
 1524                                        ;;**********************************************************************
 1525                                        ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
 1526                                        ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
 1527                                        ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
 1528  006126  123727  001134  000001  6$:   CMPB    $AUTOB,#1               ;;ARE WE RUNNING IN AUTO-MODE?
 1529  006134  001674                        BEQ     2$                      ;;BRANCH IF YES
 1530  006136  005726                        TST     (SP)+                   ;;CLEAR CONTROL-G OFF STACK
 1531  006140  004737  005604                JSR     PC,$TKINT               ;;FLUSH THE TTY INPUT QUEUE
 1532  006144  005077  172774                CLR     @$TKS                   ;;DISABLE TTY KEYBOARD INTERRUPTS
 1533  006150  112737  000001  001135        MOVB    #1,$INTAG               ;;SET INTERRUPT MODE INDICATOR
 1534
 1535  006156  104401  007005                TYPE    ,$CNTLG                 ;;ECHO THE CONTROL-G (^G)
 1536  006162  104401  007012          $GTSWR: TYPE  ,$MSWR                  ;;TYPE CURRENT CONTENTS
 1537  006166  013746  000176                MOV     $SWREG,-(SP)            ;;SAVE SWREG FOR TYPEOUT
 1538  006172  104402                        TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1539  006174  104401  007023                TYPE    ,$MNEW                  ;;PROMPT FOR NEW SWR
 1540  006200  005046                  19$:  CLR     -(SP)                   ;;CLEAR COUNTER
 1541  006202  005046                        CLR     -(SP)                   ;;THE NEW SWR
 1542  006204  105777  172734          7$:   TSTB    @$TKS                   ;;CHAR THERE?
 1543  006210  100375                        BPL     7$                      ;;IF NOT TRY AGAIN
 1544
 1545  006212  117746  172730                MOVB    @$TKB,-(SP)             ;;PICK UP CHAR
 1546  006216  042716  177600                BIC     #^C177,(SP)             ;;MAKE IT 7-BIT ASCII
 1547
 1548  006222  021627  000003                CMP     (SP),#3                 ;;IS IT A CONTROL-C?
 1549  006226  001015                        BNE     9$                      ;;BRANCH IF NOT
 1550  006230  104401  006773                TYPE    ,$CNTLC                 ;;YES, ECHO CONTROL-C (^C)
 1551  006234  062706  000006                ADD     #6,SP                   ;;CLEAN UP STACK
 1552  006240  123727  001135  000001        CMPB    $INTAG,#1               ;;REENABLE TTY KEYBOARD INTERRUPTS?
 1553  006246  001003                        BNE     8$                      ;;BRANCH IF NO
```

```
1554  006250  012777  000100  172666          MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
1555  006256  000137  001460          8$:     JMP     START1          ;;CONTROL-C RESTART
1556
1557
1558  006262  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
1559  006266  001005                          BNE     10$             ;;BRANCH IF NOT
1560  006270  104401  007000                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
1561  006274  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
1562  006300  000737                          BR      19$             ;;LET'S TRY IT AGAIN
1563
1564
1565  006302  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
1566  006306  001022                          BNE     16$             ;;BRANCH IF NO
1567  006310  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
1568  006314  001403                          BEQ     11$             ;;BRANCH IF YES
1569  006316  016677  000002  172614          MOV     2(SP),@$SWR     ;;SAVE NEW SWR
1570  006324  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
1571  006330  104401  001161          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
1572  006334  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
1573  006342  001003                          BNE     15$             ;;BRANCH IF NOT
1574  006344  012777  000100  172572          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
1575  006352  000002          15$:    RTI                             ;;RETURN
1576  006354  004737  005044          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
1577  006360  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
1578  006364  002420                          BLT     18$             ;;BRANCH IF YES
1579  006366  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
1580  006372  003015                          BGT     18$             ;;BRANCH IF YES
1581  006374  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
1582  006400  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
1583  006404  001403                          BEQ     17$             ;;BRANCH IF YES
1584  006406  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
1585  006410  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
1586  006412  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
1587  006414  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
1588  006420  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
1589  006424  000667                          BR      7$              ;;GET THE NEXT ONE
1590  006426  104401  001160          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
1591  006432  000720                          BR      20$             ;;SIMULATE CONTROL-U
1592                                  .DSABL  LSB
1593
1594
1595                                  ;;**************************************************************
1596                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1597                                  ;*CALL:
1598                                  ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
1599                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
1600                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
1601                                  ;
1602
1603  006434  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
1604  006436  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE PS
1605  006444  005066  000004                  CLR     4(SP)           ;;GET READY FOR A CHARACTER
1606  006450  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
1607  006452  012746  006460                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
```

```
1608  006456  000002                        RTI                     ;;POP NEW PC AND PS
1609  006460                        64$:
1610  006460  005737  005566        1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
1611  006464  001775                        BEQ     1$
1612  006466  005337  005566                DEC     $TKCNT          ;;DECREMENT THE COUNTER
1613  006472  117766  177074  000004        MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
1614  006500  005237  005572                INC     $TKQOUT         ;;UPDATE THE POINTER
1615  006504  023727  005572  005603        CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
1616  006512  001003                        BNE     2$              ;;BRANCH IF NO
1617  006514  012737  005574  005572        MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
1618  006522  000002                2$:     RTI                     ;;RETURN
1619                                ;;**************************************************************
1620                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1621                                ;*CALL:
1622                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
1623                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1624                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
1625
1626  006524  010346                $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
1627  006526  005046                        CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
1628  006530  012703  006760        1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
1629  006534  022703  006767        2$:     CMP     #$TTYIN+7,R3    ;;BUFFER FULL?
1630  006540  101456                        BLOS    4$              ;;BR IF YES
1631  006542  104410                        RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
1632  006544  112613                        MOVB    (SP)+,(R3)      ;;GET CHARACTER
1633  006546  122713  000177        10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
1634  006552  001022                        BNE     5$              ;;BR IF NO
1635  006554  005716                        TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
1636  006556  001007                        BNE     6$              ;;BR IF NO
1637  006560  112737  000134  006756        MOVB    #'\,9$          ;;TYPE A BACK SLASH
1638  006566  104401  006756                TYPE    ,9$
1639  006572  012716  177777                MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
1640  006576  005303                6$:     DEC     R3              ;;BACKUP BY ONE
1641  006600  020327  006760                CMP     R3,#$TTYIN      ;;STACK EMPTY?
1642  006604  103434                        BLO     4$              ;;BR IF YES
1643  006606  111337  006756                MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
1644  006612  104401  006756                TYPE    ,9$             ;;GO TYPE
1645  006616  000746                        BR      2$              ;;GO READ ANOTHER CHAR.
1646  006620  005716                5$:     TST     (SP)            ;;RUBOUT KEY SET?
1647  006622  001406                        BEQ     7$              ;;BR IF NO
1648  006624  112737  000134  006756        MOVB    #'\,9$          ;;TYPE A BACK SLASH
1649  006632  104401  006756                TYPE    ,9$
1650  006636  005016                        CLR     (SP)            ;;CLEAR THE RUBOUT KEY
1651  006640  122713  000025        7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
1652  006644  001003                        BNE     8$              ;;BR IF NO
1653  006646  104401  007000                TYPE    ,$CNTLU         ;;TYPE A CONTROL "U"
1654  006652  000726                        BR      1$              ;;GO START OVER
1655  006654  122713  000022        9$:     CMPB    #22,(R3)        ;;IS CHARACTER A "↑R"?
1656  006660  001011                        BNE     3$              ;;BRANCH IF NO
1657  006662  105013                        CLRB    (R3)            ;;CLEAR THE CHARACTER
1658  006664  104401  001161                TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
1659  006670  104401  006760                TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
1660  006674  000717                        BR      2$              ;;GO PICKUP ANOTHER CHACTER
1661  006676  104401  001160        4$:     TYPE    ,$QUES          ;;TYPE A '?'
```

```
1662  006702  000712                        BR      1$              ;;CLEAR THE BUFFER AND LOOP
1663  006704  111337  006756          3$:   MOVB    (R3),9$         ;;ECHO THE CHARACTER
1664  006710  104401  006756                TYPE    ,9$
1665  006714  122723  000015                CMPB    #15,(R3)+       ;;CHECK FOR RETURN
1666  006720  001305                        BNE     2$              ;;LOOP IF NOT RETURN
1667  006722  105063  177777                CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
1668  006726  104401  001162                TYPE    ,$LF            ;;TYPE A LINE FEED
1669  006732  005726                        TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
1670  006734  012603                        MOV     (SP)+,R3        ;;RESTORE R3
1671  006736  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
1672  006740  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
1673  006746  012766  006760  000004        MOV     #$TTYIN,4(SP)
1674  006754  000002                        RTI                     ;;RETURN
1675  006756     000          9$:   .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
1676  006757     000                .BYTE   0                       ;;TERMINATOR
1677  006760  000007          $TTYIN: .BLKB   7                     ;;RESERVE 7 BYTES FOR TTY INPUT
1678  006767     207  177777     000  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
1679  006773     136  006503  000012  $CNTLC: .ASCIZ  /↑C/<15><12>    ;;CONTROL "C"
1680  007000  052536  005015     000  $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
1681  007005     136  006507  000012  $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
1682  007012  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
1683  007020  020075     000
1684  007023     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
1685  007030  036440  000040
1686                                  .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
1687
1688                                  ;;*****************************************************************
1689                                  ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
1690                                  ;*WITH A RANGE OF 0 TO 2(+33)-1.
1691                                  ;*CALL:
1692                                  ;*      JSR     PC,$RAND        ;;CALL THE ROUTINE
1693                                  ;*      RETURN                  ;;RETURN HERE THE RANDOM
1694                                  ;*                              ;;NUMBER WILL BE IN
1695                                  ;*                              ;;$HINUM,$LONUM
1696
1697  007034                          $RAND:
1698  007034  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1699  007036  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1700  007040  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1701  007042  013700  007134                MOV     $LONUM,R0       ;;SET R0 WITH LOW
1702  007046  013701  007132                MOV     $HINUM,R1       ;;SET R1 WITH HIGH
1703  007052  012702  177771                MOV     #-7,R2          ;;SET SHIFT COUNT
1704  007056  006300          1$:   ASL     R0                      ;;SHIFT R0 LEFT AND
1705  007060  006101                ROL     R1                      ;;ROTATE CARRY INTO R1 AND
1706  007062  005202                INC     R2                      ;;CHECK FOR DONE
1707  007064  001374                BNE     1$                      ;;CONTINUE SHIFT LOOP
1708  007066  063700  007134                ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
1709  007072  005501                ADC     R1                      ;;PROPOGATE CARRY
1710  007074  063701  007132                ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
1711  007100  062700  001057                ADD     #1057,R0        ;;ADD LOW CONSTANT
1712  007104  005501                ADC     R1                      ;;PROPOGATE CARRY
1713  007106  062701  047401                ADD     #47401,R1       ;;ADD HIGH CONSTANT
1714  007112  010037  007134                MOV     R0,$LONUM       ;;SAVE R0
1715  007116  010137  007132                MOV     R1,$HINUM       ;;SAVE R1
```

```
 1716  007122  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
 1717  007124  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
 1718  007126  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
 1719  007130  000207                        RTS     PC              ;;RETURN
 1720  007132  176543              $HINUM:   .WORD   176543
 1721  007134  123456              $LONUM:   .WORD   123456
 1722                                .SBTTL  INTEGER DIVIDE ROUTINE
 1723
 1724                                ;;*****************************************************************
 1725                                ;*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
 1726                                ;*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
 1727                                ;*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
 1728                                ;*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
 1729                                ;*SAVE SIGN AS THE DIVIDEND.
 1730                                ;*CALL:
 1731                                ;*      MOV     LOW DIVIDEND,-(SP) ;;THE HIGH DIVIDEND MUST BE < 1/2
 1732                                ;*      MOV     HIGH DIVIDEND,-(SP);    AS LARGE AS THE DIVISOR
 1733                                ;*      MOV     DIVISOR,-(SP)
 1734                                ;*      JSR     PC,$DIV
 1735                                ;*      RETURN                  ;;QUOTIENT & REMAINDER ARE ON THE STACK
 1736                                ;*      "V"=0   IMPLIES NO ERROR
 1737                                ;*      "V"=1   IMPLIES ERROR OCCURRED
 1738                                ;*              "C"=0   DIVIDE OVERFLOW OCCURRED
 1739                                ;*              "C"=1   ATTEMPTED TO DIVIDE BY ZERO
 1740                                ;*
 1741                                ;*
 1742                                ;*      STACK   NO ERROR        OVERFLOW        DIVIDE BY ZERO
 1743                                ;*      -----   --------        --------        --------------
 1744                                ;*      TOP     REMAINDER       ALL ZEROS       ALL ONES
 1745                                ;*      +2      QUOTIENT        ALL ZEROS       ALL ONES
 1746
 1747  007136                        $DIV:
 1748  007136  104400                        TRAP                    ;;PUSH OLD PSW AND PC ON STACK
 1749  007140  042716  000017               BIC     #17,(SP)        ;;STRIP AWAY CONDITION CODES
 1750  007144  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 1751  007146  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 1752  007150  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 1753  007152  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 1754  007154  005046                        CLR     -(SP)           ;;SAVE A PLACE FOR SIGNS
 1755  007156  012746  000021               MOV     #17.,-(SP)      ;;SETUP THE ITERATION COUNTER
 1756  007162  016601  000024               MOV     24(SP),R1       ;;PICKUP THE DIVIDEND
 1757  007166  016600  000022               MOV     22(SP),R0
 1758  007172  100005                        BPL     1$              ;;CHECK THE SIGN
 1759  007174  105366  000003               DECB    3(SP)           ;;KEEP TRACK OF THE SIGN
 1760  007200  005400                        NEG     R0              ;;AND NEGATE THE ORIGINAL
 1761  007202  005401                        NEG     R1              ;;NUMBER
 1762  007204  005600                        SBC     R0
 1763  007206  016602  000020       1$:     MOV     20(SP),R2       ;;PICKUP THE DIVISOR
 1764  007212  002407                        BLT     2$              ;;CHECK THE SIGN
 1765  007214  003011                        BGT     3$              ;;DIVISOR OF 0 IS A NO-NO
 1766  007216  052766  000003  000014       BIS     #3,14(SP)       ;;SET "V" & "C"
 1767  007224  012700  177777               MOV     #-1,R0          ;;SET REMAINDER TO ALL ONES
 1768  007230  000424                        BR      7$              ;;EXIT
 1769  007232  005266  000002       2$:     INC     2(SP)           ;;KEEP TRACK OF DIVISORS SIGN
```

```
 1770  007236  000401                        BR      4$
 1771  007240  005402                3$:     NEG     R2              ;;NEGATE THE ORIGINAL NUMBER
 1772  007242  000241                4$:     CLC                     ;;CLEAR "C"
 1773  007244  000405                        BR      6$              ;;START FORMING QUOTIENT
 1774  007246  006100                5$:     ROL     R0              ;;POSITION MSB'S
 1775  007250  010003                        MOV     R0,R3           ;;COPY
 1776  007252  060203                        ADD     R2,R3           ;;COMPARE DIVIDEND & DIVISOR
 1777  007254  103001                        BCC     6$              ;;BR IF DIVIDEND > DIVISOR
 1778  007256  010300                        MOV     R3,R0           ;;REMAINDER AFTER THIS LOOP
 1779  007260  006101                6$:     ROL     R1              ;;QUOTIENT BIT ENTERS HERE
 1780  007262  005316                        DEC     (SP)            ;;DONE?
 1781  007264  001370                        BNE     5$              ;;BR IF NO
 1782  007266  005701                        TST     R1              ;;OVERFLOW?
 1783  007270  100005                        BPL     8$              ;;BR IF NO
 1784  007272  052766  000002  000014        BIS     #2,14(SP)       ;;SET "V" IN RETURN STATUS WORD
 1785  007300  005000                        CLR     R0              ;;SET REMAINDER TO ALL ZEROS
 1786  007302  010001                7$:     MOV     R0,R1           ;;COPY REMAINDER INTO QUOTIENT
 1787  007304  005726                8$:     TST     (SP)+           ;;CLEAR COUNTER FROM STACK
 1788  007306  005716                        TST     (SP)            ;;REMAINDER SIGN CORRECTION NEEDED?
 1789  007310  002004                        BGE     9$              ;;BR IF NO
 1790  007312  005400                        NEG     R0              ;;NEGATE REMAINDER
 1791  007314  105066  000001                CLRB    1(SP)           ;;CLEAR SIGN
 1792  007320  005316                        DEC     (SP)            ;;BUT DON'T FORGET QUOTIENT
 1793  007322  005726                9$:     TST     (SP)+           ;;QUOTIENT SIGN CORRECTION NEEDED?
 1794  007324  001401                        BEQ     10$             ;;BR IF NO
 1795  007326  005401                        NEG     R1              ;;NEGATE QUOTIENT
 1796  007330  010166  000020        10$:    MOV     R1,20(SP)       ;;RETURN QUOTIENT AND
 1797  007334  010066  000016                MOV     R0,16(SP)       ;;REMAINDER TO USER
 1798  007340  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
 1799  007342  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
 1800  007344  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
 1801  007346  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
 1802  007350  012666  000002                MOV     (SP)+,2(SP)     ;;SETUP TO RETURN CONDITION CODES
 1803  007354  000002                        RTI                     ;;RETURN
 1804                          .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
 1805
 1806                          ;;*****************************************************************
 1807                          ;*SAVE R0-R5
 1808                          ;*CALL:
 1809                          ;*      SAVREG
 1810                          ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
 1811                          ;*
 1812                          ;*TOP---(+16)
 1813                          ;* +2---(+18)
 1814                          ;* +4---R5
 1815                          ;* +6---R4
 1816                          ;* +8---R3
 1817                          ;*+10---R2
 1818                          ;*+12---R1
 1819                          ;*+14---R0
 1820
 1821  007356                  $SAVREG:
 1822  007356  010046                MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 1823  007360  010146                MOV     R1,-(SP)        ;;PUSH R1 ON STACK
```

```
1824  007362  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1825  007364  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1826  007366  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
1827  007370  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
1828  007372  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
1829  007376  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
1830  007402  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
1831  007406  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
1832  007412  000002                          RTI
1833
1834                                  ;*RESTORE R0-R5
1835                                  ;*CALL:
1836                                  ;*      RESREG
1837  007414                          $RESREG:
1838  007414  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
1839  007420  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
1840  007424  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
1841  007430  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
1842  007434  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
1843  007436  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
1844  007440  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
1845  007442  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
1846  007444  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
1847  007446  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
1848  007450  000002                          RTI
1849                                  .SBTTL  TRAP DECODER
1850
1851                                  ;;**************************************************************
1852                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
1853                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1854                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
1855                                  ;*GO TO THAT ROUTINE.
1856
1857  007452  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
1858  007454  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
1859  007460  005740                          TST     -(R0)           ;;BACKUP BY 2
1860  007462  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
1861  007464  006300                          ASL     R0              ;;POSITION FOR INDEXING
1862  007466  016000  007506                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
1863  007472  000200                          RTS     R0              ;;GO TO ROUTINE
1864
1865
1866                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
1867
1868  007474  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
1869  007476  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
1870  007504  000002                          RTI                     ;;RESTORE THE PSW
1871
1872                                  .SBTTL  TRAP TABLE
1873
1874                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
1875                                  ;*BY THE "TRAP" INSTRUCTION.
1876
1877                                  ;       ROUTINE
```

```
                                        .       -------
007506  007474                  $TRPAD: .WORD   $TRAP2
007510  004674                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
007512  005140                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
007514  005114                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
007516  005154                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
007520  005342                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

007522  006162                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

007524  006072                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
007526  006434                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
007530  006524                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
007532  007356                          $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
007534  007414                          $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE


                                .SBTTL  SINGLE/DUAL PORT RH11/RP04/5/6 DRIVER (REV 1.0)

                                ;COPYRIGHT (C) 1976
                                ;DIGITAL EQUIPMENT CORP.
                                ;MAYNARD, MA 01754
                                ;AUTHOR(S): JIM LACEY/CHUCK HESS

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

                                ;STORAGE FOR RPDS1, RPER1, RPER2, AND RPER3 ON AN ERROR "2"
                                        ;RPERRS   = RPDS1
                                        ;RPERRS+2 = RPER1
                                        ;RPERRS+4 = RPER2
                                        ;RPERRS+6 = RPER3

007536  000000  000000  000000  RPERRS: .WORD   0,0,0,0
007544  000000

                                ;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

007546      000                 DRVACT: .BYTE   0               ;DRIVE 0
007547      000                         .BYTE   0               ;DRIVE 1
007550      000                         .BYTE   0               ;DRIVE 2
007551      000                         .BYTE   0               ;DRIVE 3
007552      000                         .BYTE   0               ;DRIVE 4
007553      000                         .BYTE   0               ;DRIVE 5
007554      000                         .BYTE   0               ;DRIVE 6
007555      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
```

```
1932  007556  000                    DRVSTA: .BYTE   0               ;DRIVE 0
1933  007557  000                            .BYTE   0               ;DRIVE 1
1934  007560  000                            .BYTE   0               ;DRIVE 2
1935  007561  000                            .BYTE   0               ;DRIVE 3
1936  007562  000                            .BYTE   0               ;DRIVE 4
1937  007563  000                            .BYTE   0               ;DRIVE 5
1938  007564  000                            .BYTE   0               ;DRIVE 6
1939  007565  000                            .BYTE   0               ;DRIVE 7
1940
1941                                 ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
1942                                         ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
1943                                         ;DRVTYP=1 IF DRIVE IS RP04
1944                                         ;DRVTYP=2 IF DRIVE IS RP05
1945                                         ;DRVTYP=4 IF DRIVE IS RP06
1946                                         ;DRVTYP=-1 IF NOT RP04/5/6
1947
1948  007566  000                    DRVTYP: .BYTE   0               ;DRIVE 0
1949  007567  000                            .BYTE   0               ;DRIVE 1
1950  007570  000                            .BYTE   0               ;DRIVE 2
1951  007571  000                            .BYTE   0               ;DRIVE 3
1952  007572  000                            .BYTE   0               ;DRIVE 4
1953  007573  000                            .BYTE   0               ;DRIVE 5
1954  007574  000                            .BYTE   0               ;DRIVE 6
1955  007575  000                            .BYTE   0               ;DRIVE 7
1956
1957                                 ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
1958                                         ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
1959                                         ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
1960
1961  007576  000                    DPINT:  .BYTE   0               ;DRIVE 0
1962  007577  000                            .BYTE   0               ;DRIVE 1
1963  007600  000                            .BYTE   0               ;DRIVE 2
1964  007601  000                            .BYTE   0               ;DRIVE 3
1965  007602  000                            .BYTE   0               ;DRIVE 4
1966  007603  000                            .BYTE   0               ;DRIVE 5
1967  007604  000                            .BYTE   0               ;DRIVE 6
1968  007605  000                            .BYTE   0               ;DRIVE 7
1969
1970                                 ;TABLE OF PENDING DUAL PORT REQUESTS
1971                                         ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
1972                                         ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
1973
1974  007606  000                    DPRQS:  .BYTE   0               ;DRIVE 0
1975  007607  000                            .BYTE   0               ;DRIVE 1
1976  007610  000                            .BYTE   0               ;DRIVE 2
1977  007611  000                            .BYTE   0               ;DRIVE 3
1978  007612  000                            .BYTE   0               ;DRIVE 4
1979  007613  000                            .BYTE   0               ;DRIVE 5
1980  007614  000                            .BYTE   0               ;DRIVE 6
1981  007615  000                            .BYTE   0               ;DRIVE 7
1982
1983                                 ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
1984                                         ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
1985                                         ;"DPB" OF THE I/O OPERATION.
```

```
1986
1987  007616  000000                    TRNSWT: .WORD   0
1988
1989                                    ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
1990                                            ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
1991                                            ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
1992                                            ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
1993                                            ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
1994
1995  007620  000000                    SRCHWT: .WORD   0
1996
1997                                    ;RP04/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
1998                                            ;ACTDRV=0 IF DRIVER IS INACTIVE
1999                                            ;ACTDRV>0 IF DRIVER IS ACTIVE
2000
2001  007622      000                   ACTDRV: .BYTE   0
2002
2003                                    ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
2004                                            ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
2005                                            ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
2006
2007  007623      000                   ACTSTR: .BYTE   0
2008
2009                                    ;UNLOAD FLAG (ULDFLG=8 BYTES)
2010                                            ;ULDFLG=0 IF NO UNLOAD COMMAND
2011                                            ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
2012                                            ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
2013
2014  007624      000                   ULDFLG: .BYTE   0               ;DRIVE 0
2015  007625      000                           .BYTE   0               ;DRIVE 1
2016  007626      000                           .BYTE   0               ;DRIVE 2
2017  007627      000                           .BYTE   0               ;DRIVE 3
2018  007630      000                           .BYTE   0               ;DRIVE 4
2019  007631      000                           .BYTE   0               ;DRIVE 5
2020  007632      000                           .BYTE   0               ;DRIVE 6
2021  007633      000                           .BYTE   0               ;DRIVE 7
2022
2023                                    ;LOOK AHEAD COUNT (LACNT=8 BYTES)
2024                                            ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
2025
2026  007634      000                   LACNT:  .BYTE   0               ;DRIVE 0
2027  007635      000                           .BYTE   0               ;DRIVE 1
2028  007636      000                           .BYTE   0               ;DRIVE 2
2029  007637      000                           .BYTE   0               ;DRIVE 3
2030  007640      000                           .BYTE   0               ;DRIVE 4
2031  007641      000                           .BYTE   0               ;DRIVE 5
2032  007642      000                           .BYTE   0               ;DRIVE 6
2033  007643      000                           .BYTE   0               ;DRIVE 7
2034
2035                                    ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
2036                                            ;SAVEFG <0 IF SAVE THE RH11/RP04/5/6 REGISTERS WHEN THE
2037                                            ;OPERATION IS COMPLETED AS PER (DPB+14).
2038                                            ;SAVEFG=0 IF SAVE THE RH11/RP04/5/6 REGISTERS, AS PER
2039                                            ;(DPB+14), AFTER AN ERROR.
```

```
                                
007644  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

007646  000000                  SEEKFG: .WORD   0

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

007650  177777                  TIMER:  .WORD   -1       ;DRIVE 0
007652  177777                          .WORD   -1       ;DRIVE 1
007654  177777                          .WORD   -1       ;DRIVE 2
007656  177777                          .WORD   -1       ;DRIVE 3
007660  177777                          .WORD   -1       ;DRIVE 4
007662  177777                          .WORD   -1       ;DRIVE 5
007664  177777                          .WORD   -1       ;DRIVE 6
007666  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

007670  177777                  DTUW:   .WORD   -1

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

007672  001                     ATABIT: .BYTE   1        ;DRIVE 0
007673  002                             .BYTE   2        ;DRIVE 1
007674  004                             .BYTE   4        ;DRIVE 2
007675  010                             .BYTE   10       ;DRIVE 3
007676  020                             .BYTE   20       ;DRIVE 4
007677  040                             .BYTE   40       ;DRIVE 5
007700  100                             .BYTE   100      ;DRIVE 6
007701  200                             .BYTE   200      ;DRIVE 7

                                ;RP04/5/6 TO RH11 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
                                ;CALLING IT FATAL (MCPEMX=1 WORD)

007702  000003                  MCPEMX: .WORD   3

                                ;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RP04/5/6),
                                ;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).

007704  176700                  RPADR:  .WORD   176700
007706  000254  000240          RPVEC:  .WORD   254.5*32.

                                ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
```

```
2094
2095  007712  000004                  MXLACT: .WORD   4
2096                                  ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
2097
2098  007714  001000                  MXDLTA: .WORD   8.*64.
2099                                  ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
2100
2101  007716  000200                  MNDLTA: .WORD   2*64.
2102                                  ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
2103
2104  007720  000005                  MXWNDW: .WORD   5
2105
2106                                  ;DEFINITIONS OF THE RH11/RP04/5/6 ADDRESS INDEXES
2107
2108          000000                  RPCS1=0                 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
2109          000002                  RPWC=2                  ;WORD COUNT REGISTER (NOT A DRIVE REG)
2110          000004                  RPBA=4                  ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
2111          000006                  RPDA=6                  ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
2112          000010                  RPCS2=10                ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
2113          000012                  RPDS1=12                ;DRIVE STATUS REGISTER (DRIVE REG 01)
2114          000014                  RPER1=14                ;ERROR REGISTER #1 (DRIVE REG. 02)
2115          000016                  RPAS=16                 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
2116          000020                  RPLA=20                 ;LOOK AHEAD REGISTER (DRIVE REG. 07)
2117          000022                  RPDB=22                 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
2118          000024                  RPMR=24                 ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
2119          000026                  RPDT=26                 ;DRIVE TYPE REGISTER (DRIVE REG. 06)
2120          000030                  RPSN=30                 ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
2121          000032                  RPOF=32                 ;OFFSET REGISTER (DRIVE REG. 11)
2122          000034                  RPCA=34                 ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
2123          000036                  RPCC=36                 ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
2124          000040                  RPER2=40                ;ERROR REGISTER #2 (DRIVE REG. 14)
2125          000042                  RPER3=42                ;ERROR REGISTER #3 (DRIVE REG. 15)
2126          000044                  RPEC1=44                ;ECC POSITION REGISTER (DRIVE REG. 15)
2127          000046                  RPEC2=46                ;ECC PATTERN REGISTER (DRIVE REG.  17)

                                      ;RH11/RP04/5/6 DRIVER INITIALIZATION CODE
                                              ;THIS ROUTINE WILL DETERMINE WHICH RP04/5/6 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,RPINIT
                                      ;       RETURN
                                      ;
                                      ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
                                      ;
2144  007722  104412                  RPINIT: SAVREG                  ;SAVE R0 - R5
2145  007724  013746  177776                  MOV     @#PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
2146  007730  012737  000240  177776          MOV     #(5*32.),@#PS   ;CHANGE THE PRIORITY TO 5
2147  007736  004737  015662                  JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
2148  007742  012701  007536                  MOV     #RPERRS,R1      ;FIRST ADDRESS TO BE CLEARED
2149  007746  012702  007646                  MOV     #SEEKFG,R2      ;LAST ADDRESS TO BE CLEARED
```

```
                                1$:     CLR     (R1)+           ;CLEAR
                                        CMP     R1,R2           ;ARE WE DONE?
                                        BLOS    1$              ;BRANCH IF NO
                                        MOV     #DTUW,R2        ;LAST ADDRESS
                                2$:     MOV     #-1,(R1)+       ;INITIALIZE
                                        CMP     R1,R2           ;DONE?
                                        BLOS    2$              ;LOOP IF NO
                                        CLR     DRVSTA          ;SET ALL DRIVES TO OFFLINE
                                        CLR     DRVSTA+2
                                        CLR     DRVSTA+4
                                        CLR     DRVSTA+6
                                        MOV     RPVEC,R3        ;SETUP THE RH11/RP04/5/6 VECTOR
                                        MOV     #ISR,(R3)+
                                        MOV     RPVEC+2,(R3)
                                        MOV     RPADR,R4        ;FIRST ADDRESS OF RH11/RP04
                                        MOV     #GITOS,RPCS2(R4)  ;MASSBUS INIT
                                        CLR     R1              ;START WITH DRIVE 0
                                3$:     JSR     R0,DRVINT       ;INIT THE DRIVE
                                        BR      4$              ;'DVA' NOT SET OR PARITY ERROR
                                        BR      5$              ;NORMAL RETURN
                                4$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
                                5$:     INC     R1              ;GO TO NEXT DRIVE
                                        BIC     #^C7,R1         ;MASK OUT UNUSED BITS
                                        BNE     3$              ;BR IF MORE DRIVES TO GO
                                        MOV     #7,R1           ;START WITH DRIVE 7
                                        CLR     @#PS            ;CLEAR THE PROCESSOR STATUS
                                6$:     TSTB    DPINT(R1)       ;WAITING FOR DRIVE TO SWITCH PORTS ?
                                        BEQ     9$              ;BR NOT WAITING
                                        JSR     PC,SET.IE       ;SET INTERRUPT
                                7$:     TSTB    DPINT(R1)       ;DRIVE SWITCHED PORTS ?
                                        BNE     7$              ;BR IF NOT
                                8$:     DEC     R1              ;GO TO THE NEXT DRIVE
                                        BPL     6$              ;CHECK NEXT DRIVE
                                        MOV     (SP)+,@#PS      ;RESTORE THE PROCESSOR STATUS
                                        RESREG                  ;RESTORE R0 - R5
                                        RTS     PC              ;BYE-BYE

                                ;DRIVE INITILIZATION ROUTINE
                                        ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
                                        ;AN RP04/5/6. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
                                        ;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     RPADR,R4        ;UNIBUS ADDRESS OF RH11 RP04/5 & (RPCS1)
                                ;       JSR     R0,DRVINT       ;CALLED BY A JSR
                                ;       RETURN1                 ;ERROR OCCURRED (PARITY)
                                ;       RETURN2                 ;NORMAL RETURN
                                ;

010134  010546                  DRVINT: MOV     R5,-(SP)        ;SAVE R5
010136  105061  007556                  CLRB    DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
010142  105061  007566                  CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
```

```
 2202  010146  105061  007624                    CLRB    ULDFLG(R1)          ;CLEAR THE UNLOAD FLAG
 2203  010152  010164  000010                    MOV     R1,RPCS2(R4)        ;SELECT A DRIVE
 2204  010156  112764  000111  000000            MOVB    #111,RPCS1(R4)      ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
 2205  010164  032764  010000  000010            BIT     #BIT12,RPCS2(R4)    ;NONEXISTENT DRIVE?
 2206  010172  001403                            BEQ     1$                  ;NO---BRANCH
 2207  010174  004737  015316                    JSR     PC,SET.IE           ;GO SET "IE" WITHOUT A "TRE"
 2208  010200  000520                            BR      6$                  ;LEAVE THIS ROUTINE
 2209  010202  105061  007556          1$:       CLRB    DRVSTA(R1)          ;SET DRIVE STATUS TO OFFLINE
 2210  010206  032764  004000  000000            BIT     #BIT11,RPCS1(R4)    ;SEE IF DRIVE AVAILABLE
 2211  010214  001514                            BEQ     7$                  ;BR IF DRIVE NOT AVAILABLE
 2212  010216  004037  014636                    JSR     R0,RD.RP            ;READ THE DRIVE TYPE REG.
 2213  010222  000026                            RPDT
 2214  010224  010466                            8$                          ;ERROR RETURN ADDRESS
 2215  010226  012605                            MOV     (SP)+,R5            ;PUT DRIVE TYPE IN R5
 2216  010230  112761  000001  007566            MOVB    #1,DRVTYP(R1)       ;SET RP04 INDICATOR
 2217  010236  022705  020020                    CMP     #20020,R5           ;IS IT A SINGLE PORT RP04?
 2218  010242  001431                            BEQ     2$                  ;BRANCH IF YES
 2219  010244  022705  024020                    CMP     #24020,R5           ;IS IT A DUAL PORT RP04?
 2220  010250  001426                            BEQ     2$                  ;BR IF YES
 2221  010252  112761  000002  007566            MOVB    #2,DRVTYP(R1)       ;SET RP05 INDICATOR
 2222  010260  022705  020021                    CMP     #20021,R5           ;SINGLE PORT RP05 ?
 2223  010264  001420                            BEQ     2$                  ;BR IF YES
 2224  010266  022705  024021                    CMP     #24021,R5           ;DUAL PORT RP05 ?
 2225  010272  001415                            BEQ     2$                  ;BR IF YES
 2226  010274  112761  000004  007566            MOVB    #4,DRVTYP(R1)       ;SET RP06 INDICATOR
 2227  010302  022705  020022                    CMP     #20022,R5           ;SINGLE PORT RP06 ?
 2228  010306  001407                            BEQ     2$                  ;BR IF YES
 2229  010310  022705  024022                    CMP     #24022,R5           ;DUAL PORT RP06 ?
 2230  010314  001404                            BEQ     2$                  ;BR IF YES
 2231  010316  112761  177777  007566            MOVB    #-1,DRVTYP(R1)      ;SET INDICATOR TO 'OTHER'
 2232  010324  000446                            BR      6$                  ;EXIT
 2233  010326  012746  000121          2$:       MOV     #121,-(SP)          ;DO A "READ-IN PRESET"
 2234  010332  004037  015012                    JSR     R0,WRT.RP
 2235  010336  000000                            RPCS1
 2236  010340  010466                            8$
 2237  010342  012746  010000                    MOV     #BIT12,-(SP)        ;SET FMT22=1
 2238  010346  004037  015012                    JSR     R0,WRT.RP
 2239  010352  000032                            RPOF
 2240  010354  010466                            8$
 2241  010356  004037  014636                    JSR     R0,RD.RP            ;READ RPDS1
 2242  010362  000012                            RPDS1
 2243  010364  010466                            8$
 2244  010366  012605                            MOV     (SP)+,R5            ;AND SAVE IT IN R5
 2245  010370  100015                            BPL     4$                  ;BRANCH IF ATA=0
 2246  010372  116164  007672  000016            MOVB    ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
 2247  010400  004037  014636                    JSR     R0,RD.RP            ;FIND OUT WHY ATA=1
 2248  010404  000014                            RPER1
 2249  010406  010466                            8$
 2250  010410  006126                            ROL     (SP)+               ;IS IT UNSAFE?
 2251  010412  100004                            BPL     4$                  ;BR IF NOT
 2252  010414  112761  177777  007556            MOVB    #-1,DRVSTA(R1)      ;SET UNSAFE INDICATOR
 2253  010422  000407                            BR      6$                  ;EXIT
 2254  010424  005105                    4$:     COM     R5                  ;CHECK MOL, DPR, DRY, AND VV
 2255  010426  042705  167077                    BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
```

```
2256  010432  001003                        BNE     6$                      ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
2257  010434  112761  000001  007556        MOVB    #1,DRVSTA(R1)           ;SET DRIVE STATUS TO ONLINE
2258  010442  005720                6$:     TST     (R0)+                   ;STEP OVER THE ERROR RETURN
2259  010444  000410                        BR      8$                      ;EXIT
2260  010446  006301                7$:     ASL     R1                      ;CHANGE INDEX TO ADDRESS WORDS
2261  010450  012761  003720  007650        MOV     #2000.,TIMER(R1)        ;START 2 SEC TIMER
2262  010456  006201                        ASR     R1                      ;RESTORE R1
2263  010460  112761  177777  007576        MOVB    #-1,DPINT(R1)           ;SET PORT INITIALIZE INIDICATOR
2264  010466  012605                8$:     MOV     (SP)+,R5                ;RESTORE R5
2265  010470  000200                        RTS     R0                      ;EXIT
2266
2267                                  ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
2268                                  ;
2269                                  ;CALL
2270                                  ;
2271                                  ;       JSR     R0,@#RP04               ;CALL THE RP04/5/6 DRIVER
2272                                  ;       PNTADR                          ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
2273                                  ;       RETURN1                         ;RETURN HERE IF QUEUE IS FULL
2274                                  ;       RETURN2                         ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
2275                                  ;                                       ;IS AN ERROR CONDITION
2276
2277  010472  013746  177776        RP04:   MOV     @#PS,-(SP)              ;SAVE THE CALLING STATUS
2278  010476  013737  007710  177776        MOV     RPVEC+2,@#PS            ;DON'T ALLOW ANY RP04/5/6 INTERRUPTS
2279  010504  112737  000001  007622        MOVB    #1,ACTDRV               ;SET "ACTIVE DRIVER" FLAG
2280  010512  104412                        SAVREG                          ;SAVE R0 - R5
2281  010514  011002                        MOV     (R0),R2                 ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
2282  010516  005062  000016                CLR     16(R2)                  ;CLEAR THE STATUS/ERROR INDICATOR
2283  010522  111201                        MOVB    (R2),R1                 ;PICKUP THE DRIVE NUMBER
2284  010524  013704  007704                MOV     RPADR,R4                ;UNIBUS ADDRESS OF RPCS1
2285  010530  105761  007556                TSTB    DRVSTA(R1)              ;CHECK DRIVES STATUS
2286  010534  003014                        BGT     1$                      ;BRANCH IF ONLINE
2287  010536  105761  007624                TSTB    ULDFLG(R1)              ;UNLOAD COMMAND IN QUEUE?
2288  010542  001036                        BNE     3$                      ;BRANCH IF YES
2289  010544  105761  007576                TSTB    DPINT(R1)               ;TRYING TO INIT THE DRIVE
2290  010550  001042                        BNE     5$                      ;BR IF YES
2291  010552  004037  010134                JSR     R0,DRVINT               ;GO INIT. THE DRIVE
2292  010556  000434                        BR      4$                      ;ERROR RETURN
2293  010560  105761  007556                TSTB    DRVSTA(R1)              ;IS DRIVE STATUS ONLINE?
2294  010564  003445                        BLE     6$                      ;BR IF NOT
2295  010566  105761  007606        1$:     TSTB    DPRQS(R1)               ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
2296  010572  001031                        BNE     5$                      ;BR IF YES
2297  010574  010164  000010                MOV     R1,RPCS2(R4)            ;SELECT THE DRIVE
2298  010600  004037  015760                JSR     R0,DRVQUE               ;PUT THIS REQUEST IN QUEUE
2299  010604  000460                        BR      9$                      ;QUEUE IS FULL
2300  010606  122762  000103  000002        CMPB    #103,2(R2)              ;IS THIS REQ. FOR AN UNLOAD?
2301  010614  001003                        BNE     2$                      ;BR IF NO
2302  010616  112761  177777  007624        MOVB    #-1,ULDFLG(R1)          ;SET THE "UNLOAD IN QUEUE" FLAG
2303  010624  105761  007546        2$:     TSTB    DRVACT(R1)              ;IS THIS DRIVE ACTIVE?
2304  010630  001043                        BNE     8$                      ;BR IF YES
2305  010632  004737  010764                JSR     PC,OPT                  ;CALL THE OPTIMIZER
2306  010636  000440                        BR      8$
2307  010640  012762  120000  000016  3$:   MOV     #BIT15!BIT13,16(R2)     ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
2308  010646  000434                        BR      8$                      ;EXIT
2309  010650  004737  012074        4$:     JSR     PC,CI7                  ;GO HANDLE THE PARITY ERROR
```

```
2310  010654  000431                          BR      8$
2311  010656  004037  015760          5$:     JSR     R0,DRVQUE       ;PUT REQUEST IN QUEUE
2312  010662  000431                          BR      9$              ;QUEUE IS FULL
2313  010664  032714  000100                  BIT     #BIT06,(R4)     ;IS 'IE' SET ALREADY ?
2314  010670  001023                          BNE     8$              ;BR IF IT IS
2315  010672  004737  015316                  JSR     PC,SET.IE       ;SET INTERRUPT
2316  010676  000420                          BR      6$              ;RETURN, REQUEST IN QUEUE
2317  010700  105761  007556          6$:     TSTB    DRVSTA(R1)      ;SEE IF DRIVE OFFLINE OR UNSAFE
2318  010704  002412                          BLT     7$              ;BR IF UNSAFE
2319  010706  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE ERROR INDICATOR
2320  010714  105761  007566                  TSTB    DRVTYP(R1)      ;SEE IF OFFLINE OR NONEXISTENT
2321  010720  001007                          BNE     8$              ;BR IF OFFLINE
2322  010722  012762  100002  000016          MOV     #BIT15!BIT01,16(R2)  ;REPORT DRIVE NONEXISTENT
2323  010730  000403                          BR      8$              ;GO TO EXIT
2324  010732  012762  110000  000016  7$:     MOV     #BIT15!BIT12,16(R2)  ;DRIVE IS UNSAFE
2325  010740  104413                  8$:     RESREG                  ;RESTORE R0 - R5
2326  010742  005720                          TST     (R0)+           ;SETUP FOR NORMAL RETURN
2327  010744  000401                          BR      10$             ;FINISH UP, THEN EXIT
2328  010746  104413                  9$:     RESREG                  ;RESTORE R0 - R5
2329  010750  005720                  10$:    TST     (R0)+           ;CORRECT THE RETURN ADDRESS
2330  010752  105037  007622                  CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
2331  010756  012637  177776                  MOV     (SP)+,@#PS      ;RETURN "PS" TO USER LEVEL
2332  010762  000200                          RTS     R0              ;RETURN TO CALLER
2333
2334                                  ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
2335                                  ;
2336                                  ;CALL
2337                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
2338                                  ;       JSR     PC,OPT          ;SETUP A COMMAND
2339                                  ;
2340  010764  104412                  OPT:    SAVREG                  ;SAVE R0 - R5
2341  010766  013746  177776                  MOV     @#PS,-(SP)      ;SAVE PROC. STATUS
2342  010772  146137  007672  007620          BICB    ATABIT(R1),SRCHWT       ;CLEAR "SEARCH WAIT" KEY
2343  011000  004737  016034                  JSR     PC,GETREQ       ;GET "DPB" POINTER OF REQUEST
2344  011004  005702                          TST     R2              ;IS THERE A REQUEST IN QUEUE?
2345  011006  001505                          BEQ     7$              ;NO--BRANCH TO EXIT
2346  011010  032764  010000  000012          BIT     #BIT12,RPDS1(R4)  ;IS MOL STILL SET ?
2347  011016  001404                          BEQ     9$              ;BR IF NOT
2348  011020  032764  000100  000012          BIT     #BIT6,RPDS1(R4)  ;IS VV SET ?
2349  011026  001003                          BNE     10$             ;BR IF IT IS
2350  011030  004037  010134          9$:     JSR     R0,DRVINT       ;SEE IF DRIVE STILL ONLINE ?
2351  011034  000470                          BR      6$              ;PARITY OR 'DVA' NOT SET
2352  011036  105761  007556          10$:    TSTB    DRVSTA(R1)      ;IS DRIVE ONLINE?
2353  011042  003014                          BGT     1$              ;YES--BRANCH
2354  011044  004737  016056                  JSR     PC,POPQUE       ;NO--REMOVE REQUEST FROM QUEUE
2355  011050  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE STATUS/ERROR INDICATOR
2356  011056  105761  007556                  TSTB    DRVSTA(R1)      ;IS DRIVE UNSAFE ?
2357  011062  100064                          BPL     8$              ;BR TO EXIT IF NOT
2358  011064  012762  110000  000016          MOV     #BIT15!BIT12,16(R2)  ;SET UNSAFE STATUS/ERROR INDICATOR
2359  011072  000460                          BR      8$              ;BRANCH TO EXIT
2360  011074  012746  000111          1$:     MOV     #111,-(SP)      ;LOAD COMMAND ONTO THE STACK
2361  011100  004037  015012                  JSR     R0,WRT.RP       ;LOAD THE REGISTER
2362  011104  000000                          RPCS1                   ;REGISTER INCREMENT
2363  011106  011216                          6$                      ;ERROR RETURN ADDRESS
```

```
2364  011110  032714  004000                    BIT     #BIT11,(R4)     ;DRIVE AVAILABLE ?
2365  011114  001427                            BEQ     5$              ;BR IF NOT
2366  011116  122762  000150  000002            CMPB    #150,2(R2)      ;IS THE REQUEST FOR I/O?
2367  011124  002403                            BLT     2$              ;YES--BRANCH
2368  011126  004737  011460                    JSR     PC,CI4          ;CALL THE COMMAND INITIATOR
2369  011132  000440                            BR      8$              ;BRANCH TO EXIT
2370  011134  005737  007670          2$:       TST     DTUW            ;DATA TRANSFER UNDERWAY?
2371  011140  002012                            BGE     4$              ;YES--GO START A SEARCH
2372  011142  005737  007646                    TST     SEEKFG          ;DO IMPLIED SEEKS?
2373  011146  100404                            BMI     3$              ;YES---BRANCH
2374  011150  004037  012430                    JSR     R0,LA           ;NO--DO LOOK AHEAD
2375  011154  000427                            BR      8$              ;RETURN HERE ON A PARITY ERROR
2376  011156  000403                            BR      4$              ;GO START A SEARCH
2377  011160  004737  011244          3$:       JSR     PC,CI1          ;START A DATA TRANSFER
2378  011164  000423                            BR      8$
2379  011166  004737  011352          4$:       JSR     PC,CI3          ;START A SEARCH
2380  011172  000420                            BR      8$              ;GO TO THE EXIT
2381  011174  112761  177777  007606  5$:       MOVB    #-1,DPRQS(R1)   ;SET PORT REQUEST INDICATOR
2382  011202  010103                            MOV     R1,R3           ;SET UP TO ADDRESS WORDS
2383  011204  006303                            ASL     R3              ;CONVERT TO WORD INDEX
2384  011206  012763  023420  007650            MOV     #10000.,TIMER(R3)  ;START 10 SEC TIMER
2385  011214  000402                            BR      7$              ;EXIT
2386  011216  004737  012074          6$:       JSR     PC,CI7          ;PROCESS THE PARITY ERROR
2387  011222  032714  000100          7$:       BIT     #BIT06,(R4)     ;SEE IF 'IE' ALREADY SET
2388  011226  001002                            BNE     8$              ;BR IF SET
2389  011230  004737  015316                    JSR     PC,SET.IE       ;SET "IE" WITHOUT A "TRE"
2390  011234  012637  177776          8$:       MOV     (SP)+,@#PS      ;RESTORE PROC. STATUS
2391  011240  104413                            RESREG                  ;RESTORE R0 - R5
2392  011242  000207                            RTS     PC
2393
2394                                    ;COMMAND INITIATOR
2395                                    ;
2396                                    ;CALL
2397                                    ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
2398                                    ;       MOV     #DPB,R2         ;ADDRESS OF DPB
2399                                    ;       JSR     PC,CI?          ;CI?= CI1,CI3, OR CI4
2400                                    ;                               ;WHERE:
2401                                    ;                               ;CI1=DATA TRANSFER
2402                                    ;                               ;CI2=SEARCH REQUESTED BY DATA XFER
2403                                    ;                               ;CI4=NOT DATA TRANSFER
2404
2405  011244  004737  016056          CI1:      JSR     PC,POPQUE       ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
2406  011250  010237  007616                    MOV     R2,TRNSWT       ;PUT REQ. IN TRANSFER WAIT QUEUE
2407  011254  010203                            MOV     R2,R3           ;DPB ADDRESS TO R3
2408  011256  013704  007704                    MOV     RPADR,R4        ;RPCS1 ADDRESS
2409  011262  010164  000010                    MOV     R1,RPCS2(R4)    ;SELECT DRIVE
2410  011266  062703  000004                    ADD     #4,R3           ;DESIRED WORD COUNT
2411  011272  062704  000002                    ADD     #2,R4           ;RPWC ADDRESS
2412  011276  012324                            MOV     (R3)+,(R4)+     ;LOAD WORD COUNT
2413  011300  012324                            MOV     (R3)+,(R4)+     ;LOAD BUFFER ADDRESS
2414  011302  012346                            MOV     (R3)+,-(SP)     ;LOAD SECTOR AND TRACK
2415  011304  004037  015012                    JSR     R0,WRT.RP       ;CALL THE LOAD(WRITE) ROUTINE
2416  011310  000006                            RPDA                    ;INDEX OF REGISTER TO LOAD
2417  011312  012074                            CI7                     ;ERROR RETURN ADDRESS
```

```
2418  011314  012346                        MOV     (R3)+,-(SP)     ;LOAD CYLINDER ADDRESS
2419  011316  004037  015012                JSR     R0,WRT.RP
2420  011322  000034                        RPCA
2421  011324  012074                        CI7
2422  011326  016246  000002                MOV     2(R2),-(SP)     ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
2423  011332  004037  015012                JSR     R0,WRT.RP
2424  011336  000000                        RPCS1
2425  011340  012074                        CI7
2426  011342  010137  007670                MOV     R1,DTUW         ;SET "DATA TRANSFER UNDERWAY"
2427  011346  000137  012036                JMP     CI5
2428  011352  013704  007704        CI3:    MOV     RPADR,R4        ;RPCS1 ADDRESS
2429  011356  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
2430  011362  016246  000012                MOV     12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
2431  011366  004037  015012                JSR     R0,WRT.RP
2432  011372  000034                        RPCA
2433  011374  012074                        CI7
2434  011376  116203  000010                MOVB    10(R2),R3       ;PICKUP SECTOR ADDRESS
2435  011402  163703  007720                SUB     MXWNDW,R3       ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
2436  011406  002002                        BGE     1$
2437  011410  062703  000026                ADD     #22.,R3
2438  011414  010346        1$:             MOV     R3,-(SP)        ;COMBINE THE ADJUSTED SECTOR WITH
2439  011416  116266  000011  000001        MOVB    11(R2),1(SP)    ;THE DESIRED TRACK
2440  011424  004037  015012                JSR     R0,WRT.RP       ;LOAD DESIRED TRACK & SECTOR
2441  011430  000006                        RPDA
2442  011432  012074                        CI7
2443  011434  012746  000131                MOV     #131,-(SP)      ;START A SEARCH
2444  011440  004037  015012                JSR     R0,WRT.RP
2445  011444  000000                        RPCS1
2446  011446  012074                        CI7
2447  011450  156137  007672  007620        BISB    ATABIT(R1),SRCHWT       ;SET "SEARCH WAIT" KEY
2448  011456  000567                        BR      CI5
2449  011460  013704  007704        CI4:    MOV     RPADR,R4        ;RPCS1 ADDRESS
2450  011464  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
2451  011470  116203  000002                MOVB    2(R2),R3        ;PICKUP THE REQUESTED COMMAND
2452  011474  122703  000131                CMPB    #131,R3         ;IS IT A SEARCH COMMAND?
2453  011500  001007                        BNE     1$              ;BRANCH IF NO
2454  011502  016246  000010                MOV     10(R2),-(SP)    ;LOAD DESIRED TRACK & SECTOR
2455  011506  004037  015012                JSR     R0,WRT.RP
2456  011512  000006                        RPDA
2457  011514  012074                        CI7
2458  011516  000403                        BR      2$              ;GO LOAD CYLINDER
2459  011520  122703  000105        1$:     CMPB    #105,R3         ;IS IT A SEEK COMMAND
2460  011524  001007                        BNE     3$              ;BRANCH IF NO
2461  011526  016246  000012        2$:     MOV     12(R2),-(SP)    ;LOAD DESIRED CYLINDER
2462  011532  004037  015012                JSR     R0,WRT.RP
2463  011536  000034                        RPCA
2464  011540  012074                        CI7
2465  011542  000546                        BR      CI6
2466  011544  122703  000115        3$:     CMPB    #115,R3         ;IS IT AN "OFFSET" COMMAND?
2467  011550  001013                        BNE     4$              ;BR IF NO
2468  011552  004037  014636                JSR     R0,RD.RP        ;MERGE THE OFFSET VALUE INTO RPOF
2469  011556  000032                        RPOF                    ;BUT DON'T CHANGE THE UPPER
2470  011560  012074                        CI7
2471  011562  116216  000001                MOVB    1(R2),(SP)      ;BYTE WHEN LOADING THE
```

```
2472  011566  004037  015012                JSR     R0,WRT.RP       ;REGISTER (RPOF)
2473  011572  000032                        RPOF
2474  011574  012074                        CI7
2475  011576  000530                        BR      CI6             ;GO START THE COMMAND
2476  011600  122703  000107        4$:     CMPB    #107,R3         ;IS IT A "RECALIBRATE" COMMAND?
2477  011604  001525                        BEQ     CI6             ;BRANCH IF YES
2478  011606  122703  000117                CMPB    #117,R3         ;IS IT A RETURN TO CENTER?
2479  011612  001522                        BEQ     CI6             ;BRANCH IF YES
2480  011614  122703  000103                CMPB    #103,R3         ;IS IT AN "UNLOAD" COMMAND?
2481  011620  001016                        BNE     5$              ;BRANCH IF NO
2482  011622  112761  000001  007546        MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE INDICATOR
2483  011630  105061  007556                CLRB    DRVSTA(R1)      ;PUT DRIVE STATUS TO OFFLINE
2484  011634  112761  000001  007624        MOVB    #1,ULDFLG(R1)   ;SET "UNLOAD IN PROGRESS" FLAG
2485  011642  010346                        MOV     R3,-(SP)        ;START THE "UNLOAD" COMMAND
2486  011644  004037  015012                JSR     R0,WRT.RP
2487  011650  000000                        RPCS1
2488  011652  012074                        CI7
2489  011654  000207                        RTS     PC              ;RETURN TO USER
2490  011656  122703  000143        5$:     CMPB    #143,R3         ;IS IT A "SET FORMAT" COMMAND?
2491  011662  001014                        BNE     6$              ;BRANCH IF NO
2492  011664  004037  014636                JSR     R0,RD.RP        ;READ THE OFFSET REGISTER
2493  011670  000032                        RPOF
2494  011672  012074                        CI7
2495  011674  116266  000001  000001        MOVB    1(R2),1(SP)     ;COMBINE "FMT22","ECI", AND "HCI"
2496  011702  004037  015012                JSR     R0,WRT.RP       ;LOAD "FMT22", "ECI", AND/OR "HCI".
2497  011706  000032                        RPOF
2498  011710  012074                        CI7
2499  011712  000436                        BR      12$
2500  011714  122703  000141        6$:     CMPB    #141,R3         ;IS IT A "GET REGISTER" COMMAND?
2501  011720  001023                        BNE     10$             ;BRANCH IF NO
2502  011722  016203  000006        7$:     MOV     6(R2),R3        ;POINTS TO 1ST ADDRESS OF WHERE
2503                                                                ;TO PUT THE REGISTER(S)
2504  011726  116237  000010  011744        MOVB    10(R2),9$       ;INIT. THE INDEX FOR THE FIRST REG.
2505  011734  116205  000011                MOVB    11(R2),R5       ;INDEX OF LAST REG. TO MOVE
2506  011740  004037  014636        8$:     JSR     R0,RD.RP        ;READ RP04/5/6 REGISTER
2507  011744  000000                9$:     RPCS1                   ;INDEX OF REG. TO READ
2508  011746  012074                        CI7
2509  011750  012623                        MOV     (SP)+,(R3)+     ;GET THE CONTENTS OF RH11/RP04/5/6 REG.
2510  011752  023705  011744                CMP     9$,R5           ;LAST REG. BEEN READ?
2511  011756  001414                        BEQ     12$             ;GET OUT IF YES
2512  011760  062737  000002  011744        ADD     #2,9$           ;INCREASE THE INDEX BY 2
2513  011766  000764                        BR      8$              ;LOOP--MORE TO READ
2514  011770  122703  000145        10$:    CMPB    #145,R3         ;IS IT A "SELECT DRIVE" COMMAND?
2515  011774  001405                        BEQ     12$             ;BRANCH IF YES
2516  011776  010346                11$:    MOV     R3,-(SP)        ;LOAD THE COMMAND
2517  012000  004037  015012                JSR     R0,WRT.RP
2518  012004  000000                        RPCS1
2519  012006  012074                        CI7
2520  012010  004737  016056        12$:    JSR     PC,POPQUE       ;REMOVE REQ. FROM QUEUE
2521  012014  052762  000200  000016        BIS     #BIT07,16(R2)   ;SET THE "DONE" BIT
2522  012022  005737  007644                TST     SAVEFG          ;SAVE THE RH11/RP04/5/6 REGISTERS?
2523  012026  100002                        BPL     13$             ;BRANCH IF NO
2524  012030  004737  015200                JSR     PC,SVRH11       ;YES--GO SAVE THE REGISTERS
2525  012034  000207                13$:    RTS     PC              ;RETURN TO USER
```

```
2526  012036  006301                    CI5:    ASL     R1
2527  012040  012761  001750  007650            MOV     #1000.,TIMER(R1)        ;SET A ONE SECOND TIMER
2528  012046  006201                            ASR     R1
2529  012050  112761  000001  007546            MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
2530  012056  000207                            RTS     PC              ;RETURN TO THE USER
2531  012060  010346                    CI6:    MOV     R3,-(SP)        ;LOAD THE COMMAND
2532  012062  004037  015012                    JSR     R0,WRT.RP
2533  012066  000000                            RPCS1
2534  012070  012074                            CI7
2535  012072  000761                            BR      CI5
2536  012074  032764  .010000  000010  CI7:     BIT     #BIT12,RPCS2(R4)  ;DRIVE NON-EXISTENT ?
2537  012102  001034                            BNE     CI8             ;BR IF YES
2538  012104  005702                    1$:     TST     R2              ;ANYTHING IN QUEUE ?
2539  012106  001405                            BEQ     CI7B            ;BR IF NOT
2540  012110  012762  104000  000016            MOV     #BIT15!BIT11,16(R2)     ;SET "PARITY" ERROR INDICATOR
2541  012116  004737  015200                    JSR     PC,SVRH11       ;GO SAVE THE RH11/RP04/5/6 REGISTERS
2542  012122  012746  000111            CI7B:  MOV     #111,-(SP)      ;DO A "DRIVE CLEAR"
2543  012126  004037  015012                    JSR     R0,WRT.RP
2544  012132  000000                            RPCS1
2545  012134  012174                            CI8
2546  012136  004737  015740                    JSR     PC,EMPTYQ       ;EMPTY THE QUEUE
2547  012142  105061  007624                    CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD IN QUEUE FLAG
2548  012146  105061  007546                    CLRB    DRVACT(R1)      ;DRIVE IS IDLE
2549  012152  020137  007670                    CMP     R1,DTUW         ;IF THIS DRIVE HAD AN I/O REQUEST
2550  012156  001005                            BNE     1$              ;IN PROGRESS CLEAR ALL OF THE FLAGS
2551  012160  005037  007616                    CLR     TRNSWT
2552  012164  012737  177777  007670            MOV     #-1,DTUW
2553  012172  000207                    1$:     RTS     PC
2554  012174  104412                    CI8:    SAVREG                  ;SAVE R0 - R5
2555  012176  032764  010000  000010            BIT     #BIT12,RPCS2(R4)  ;IS 'NED' SET ?
2556  012204  001002                            BNE     1$              ;BR IF YES
2557  012206  005001                            CLR     R1
2558  012210  005003                            CLR     R3
2559  012212  105761  007546            1$:    TSTB    DRVACT(R1)      ;DRIVE ACTIVE?
2560  012216  001443                            BEQ     5$              ;BRANCH IF NO
2561  012220  013702  007616                    MOV     TRNSWT,R2       ;GET THE "TRANSFER WAIT" QUEUE
2562  012224  020137  007670                    CMP     R1,DTUW         ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
2563  012230  001402                            BEQ     2$              ;BRANCH IF YES
2564  012232  004737  016034                    JSR     PC,GETREQ       ;GET THE DPB POINTER
2565  012236  005702                    2$:     TST     R2              ;QUEUE ENTRY FOR DRIVE ?
2566  012240  001415                            BEQ     4$              ;BR IF NOT
2567  012242  032764  010000  000010            BIT     #BIT12,RPCS2(R4)  ;'NED' SET ?
2568  012250  001404                            BEQ     3$              ;BR IF NOT
2569  012252  012762  100002  000016            MOV     #BIT15!BIT01,16(R2)  ;SET 'DRIVE NON-EXISTENT' INDICATOR
2570  012260  000405                            BR      4$              ;CONTINUE
2571  012262  012762  102000  000016    3$:     MOV     #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
2572  012270  004737  015200                    JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
2573  012274  012763  177777  007650    4$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
2574  012302  105061  007546                    CLRB    DRVACT(R1)      ;SET "DRIVE ACTIVE" TO IDLE
2575  012306  020137  007670                    CMP     R1,DTUW         ;IS THIS DRIVE SETUP FOR A TRANSFER
2576  012312  001005                            BNE     5$              ;BR IF NOT
2577  012314  012737  177777  007670            MOV     #-1,DTUW        ;RESET THE INDICATOR
2578  012322  005037  007616                    CLR     TRNSWT          ;CLEAR THE TRANSFER QUEUE
2579  012326  105061  007624            5$:    CLRB    ULDFLG(R1)      ;CLEAR UNLOAD FLAG
```

```
2580  012332  032764  010000  000010          BIT     #BIT12,RPCS2(R4)  ;'NED' SET ?
2581  012340  001021                          BNE     6$                ;BR IF YES
2582  012342  005201                          INC     R1                ;MOVE TO THE NEXT DRIVE
2583  012344  062703  000002                  ADD     #2,R3
2584  012350  042701  177770                  BIC     #^C7,R1
2585  012354  001316                          BNE     1$                ;BRANCH IF MORE DRIVES
2586  012356  012737  177777  007670          MOV     #-1,DTUW          ;NO DATA TRANSFERS UNDERWAY
2587  012364  005037  007616                  CLR     TRNSWT            ;CLEAR THE 'TRANSFER WAIT' QUEUE
2588  012370  004737  015662                  JSR     PC,CLRQUE         ;CLEAR ALL OF THE REQUEST QUEUES
2589  012374  012764  000040  000010          MOV     #BIT05,RPCS2(R4)  ;DO A MASSBUS INIT.
2590  012402  000406                          BR      7$                ;CONTINUE
2591  012404  004737  015740          6$:     JSR     PC,EMPTYQ         ;CLEAR THE DRIVE'S QUEUE
2592  012410  105061  007556                  CLRB    DRVSTA(R1)        ;SET DRIVE TO OFFLINE
2593  012414  105061  007566                  CLRB    DRVTYP(R1)        ;CLEAR THE DRIVE TYPE INDICATOR
2594  012420  004737  015316          7$:     JSR     PC,SET.IE         ;SET "IE" WITHOUT "TRE"
2595  012424  104413                          RESREG                    ;RESTORE R0 - R5
2596  012426  000207                          RTS     PC                ;RETURN
2597
2598                                  ;LOOK AHEAD ROUTINE
2599                                  ;
2600                                  ;CALL
2601                                  ;       MOV     #DRVNUM,R1        ;DRIVE NUMBER
2602                                  ;       MOV     #DPB,R2           ;POINT TO DPB
2603                                  ;       JSR     R0,LA             ;GO CHECK THE WINDOW
2604                                  ;       RETURN1                   ;ERROR RETURN
2605                                  ;       RETURN2                   ;START A SEARCH
2606                                  ;       RETURN3                   ;START A DATA TRANSFER
2607
2608  012430  013704  007704          LA:     MOV     RPADR,R4          ;GET RPCS1'S ADDRESS
2609  012434  010164  000010                  MOV     R1,RPCS2(R4)      ;SELECT DRIVE
2610  012440  004037  014636                  JSR     R0,RD.RP          ;READ CURRENT CYLINDER
2611  012444  000036                          RPCC
2612  012446  012560                          4$                        ;ERROR RETURN ADDRESS
2613  012450  022662  000012                  CMP     (SP)+,12(R2)      ;IS CURRENT CYLINDER=DESIRED
2614                                                                    ;CYLINDER?
2615  012454  001037                          BNE     3$                ;EXIT IF NO
2616  012456  105261  007634                  INCB    LACNT(R1)         ;INCREMENT THE LOOK AHEAD COUNT
2617  012462  126137  007634  007712          CMPB    LACNT(R1),MXLACT  ;EXCEED MAX?
2618  012470  003026                          BGT     2$                ;BRANCH IF YES
2619  012472  116203  000010                  MOVB    10(R2),R3         ;GET DESIRED SECTOR ADDRESS AND
2620  012476  000303                          SWAB    R3                ;MULT. BY 64--ALIGN WITH
2621  012500  006203                          ASR     R3                ;LOOK AHEAD REGISTER
2622  012502  006203                          ASR     R3
2623  012504  012737  000340  177776          MOV     #340,@#PS         ;PRIORITY LEVEL "7"
2624  012512  004037  014636                  JSR     R0,RD.RP          ;READ LOOK AHEAD REGISTER
2625  012516  000020                          RPLA
2626  012520  012560                          4$
2627  012522  162603                          SUB     (SP)+,R3          ;CALCULATE THE DELTA
2628  012524  002002                          BGE     1$
2629  012526  062703  002600                  ADD     #<22.*64.>,R3     ;MAKE THE DELTA POSITIVE
2630  012532  023703  007714          1$:     CMP     MXDLTA,R3         ;CHECK THE DELTA TO SEE
2631  012536  002406                          BLT     3$                ;IF IT IS WITHIN THE
2632  012540  023703  007716                  CMP     MNDLTA,R3         ;WINDOW---IF YES, ZERO
2633  012544  002003                          BGE     3$                ;THE LOOK AHEAD COUNT
```

```
2634  012546  105061  007634          2$:     CLRB    LACNT(R1)       ;AND TAKE THE I/O EXIT
2635  012552  005720                  TST     (R0)+
2636  012554  005720          3$:     TST     (R0)+           ;ADJUST THE RETURN ADDRESS
2637  012556  000402                  BR      5$              ;EXIT
2638  012560  004737  012074    4$:     JSR     PC,CI7          ;PROCESS THE ERROR
2639  012564  000200          5$:     RTS     R0              ;RETURN
2640
2641                                  ;INTERRUPT SERVICE ROUTINE
2642
2643  012566  112737  000001  007622  ISR:    MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
2644  012574  104412                  SAVREG                  ;SAVE R0 - R5
2645  012576  013704  007704          MOV     RPADR,R4        ;ADDRESS OF RHSCS1
2646  012602  013701  007670          MOV     DTUW,R1         ;GET "DATA TRANSFER UNDERWAY" INDICATOR
2647  012606  002403                  BLT     1$              ;BRANCH IF NO DATA TRANSFER UNDERWAY
2648  012610  004737  012632          JSR     PC,TD           ;CALL TRANSFER DONE
2649  012614  000402                  BR      2$              ;EXIT
2650  012616  004737  012772    1$:     JSR     PC,SC           ;CALL SPECIAL CONDITIONS
2651  012622  104413          2$:     RESREG                  ;RESTORE R0 - R5
2652  012624  105037  007622          CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
2653  012630  000002                  RTI                     ;RETURN
2654
2655                                  ;TRANSFER DONE ROUTINE
2656
2657  012632  105061  007546  TD:     CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE INDICATOR TO IDLE
2658  012636  012737  177777  007670  MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
2659  012644  006301                  ASL     R1
2660  012646  012761  177777  007650  MOV     #-1,TIMER(R1)   ;CANCEL TIMEOUT
2661  012654  006201                  ASR     R1
2662  012656  013702  007616          MOV     TRNSWT,R2       ;GET "DPB" ADDRESS FROM THE
2663  012662  005037  007616          CLR     TRNSWT          ;TRANSFER WAIT QUEUE--CLEAR QUEUE
2664  012666  052762  000200  000016  BIS     #BIT07,16(R2)   ;SET DONE
2665  012674  010164  000010          MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
2666  012700  004037  014636          JSR     R0,RD.RP        ;TRANSFER ERROR(TRE=1)?
2667  012704  000000                  RPCS1
2668  012706  012074                  CI7
2669  012710  006126                  ROL     (SP)+
2670  012712  100413                  BMI     3$              ;BR IF YES
2671  012714  005737  007644          TST     SAVEFG          ;SAVE THE RH11/RP04/5/6 REGISTERS?
2672  012720  100002                  BPL     1$              ;BRANCH IF NO
2673  012722  004737  015200          JSR     PC,SVRH11       ;YES--SAVE THE REGISTERS
2674  012726  004737  010764    1$:     JSR     PC,OPT          ;CALL OPTIMIZER
2675  012732  000417                  BR      SC              ;CHECK OTHER DRIVES
2676  012734  012714  000113    2$:     MOV     #113,(R4)       ;RELEASE THE DRIVE
2677  012740  000414                  BR      SC              ;CHECK FOR OTHER DRIVES
2678  012742  052762  100100  000016  3$:     BIS     #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
2679  012750  004737  015740          JSR     PC,EMPTYQ       ;EMPTY THE "DRIVE'S WAIT" QUEUE
2680  012754  004737  015200          JSR     PC,SVRH11       ;SAVE THE RH11/RP04/5/6 REGISTERS
2681  012760  012714  040111          MOV     #40111,(R4)     ;ISSUE A "DRIVE CLEAR"
2682  012764  012714  000113          MOV     #113,(R4)       ;ISSUE A RELEASE TO THE DRIVE
2683  012770  000400                  BR      SC              ;CHECK FOR OTHER DRIVES
2684
2685                                  ;SPECIAL CONDITION ROUTINE
2686
2687  012772  116403  000016  SC:     MOVB    RPAS(R4),R3     ;READ "RPAS"
```

```
2688  012776  001012                    BNE     2$              ;BRANCH IF ANY 'ATA' BITS SET
2689  013000  004037  014636            JSR     R0,RD.RP        ;READ CONTROL AND STATUS REGISTER
2690  013004  000000                    RPCS1
2691  013006  012174                    CI8
2692  013010  106126                    ROLB    (SP)+           ;IS "IE"=1?
2693  013012  100403                    BMI     1$              ;YES, NO DRIVES TO CHECK
2694  013014  104001                    ERROR   1               ;REPORT AN ILLEGAL INTERRUPT
2695  013016  004737  015316            JSR     PC,SET.IE       ;SET INTERRUPT ENABLE
2696  013022  000207            1$:     RTS     PC              ;RETURN
2697  013024  005046            2$:     CLR     -(SP)           ;PROCESS ALL DRIVES THAT HAVE
2698  013026  110316                    MOVB    R3,(SP)         ;AN "ATA"=1
2699  013030  012703  000001            MOV     #1,R3
2700  013034  005001                    CLR     R1
2701  013036  030316            SC3:    BIT     R3,(SP)         ;ATA=1?
2702  013040  001005                    BNE     SC5             ;YES--BRANCH
2703  013042  005201            SC4:    INC     R1              ;MOVE TO THE NEXT DRIVE
2704  013044  106303                    ASLB    R3
2705  013046  001373                    BNE     SC3             ;BRANCH IF MORE TO CHECK?
2706  013050  005726                    TST     (SP)+           ;CLEAN OFF THE STACK
2707  013052  000207                    RTS     PC              ;RETURN TO USER
2708  013054  105761  007576    SC5:    TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
2709  013060  001402                    BEQ     1$              ;BR IF NOT
2710  013062  000137  013750            JMP     SC13            ;PROCESS THE DRIVE
2711  013066  105761  007606    1$:     TSTB    DPRQS(R1)       ;PORT REQUEST OUTSTANDING ?
2712  013072  001402                    BEQ     2$              ;BR IF NOT
2713  013074  000137  013750            JMP     SC13            ;START THE OUTSTANDING COMMAND
2714  013100  105761  007556    2$:     TSTB    DRVSTA(R1)      ;CHECK THE DRIVE STATUS
2715  013104  003025                    BGT     5$              ;BRANCH IF ONLINE
2716  013106  105761  007624            TSTB    ULDFLG(R1)      ;UNLOAD IN PROGRESS?
2717  013112  003422                    BLE     5$              ;BRANCH IF NOT
2718  013114  004737  016034            JSR     PC,GETREQ       ;GET DPB POINTER
2719  013120  004737  015200            JSR     PC,SVRH11       ;SAVE THE RH11/RP04/5/6 REGISTERS
2720  013124  004737  013700            JSR     PC,SC12         ;SAVE RPDS1, RPER1, RPER2, AND RPER3
2721                                                            ;ALSO DO A DRIVE INIT (DRVINT)
2722  013130  105761  007556            TSTB    DRVSTA(R1)      ;DID DRIVE COME ONLINE?
2723  013134  003416                    BLE     6$              ;NO---BRANCH
2724  013136  032737  040000  007536    BIT     #BIT14,RPERRS   ;WAS THERE AN ERROR?
2725  013144  001002                    BNE     3$              ;BR IF ERROR
2726  013146  000137  013610            JMP     SC11            ;NO ERROR
2727  013152  013705  007540    3$:     MOV     RPERRS+2,R5     ;YES -- PICKUP RPER1 AND
2728  013156  000476                    BR      SC6A            ;GO PROCESS THE ERROR
2729  013160  105761  007546    5$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
2730  013164  001027                    BNE     SC6             ;BR IF EITHER
2731  013166  004737  013700            JSR     PC,SC12         ;SAVE RPDS1, RPER1, RPER2, AND RPER3
2732                                                            ;ALSO DO A DRVINT
2733  013172  105761  007576    6$:     TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE ?
2734  013176  001321                    BNE     SC4             ;BR IF YES, CHECK ON MORE DRIVES
2735  013200  105761  007556            TSTB    DRVSTA(R1)      ;CHECK ON DRIVE'S STATUS
2736  013204  100412                    BMI     7$              ;BR IF UNSAFE
2737  013206  032737  020000  007544    BIT     #BIT13,RPERRS+6 ;ADDRESS PLUG CHANGED ?
2738  013214  001011                    BNE     8$              ;BR IF YES
2739  013216  012746  000113            MOV     #113,-(SP)      ;RELEASE COMMAND
2740  013222  004037  015012            JSR     R0,WRT.RP       ;WRITE THE COMMAND INTO RPCS1
2741  013226  000000                    RPCS1                   ;REGISTER INDEX
```

```
                                                SC8             ;PARITY EXIT ADDRESS
                                        7$:     MOV     (SP),R5         ;PICKUP (RPAS) BEFORE THE ERROR CALL
                                                ERROR   2               ;REPORT THE UNEXPECTED ATTENTION
                                                BR      SC4             ;GO CHECK FOR MORE ATA'S
                                        8$:
                                                ERROR   5               ;REPORT THE ADDRESS PLUG CHANGE
                                                BR      SC4             ;CHECK FOR MORE DRIVES
                                        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,RPCS2(R4)    ;SELECT DRIVE
                                                JSR     R0,RD.RP        ;READ THE RP04'S STATUS REG.
                                                RPDS1
                                                SC8
                                                MOV     (SP),R5         ;AND PUT IT IN R5
                                                ROL     (SP)+           ;WAS THERE AN ERROR?
                                                BMI     1$              ;BR IF ERROR
                                                TSTB    DRVACT(R1)      ;CHECK DRIVE'S STATE
                                                BGT     SC11            ;BR IF DRIVE ACTIVE WITH ORDER
                                                BIS     #BIT15!BIT07!BIT03,16(R2)  ;INFORM USER OF ERROR RECOVER COMPLETION
                                                BR      SC7
                                        1$:     JSR     R0,RD.RP        ;READ ERROR REGISTER #1
                                                RPER1
                                                SC8
                                                MOV     (SP)+,R5        ;AND SAVE IT IN R5
                                                JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
                                                MOV     #111,-(SP)      ;ISSUE A DRIVE CLEAR
                                                JSR     R0,WRT.RP
                                                RPCS1
                                                SC8
                                        SC6A:   ROL     R5              ;WAS "UNSAFE" CONDITION =1?
                                                BMI     1$              ;BRANCH IF YES
                                                TST     R2              ;ANYTHING IN QUEUE ?
                                                BEQ     SC7             ;BR IF NOT
                                                BIS     #BIT15!BIT07!BIT05,16(R2)       ;INFORM USER OF ERROR
                                                BR      SC7
                                        1$:     JSR     R0,RD.RP        ;READ DRIVE STATUS REG. #1
                                                RPDS1
                                                SC8
                                                MOV     (SP),R5         ;SAVE RPDS1 IN R5
                                                ROL     (SP)+           ;"ERR"=1?
                                                BPL     2$              ;BR IF NO--UNSAFE CLEARED
                                                MOVB    #-1,DRVSTA(R1)  ;DRIVE IS UNSAFE
                                                JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
                                                BIS     #BIT15!BIT12,16(R2)     ;INFORM USER OF UNSAFE ERROR
                                                BR      SC7
                                        2$:     BIT     #BIT12,R5       ;"MOL" = 1 ?
                                                BNE     3$              ;BR IF YES
                                                MOVB    #-1,DRVACT(R1)  ;ACTIVE ERROR RECOVER
                                                MOVB    #1,DRVSTA(R1)   ;ONLINE
                                                ASL     R1
                                                MOV     #30000.,TIMER(R1) ;START 30 SECOND TIMER
                                                ASR     R1
```

```
2796  013470  000137  013042                  JMP     SC4
2797  013474  052762  100220  000016  3$:     BIS     #BIT15!BIT07!BIT04,16(R2)       ;INFORM USER OF ERROR
2798  013502  105061  007546          SC7:    CLRB    DRVACT(R1)              ;DRIVE IS IDLE
2799  013506  004737  015740                  JSR     PC,EMPTYQ               ;DUMP THE QUEUE
2800  013512  105761  007624                  TSTB    ULDFLG(R1)              ;UNLOAD IN PROGRESS OR QUEUE?
2801  013516  003002                          BGT     1$                      ;BR IF NOT
2802  013520  105061  007624                  CLRB    ULDFLG(R1)              ;CLEAR UNLOAD FLAG
2803  013524  116164  007672  000016  1$:     MOVB    ATABIT(R1),RPAS(R4)     ;CLEAR ATTENTION BIT
2804  013532  105761  007556                  TSTB    DRVSTA(R1)              ;IS THE DRIVE UNSAFE ?
2805  013536  100406                          BMI     2$                      ;BR IF IT IS
2806  013540  012746  000113                  MOV     #113,-(SP)              ;RELEASE COMMAND
2807  013544  004037  015012                  JSR     R0,WRT.RP               ;WRITE THE COMMAND INTO RPCS1
2808  013550  000000                          RPCS1                           ;REGISTER INDEX
2809  013552  013560                          SC8                             ;PARITY EXIT ADDRESS
2810  013554  000137  013042          2$:     JMP     SC4                     ;CHECK FOR MORE DRIVES
2811  013560  105761  007546          SC8:    TSTB    DRVACT(R1)              ;IS DRIVE IDLE?
2812  013564  001405                          BEQ     1$                      ;YES--BRANCH
2813  013566  004737  016034                  JSR     PC,GETREQ               ;GET DPB POINTER
2814  013572  004737  012074                  JSR     PC,CI7                  ;PROCESS THE PARITY ERROR
2815  013576  000402                          BR      2$                      ;CONTINUE
2816  013600  004737  012122          1$:     JSR     PC,CI7B                 ;PROCESS THE UNCORRECTABLE PARITY ERROR
2817  013604  000137  013042          2$:     JMP     SC4                     ;CHECK MORE DRIVES
2818  013610  105761  007624          SC11:   TSTB    ULDFLG(R1)              ;"UNLOAD IN PROGRESS"?
2819  013614  003402                          BLE     1$                      ;BRANCH IF NO
2820  013616  105061  007624                  CLRB    ULDFLG(R1)              ;CLEAR UNLOAD FLAG
2821  013622  105061  007546          1$:     CLRB    DRVACT(R1)              ;SET DRIVE IDLE
2822  013626  136137  007672  007620          BITB    ATABIT(R1),SRCHWT       ;DOING A SEARCH OPERATION FOR
2823                                                                          ;AN I/O COMMAND?
2824  013634  001012                          BNE     2$                      ;BRANCH IF YES
2825  013636  004737  016056                  JSR     PC,POPQUE               ;REMOVE REQUEST FROM QUEUE
2826  013642  052762  000200  000016          BIS     #BIT07,16(R2)           ;SET "DONE" BIT
2827  013650  005737  007644                  TST     SAVEFG                  ;SAVE THE REGISTERS?
2828  013654  100002                          BPL     2$                      ;BRANCH IF NO
2829  013656  004737  015200                  JSR     PC,SVRH11               ;YES--SAVE ALL OF THE RH11/RP04/5/6 REG'S
2830  013662  116164  007672  000016  2$:     MOVB    ATABIT(R1),RPAS(R4)     ;CLEAR ATTENTION BIT
2831  013670  004737  010764                  JSR     PC,OPT                  ;START A REQUEST
2832  013674  000137  013042                  JMP     SC4                     ;CHECK FOR MORE DRIVES
2833  013700  010164  000010          SC12:   MOV     R1,RPCS2(R4)            ;SELECT DRIVE
2834  013704  016437  000012  007536          MOV     RPDS1(R4),RPERRS        ;SAVE THE FOUR REGISTERS THAT
2835  013712  016437  000014  007540          MOV     RPER1(R4),RPERRS+2      ;WILL TELL US SOMETHING
2836  013720  016437  000040  007542          MOV     RPER2(R4),RPERRS+4
2837  013726  016437  000042  007544          MOV     RPER3(R4),RPERRS+6
2838  013734  004037  010134                  JSR     R0,DRVINT               ;INIT. THE STATE OF THE DRIVE
2839  013740  000401                          BR      1$                      ;TAKE ERROR EXIT
2840  013742  000207                          RTS     PC                      ;RETURN
2841  013744  005726                  1$:     TST     (SP)+                   ;POP PC OFF OF THE STACK
2842  013746  000704                          BR      SC8                     ;PROCESS THE PARITY ERROR
2843  013750  006301                  SC13:   ASL     R1                      ;SETUP TO ADDRESS WORDS
2844  013752  012761  177777  007650          MOV     #-1,TIMER(R1)           ;STOP THE TIMER
2845  013760  006201                          ASR     R1                      ;
2846  013762  010164  000010                  MOV     R1,RPCS2(R4)            ;SELECT THE DRIVE
2847  013766  116164  007672  000016          MOVB    ATABIT(R1),RPAS(R4)     ;CLEAR THE ATTENTION BIT
2848  013774  032714  004000                  BIT     #BIT11,(R4)             ;DRIVE AVAILABLE ?
2849  014000  001006                          BNE     1$                      ;BR IF AVAILABLE
```

```
                                        ASL     R1
                                        MOV     #10000.,TIMER(R1)  ;START 10 SEC 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,SVRH11          ;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

                                ;RP04/5/6 TIMER ROUTINE
                                ;CALL
                                ;       MOV     #TIME,-(SP)        ;ELASPED TIME IN MILLISECONDS ON THE STACK
                                ;       JSR     PC,RPTMR           ;CALL RP04/5/6 TIME ROUTINE

                                RPTMR:  TST     ACTDRV             ;CHECK "ACTDRV & ACTSTR"
                                        BNE     4$                 ;IF NON ZERO EXIT
                                        MOVB    #1,ACTSTR          ;SET "ACTSTR"
                                        SAVREG                     ;SAVE R0 - R5
                                        CLR     R1                 ;START WITH DRIVE 0
                                        CLR     R3
                                1$:     TST     TIMER(R3)          ;IS THE TIMER RUNNING?
                                        BLT     2$                 ;BRANCH IF NO
                                        SUB     2(SP),TIMER(R3)    ;COUNT THE INTERVAL
                                        BGT     2$                 ;BR IF NO SOFTWARE TIMEOUT
                                        JSR     PC,STO             ;CALL SOFTWARE TIMEOUT ROUTINE
                                        BR      3$                 ;GO TO THE EXIT
                                2$:     INC     R1                 ;MOVE TO NEXT DRIVE
                                        TST     (R3)+
                                        CMP     #8.,R1             ;OUT OF DRIVES?
                                        BGT     1$                 ;BRANCH IF NO
                                3$:     RESREG                     ;RESTORE R0 - R5
                                        CLRB    ACTSTR             ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
                                4$:     MOV     (SP)+,(SP)         ;ADJUST THE STACK
                                        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
                                        STO:    MOV     R1,-(SP)        ;SAVE R1
                                                MOV     R3,-(SP)        ;SAVE R3
                                                MOV     RPADR,R4        ;GET ADDRESS OF "RPCS1"
                                                MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
                                                JSR     R0,RD.RP        ;READ "DRIVE STATUS REG"
                                                RPDS1
                                                STO5
                                                TSTB    (SP)+           ;IS "DRY"=1?
                                                BMI     STO2            ;BR IF YES
                                        STO1:   TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
                                                BNE     STO2            ;BR IF YES
                                                TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
                                                BNE     STO2            ;BR IF YES
                                                MOV     TRNSWT,R2       ;PICKUP TRANSFER WAIT QUEUE
                                                CMP     R1,DTUW         ;TRANSFER UNDERWAY ON THIS DRIVE?
                                                BEQ     1$              ;BRANCH IF YES
                                                JSR     PC,GETREQ       ;GET DPB ADDRESS
                                        1$:     BIS     #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
                                                JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
                                                MOV     #BIT05,RPCS2(R4) ;"INIT" THE MASS BUS
                                                CLRB    DRVACT(R1)      ;DRIVE IS IDLE
                                                CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
                                                CLR     R1              ;START WITH DRIVE 0
                                                CLR     R3
                                        2$:     JSR     R0,DRVINT       ;INIT. THIS DRIVE
                                                BR      STO5            ;PARITY ERROR RETURN
                                                TSTB    DRVACT(R1)      ;DRIVE IDLE BEFORE THE INIT.?
                                                BEQ     4$              ;YES--BRANCH
                                                MOV     TRNSWT,R2       ;GET TRANSFER WAIT QUEUE
                                                CMP     DTUW,R1         ;WAS THERE I/O ON THIS DRIVE?
                                                BEQ     3$              ;YES--BRANCH
                                                JSR     PC,GETREQ       ;GET THE DPB POINTER FROM QUEUE
                                        3$:     BIS     #BIT15!BIT08,16(R2) ;INFORM USER OF INIT.
                                                CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE TO IDLE
                                        4$:     CLRB    ULDFLG(R1)      ;NO UNLOAD
                                                MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                                TST     (R3)+           ;UPDATE THE INDEX
                                                INC     R1              ;INCREMENT THE DRIVE NUMBER
                                                CMP     #8.,R1          ;LAST DRIVE BEEN CHECKED?
                                                BGT     2$              ;NO--LOOP
                                                MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
                                                CLR     TRNSWT          ;CLEAR TRANSFER WAIT QUEUE
                                                JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
                                                BR      STO9            ;EXIT
                                        STO2:   MOVB    RPAS(R4),R5     ;READ ATTENTION REG
                                                BITB    ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
                                                BNE     STO3            ;YES--BRANCH
                                                TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
                                                BNE     STO6            ;BR IF YES - DRIVE NOT ONLINE
                                                TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
```

```
                014454  001045                          BNE     ST07            ;BR IF YES - NO RESPONSE TO REQUEST
                014456  020137  007670                  CMP     R1,DTUW         ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
                014462  001263                          BNE     ST01            ;BR IF NO
                014464  004037  014636                  JSR     R0,RD.RP        ;YES--CHECK "RDY"
                014470  000000                          RPCS1
                014472  014524                          ST05
                014474  105726                          TSTB    (SP)+
                014476  100255                          BPL     ST01            ;BR IF "RDY"=0
                014500  105761  007576          ST03:   TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
                014504  001003                          BNE     1$              ;BR IF INIT PENDING
                014506  105761  007606                  TSTB    DPRQS(R1)       ;PORT REQUEST PENDING ?
                014512  001446                          BEQ     ST09            ;BR IF NOT
                014514  012763  177777  007650  1$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                014522  000442                          BR      ST09            ;EXIT
                014524  004737  012174          ST05:   JSR     PC,CI8          ;GO HANDLE THE PARITY ERROR
                014530  000437                          BR      ST09
                014532  105061  007576          ST06:   CLRB    DPINT(R1)       ;CLEAR THE INITIALIZE INDICATOR
                014536  105061  007556                  CLRB    DRVSTA(R1)      ;SET UNIT OFFLINE
                014542  012763  177777  007650          MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                014550  004737  016034                  JSR     PC,GETREQ       ;GET THE DPB ADDRESS
                014554  005702                          TST     R2              ;REQUEST IN QUEUE ?
                014556  001424                          BEQ     ST09            ;BR IF NOT
                014560  052762  140000  000016          BIS     #BIT15!BIT14,16(R2)  ;INFORM THE USER DRIVE NOT AVAILABLE
                014566  000414                          BR      ST08            ;FINISH
                014570  012763  177777  007650  ST07:   MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                014576  105061  007606                  CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST INDICATOR
                014602  004737  016034                  JSR     PC,GETREQ       ;GET DPB ADDRESS
                014606  005702                          TST     R2              ;QUEUE ENTRY FOR DRIVE ?
                014610  001407                          BEQ     ST09            ;BR IF NONE
                014612  012762  100004  000016          MOV     #BIT15!BIT2,16(R2)  ;INFORM USER OF PORT REQUEST ERROR
                014620  004737  015740          ST08:   JSR     PC,EMPTYQ       ;CLEAR THE QUEUE FOR THE DRIVE
                014624  004737  015200                  JSR     PC,SVRH11       ;SAVE THE REGISTERS
                014630  012603                  ST09:   MOV     (SP)+,R3        ;RESTORE R3
                014632  012601                          MOV     (SP)+,R1        ;RESTORE R1
                014634  000207                          RTS     PC              ;RETURN

                                                ;ROUTINE TO READ A RH11/RP04/5/6 REGISTER
                                                ;
                                                ;CALL
                                                ;       JSR     R0,RD.RP        ;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
                                                ;
                014636  013737  007702  015000  RD.RP:   MOV     MCPEMX,RD.RP2   ;MAX. RETRYS ALLOWED
                014644  011646                          MOV     (SP),-(SP)      ;SAVE R0 FOR RETURN
                014646  013737  007704  014662          MOV     RPADR,RD.ADR    ;FORM THE DESIRED ADDRESS
                014654  062037  014662                  ADD     (R0)+,RD.ADR    ;USING THE BASE AND THE INDEX
                014660  013727                  RD.RP1: MOV     @(PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RP04
                014662  000000                  RD.ADR: .WORD   0               ;ADDRESS IS FORMED HERE
                014664  000000                  RD.WRD: .WORD   0               ;REG. CONTENTS PUT HERE
                014666  013766  014664  000002          MOV     RD.WRD,2(SP)    ;RETURN IT TO THE USER
                014674  013746  007704                  MOV     RPADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
```

```
3012  014700  062716  000010                ADD     #RPCS2,(SP)     ;FORM THE ADDRESS OF RPCS2
3013  014704  032736  010000                BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
3014  014710  001035                        BNE     RD.RP3          ;BR IF DRIVE NON-EXISTENT
3015  014712  017746  172766                MOV     @RPADR,-(SP)    ;READ RPCS1
3016  014716  032716  020000                BIT     #BIT13,(SP)     ;DID MCPE SET?
3017  014722  001002                        BNE     1$              ;BRANCH IF YES
3018  014724  022620                        CMP     (SP)+,(R0)+     ;ADJUST FOR RETURN
3019  014726  000430                        BR      RD.RP4          ;EXIT
3020  014730                          1$:
3021  014730  104003                        ERROR   3               ;REPORT "MCPE" ERROR
3022  014732  005737  007670                TST     DTUW            ;DATA TRANSFER UNDERWAY?
3023  014736  100405                        BMI     2$              ;NO--BRANCH
3024  014740  032716  040000                BIT     #BIT14,(SP)     ;NO--"TRE"=1?
3025  014744  001402                        BEQ     2$              ;NO--BRANCH
3026  014746  005726                        TST     (SP)+           ;YES--CLEAN OFF THE STACK AND
3027  014750  000415                        BR      RD.RP3          ;TAKE THE FATAL ERROR EXIT
3028  014752  052716  040000          2$:   BIS     #BIT14,(SP)     ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
3029  014756  000316                        SWAB    (SP)            ;POSITION BEFORE WRITING
3030  014760  013737  007704  014774        MOV     RPADR,3$        ;FORM ADDRESS OF HIGH BYTE
3031  014766  005237  014774                INC     3$
3032  014772  112637                        MOVB    (SP)+,@(PC)+    ;WRITE THE HIGH BYTE OF RPCS1
3033  014774  000000                  3$:   .WORD   0               ;ADDRESS STORAGE
3034  014776  005327                        DEC     (PC)+           ;EXCEEDED MAX. RETRYS
3035  015000  000003                RD.RP2: .WORD   3
3036  015002  002326                        BGE     RD.RP1          ;BRANCH IF NO
3037  015004  011000                RD.RP3: MOV     (R0),R0         ;FATAL ERROR EXIT
3038  015006  012616                        MOV     (SP)+,(SP)
3039  015010  000200                RD.RP4: RTS     R0
3040
3041                                  ;ROUTINE TO WRITE A REGISTER
3042                                  ;
3043                                  ;CALL
3044                                  ;       MOV     DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
3045                                  ;       JSR     R0,WRT.RP       ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
3046                                  ;       INDEX                   ;INDEX OF THE REGISTER TO BE LOADED
3047                                  ;       ERRADR                  ;ADDRESS TO RETURN TO ON AN ERROR
3048                                  ;       RETURN                  ;ERROR FREE RETURN
3049
3050  015012  013737  007702  015162  WRT.RP: MOV     MCPEMX,WRT.R2   ;MAX RETRYS ALLOWED
3051  015020  016637  000002  015100          MOV     2(SP),WRT.WD    ;SAVE THE WORD TO WRITE
3052  015026  012616                        MOV     (SP)+,(SP)      ;ADJUST THE STACK
3053  015030  012037  015102                MOV     (R0)+,WRT.AD    ;GET INDEX OF REGISTER TO BE WRITTEN
3054  015034  001015                        BNE     1$              ;BRANCH IF NOT RPCS1
3055  015036  122737  000150  015100        CMPB    #150,WRT.WD     ;IS THE COMMAND FOR DATA TRANSFERS?
3056  015044  002411                        BLT     1$              ;YES--DON'T GET THE OLD A16 & A17, & PSEL
3057  015046  004037  014636                JSR     R0,RD.RP        ;NO---COMBINE A16&A17, & PSEL WITH
3058  015052  000000                        RPCS1                   ;THE COMMAND BEFORE SENDING IT TO
3059  015054  015170                        WRT.R3                  ;THE RH11/RP04
3060  015056  000316                        SWAB    (SP)
3061  015060  042716  177770                BIC     #^C7,(SP)
3062  015064  112637  015101                MOVB    (SP)+,WRT.WD+1
3063  015070  063737  007704  015102  1$:   ADD     RPADR,WRT.AD    ;FORM THE ADDRESS OF THE DISK REG.
3064  015076  012737                  WRT.R1: MOV     (PC)+,@(PC)+    ;LOAD THE DESIRED REG.
3065  015100  000000                  WRT.WD: .WORD   0               ;WORD TO WRITE GOES HERE
```

```
3066  015102  000000                    WRT.AD: .WORD   0               ;ADDRESS IS FORMED HERE
3067  015104  013746  007704                    MOV     RPADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
3068  015110  062716  000010                    ADD     #RPCS2,(SP)     ;FORM THE ADDRESS OF RPCS2
3069  015114  032736  010000                    BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
3070  015120  001023                            BNE     WRT.R3          ;BR IF DRIVE NON-EXISTENT
3071  015122  004037  014636                    JSR     R0,RD.RP        ;CHECK FOR PARITY ERROR ON WRITE
3072  015126  000014                            RPER1
3073  015130  015170                            WRT.R3
3074  015132  032726  000010                    BIT     #BIT03,(SP)+
3075  015136  001416                            BEQ     WRT.R4          ;BRANCH IF "PAR=0"
3076  015140  016037  177776  015152            MOV     -2(R0),1$       ;PICKUP THE INDEX
3077  015146  004037  014636                    JSR     R0,RD.RP        ;READ THE REG.
3078  015152  000000                    1$:     .WORD   0               ;REG. INDEX
3079  015154  015170                            WRT.R3                  ;RETURN TO THIS ADDRESS ON ERROR
3080  015156  104004                            ERROR   4               ;REPORT THE PARITY ON WRITE ERROR
3081  015160  005327                            DEC     (PC)+           ;DECREMENT THE ERROR COUNT
3082  015162  000003                    WRT.R2: .WORD   3               ;RETRY COUNTER
3083  015164  002344                            BGE     WRT.R1          ;TRY AGAIN IF NOT FINISHED
3084  015166  005726                            TST     (SP)+           ;CLEAN OFF THE STACK
3085  015170  011000                    WRT.R3: MOV     (R0),R0         ;TAKE THE "PARITY ON WRITE" ERROR EXIT
3086  015172  000401                            BR      WRT.R5          ;EXIT
3087  015174  005720                    WRT.R4: TST     (R0)+           ;ADJUST FOR ERROR FREE EXIT
3088  015176  000200                    WRT.R5: RTS     R0
3089
3090                                    ;ROUTINE TO SAVE THE RH11/RP04/5/6 REGISTERS AS PER DPB+14
3091                                    ;
3092                                    ;CALL
3093                                    ;       MOV     #DPBNUM,R2      ;DPB POINTER TO R2
3094                                    ;       JSR     PC,SVRH11       ;SAVE THE DRIVES REG'S
3095
3096  015200  104412                    SVRH11: SAVREG                  ;SAVE R0 - R5
3097  015202  005702                            TST     R2              ;QUEUE ENTRY FOR THE DRIVE ?
3098  015204  001430                            BEQ     4$              ;BR IF NONE
3099  015206  013704  007704                    MOV     RPADR,R4
3100  015212  111264  000010                    MOVB    (R2),RPCS2(R4)  ;SELECT DRIVE
3101  015216  016203  000014                    MOV     14(R2),R3       ;GET THE ERROR TABLE POINTER
3102  015222  001433                            BEQ     6$              ;EXIT IF NO ADDRESS
3103  015224  005037  015260                    CLR     3$              ;COUNTER & POINTER
3104  015230  023727  015260  000022    1$:     CMP     3$,#RPDB        ;REACHED THE BUFFER REGISTER ?
3105  015236  001006                            BNE     2$              ;BR IF NOT
3106  015240  032764  000200  000010            BIT     #BIT07,RPCS2(R4)  ;'OR' SET ?
3107  015246  001002                            BNE     2$              ;BR IF SET
3108  015250  005023                            CLR     (R3)+           ;STORE RPDB AS ZEROES
3109  015252  000405                            BR      4$              ;CONTINUE
3110  015254  004037  014636            2$:     JSR     R0,RD.RP        ;READ THE SELECTED REGISTER
3111  015260  000000                    3$:     .WORD   0               ;REGISTER INDEX
3112  015262  015306                            5$                      ;ERROR RETURN ADDRESS
3113  015264  012623                            MOV     (SP)+,(R3)+     ;STORE THE REGISTER CONTENTS
3114  015266  023727  015260  000046    4$:     CMP     3$,#RPEC2       ;REACHED THE END ?
3115  015274  001406                            BEQ     6$              ;BR IF YES
3116  015276  062737  000002  015260            ADD     #2,3$           ;INCREMENT THE REGISTER INDEX
3117  015304  000751                            BR      1$              ;CONTINUE READING THE REGISTERS
3118  015306  004737  012074            5$:     JSR     PC,CI7          ;PROCESS THE UNCORRECTABLE PARITY ERROR
3119  015312  104413                    6$:     RESREG                  ;RESTORE R0 - R5
```

```
3120  015314  000207                          RTS     PC              ;RETURN
3121
3122                                  ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
3123                                  ;CALL
3124                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
3125                                  ;       JSR     PC,SET.IE       ;SET "IE"
3126                                  ;       RETURN
3127
3128  015316  010446                  SET.IE: MOV     R4,-(SP)        ;SAVE R4
3129  015320  013704  007704                  MOV     RPADR,R4        ;PICKUP ADDRESS OF RPCS1
3130  015324  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT DRIVE
3131  015330  011446                          MOV     (R4),-(SP)      ;READ RPCS1
3132  015332  052716  040000                  BIS     #BIT14,(SP)     ;SET THE "TRE" BIT OF THE WORD READ
3133  015336  000316                          SWAB    (SP)            ;ADJUST FOR DATO
3134  015340  112714  000100                  MOVB    #BIT06,(R4)     ;SET "IE"
3135  015344  032764  010000  000010          BIT     #BIT12,RPCS2(R4) ;IS "NED"=1?
3136  015352  001002                          BNE     1$              ;YES--CLEAR "TRE"
3137  015354  005726                          TST     (SP)+           ;CLEAN OFF THE STACK
3138  015356  000402                          BR      2$
3139  015360  112664  000001          1$:     MOVB    (SP)+,1(R4)     ;CLEAR "TRE"
3140  015364  012604                  2$:     MOV     (SP)+,R4        ;RESTORE R4
3141  015366  000207                          RTS     PC              ;RETURN TO CALLER
3142
3143                                  ;QUEUE COUNT
3144  015370  000                     QCNT:   .BYTE   0               ;DRIVE 0
3145  015371  000                             .BYTE   0               ;DRIVE 1
3146  015372  000                             .BYTE   0               ;DRIVE 2
3147  015373  000                             .BYTE   0               ;DRIVE 3
3148  015374  000                             .BYTE   0               ;DRIVE 4
3149  015375  000                             .BYTE   0               ;DRIVE 5
3150  015376  000                             .BYTE   0               ;DRIVE 6
3151  015377  000                             .BYTE   0               ;DRIVE 7
3152
3153                                  ;QUEUE INPUT POINTERS
3154
3155  015400  015462                  QINPT:  .WORD   QDRV0            ;DRIVE 0
3156  015402  015502                          .WORD   QDRV1           ;DRIVE 1
3157  015404  015522                          .WORD   QDRV2           ;DRIVE 2
3158  015406  015542                          .WORD   QDRV3           ;DRIVE 3
3159  015410  015562                          .WORD   QDRV4           ;DRIVE 4
3160  015412  015602                          .WORD   QDRV5           ;DRIVE 5
3161  015414  015622                          .WORD   QDRV6           ;DRIVE 6
3162  015416  015642                          .WORD   QDRV7           ;DRIVE 7
3163
3164                                  ;QUEUE OUTPUT POINTERS
3165
3166  015420  015462                  QOUTPT: .WORD   QDRV0            ;DRIVE 0
3167  015422  015502                          .WORD   QDRV1           ;DRIVE 1
3168  015424  015522                          .WORD   QDRV2           ;DRIVE 2
3169  015426  015542                          .WORD   QDRV3           ;DRIVE 3
3170  015430  015562                          .WORD   QDRV4           ;DRIVE 4
3171  015432  015602                          .WORD   QDRV5           ;DRIVE 5
3172  015434  015622                          .WORD   QDRV6           ;DRIVE 6
3173  015436  015642                          .WORD   QDRV7           ;DRIVE 7
```

```
3174
3175  015440  015462                    QSTART: .WORD   QDRV0            ;DRIVE 0 START ADDRESS
3176  015442  015502                    QSTOP:  .WORD   QDRV1            ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
3177  015444  015522                            .WORD   QDRV2            ;STOP DRIVE 1--START DRIVE 2
3178  015446  015542                            .WORD   QDRV3            ;STOP DRIVE 2--START DRIVE 3
3179  015450  015562                            .WORD   QDRV4            ;STOP DRIVE 3--START DRIVE 4
3180  015452  015602                            .WORD   QDRV5            ;STOP DRIVE 4--START DRIVE 5
3181  015454  015622                            .WORD   QDRV6            ;STOP DRIVE 5--START DRIVE 6
3182  015456  015642                            .WORD   QDRV7            ;STOP DRIVE 6--START DRIVE 7
3183  015460  015662                            .WORD   QTERM            ;STOP DRIVE 7
3184
3185                                    ;DRIVE REQUEST QUEUES
3186
3187  015462  000010                    QDRV0:  .BLKW   10
3188  015502  000010                    QDRV1:  .BLKW   10
3189  015522  000010                    QDRV2:  .BLKW   10
3190  015542  000010                    QDRV3:  .BLKW   10
3191  015562  000010                    QDRV4:  .BLKW   10
3192  015602  000010                    QDRV5:  .BLKW   10
3193  015622  000010                    QDRV6:  .BLKW   10
3194  015642  000010                    QDRV7:  .BLKW   10
3195          015662                    QTERM=.
3196
3197                                    ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
3198                                    ;
3199                                    ;CALL
3200                                    ;       JSR     PC,CLRQUE
3201
3202  015662  104412                    CLRQUE: SAVREG                   ;SAVE R0 - R5
3203  015664  012702  015370                    MOV     #QCNT,R2         ;ZERO THE QUEUE COUNTS
3204  015670  005022                            CLR     (R2)+            ;DRIVES 0 & 1
3205  015672  005022                            CLR     (R2)+            ;DRIVES 2 & 3
3206  015674  005022                            CLR     (R2)+            ;DRIVES 4 & 5
3207  015676  005022                            CLR     (R2)+            ;DRIVES 6 & 7
3208  015700  012703  000010                    MOV     #8.,R3           ;MOVE THE STARTING
3209  015704  012701  015440                    MOV     #QSTART,R1       ;ADDRESS OF THE QUEUE INTO
3210  015710  012122                    1$:     MOV     (R1)+,(R2)+      ;THE QUEUE INPUT POINTER
3211  015712  005303                            DEC     R3
3212  015714  001375                            BNE     1$
3213  015716  012703  000010                    MOV     #8.,R3           ;MOVE THE STARTING ADDRESS
3214  015722  012701  015440                    MOV     #QSTART,R1       ;OF THE QUEUE INTO THE
3215  015726  012122                    2$:     MOV     (R1)+,(R2)+      ;QUEUE OUTPUT POINTER
3216  015730  005303                            DEC     R3
3217  015732  001375                            BNE     2$
3218  015734  104413                            RESREG                   ;RESTORE R0 - R5
3219  015736  000207                            RTS     PC
3220
3221                                    ;EMPTY THE QUEUE SPECIFIED BY R1
3222                                    ;
3223                                    ;CALL
3224                                    ;       MOV     DRVNUM,R1        ;DRIVE NUMBER TO R1
3225                                    ;       JSR     PC,EMPTYQ
3226
3227  015740  105061  015370            EMPTYQ: CLRB    QCNT(R1)         ;CLEAR NUMBER OF ITEMS IN QUEUE
```

```
3228  015744  006301                          ASL     R1
3229  015746  016161  015400  015420          MOV     QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
3230  015754  006201                          ASR     R1
3231  015756  000207                          RTS     PC
3232
3233                                  ;ROUTINE TO PUT A REQUEST IN QUEUE
3234                                  ;
3235                                  ;CALL
3236                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
3237                                  ;       MOV     #DPB,R2         ;ADDRESS OF PARAMETER BLOCK
3238                                  ;       JSR     R0,DRVQUE       ;GO PUT REQUEST IN QUEUE
3239                                  ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
3240                                  ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE
3241
3242  015760  122761  000010  015370  DRVQUE: CMPB    #10,QCNT(R1)    ;IS QUEUE FULL?
3243  015766  001421                          BEQ     2$              ;BR IF YES-TAKE RETURN1
3244  015770  105261  015370                  INCB    QCNT(R1)        ;INCREMENT QUEUE COUNT
3245  015774  006301                          ASL     R1
3246  015776  010271  015400                  MOV     R2,@QINPT(R1)   ;PUT THIS REQUEST IN QUEUE
3247  016002  062761  000002  015400          ADD     #2,QINPT(R1)    ;UPDATE THE QUEUE POINTER
3248  016010  026161  015400  015442          CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
3249  016016  001003                          BNE     1$              ;BRANCH IF NO
3250  016020  016161  015440  015400          MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
3251  016026  006201                  1$:     ASR     R1
3252  016030  005720                          TST     (R0)+           ;TAKE RETURN 2
3253  016032  000200                  2$:     RTS     R0              ;RETURN TO USER
3254
3255                                  ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
3256                                  ;
3257                                  ;CALL
3258                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
3259                                  ;       JSR     PC,GETREQ       ;GO GET THE REQUEST
3260                                  ;       RETURN                  ;R2="DPB" ADDRESS OF THE REQUEST
3261                                  ;                               ;R2=0 IF NO REQUEST IN QUEUE
3262
3263  016034  005002                  GETREQ: CLR     R2
3264  016036  105761  015370                  TSTB    QCNT(R1)        ;IS THERE ANY REQUEST IN QUEUE?
3265  016042  001404                          BEQ     2$              ;NO---BRANCH
3266  016044  006301                  1$:     ASL     R1
3267  016046  017102  015420                  MOV     @QOUTPT(R1),R2  ;PICKUP "DPB" POINTER FOR THIS DRIVE
3268  016052  006201                          ASR     R1
3269  016054  000207                  2$:     RTS     PC              ;RETURN TO USER
3270
3271                                  ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
3272                                  ;
3273                                  ;CALL
3274                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
3275                                  ;       JSR     PC,POPQUE       ;CALL TO REMOVE REQUEST
3276                                  ;       RETURN                  ;R2=ADDRESS OF DPB REMOVED
3277
3278  016056  105361  015370          POPQUE: DECB    QCNT(R1)        ;DECREMENT QUEUE COUNT
3279  016062  006301                          ASL     R1
3280  016064  017102  015420                  MOV     @QOUTPT(R1),R2  ;GET THE "DPB" POINTER
3281  016070  062761  000002  015420          ADD     #2,QOUTPT(R1)   ;UPDATE THE QUEUE POINTER
```

```
3282  016076  026161  015420  015442          CMP     QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
3283  016104  001003                          BNE     1$                   ;NO--BRANCH TO EXIT
3284  016106  016161  015440  015420          MOV     QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
3285  016114  006201                  1$:     ASR     R1
3286  016116  000207                          RTS     PC                   ;RETURN TO USER
3287
3288
3289
3290                                  ;;*****************************************************************
3291
3292                                  .SBTTL  DATA PARAMETER BLOCK
3293
3294                                  ;;*****************************************************************
3295
3296  016120     000                  DPB:    .BYTE 0                      ;DRIVE NUMBER
3297  016121     000                          .BYTE 0                      ;OFFSET VALUE OR FMT22, ECI, AND HCI
3298  016122     000                          .BYTE 0                      ;COMMAND
3299  016123     000                          .BYTE 0                      ;PSEL AND A17 AND A16
3300  016124  000000                          .WORD 0                      ;WORD COUNT (MUST BE NEG)
3301  016126  001206                          .WORD INREG                  ;BUFFER ADDRESS OR
3302                                                                       ;REGISTER TABLE POINTER
3303  016130     000                          .BYTE 0                      ;SECTOR ADDRESS OR
3304                                                                       ;FIRST REGISTER INDEX
3305  016131     000                          .BYTE 0                      ;TRACK ADDRESS OR
3306                                                                       ;LAST REGISTER INDEX
3307  016132  000000                          .WORD 0                      ;CYLINDER ADDRESS
3308  016134  016140                          .WORD REG                    ;ERROR TABLE POINTER
3309                                                                       ;POINTS TO THE FIRST OF TWENTY
3310                                                                       ;LOCATIONS WHERE THE DRIVER IS
3311                                                                       ;TO STORE THE RH11/RP04 REGISTERS
3312                                                                       ;ON AN ERROR.  IF ZERO, REGISTERS
3313                                                                       ;ARE NOT SAVED.
3314  016136  000000                          .WORD 0                      ;STATUS/ERROR INDICATOR
3315                                                                       ;BIT15 = 1 => ERROR OCCURRED
3316                                                                       ;BIT07 = 1 => DONE
3317                                                                       ;BIT 14 - BIT10 AND BIT06 - BIT03
3318                                                                       ;INDICATE TYPE OF ERROR
3319
3320  016140  000000                  REG:    .WORD   0                    ;STORE RP04 REGISTERS HERE
3321  016142  000000                          .WORD   0                    ;RPWC
3322  016144  000000                          .WORD   0                    ;RPBA
3323  016146  000000                          .WORD   0                    ;RPDA
3324  016150  000000                          .WORD   0                    ;RPCS2
3325  016152  000000                          .WORD   0                    ;RPDS1
3326  016154  000000                          .WORD   0                    ;RPER1
3327  016156  000000                          .WORD   0                    ;RPAS
3328  016160  000000                          .WORD   0                    ;RPLA
3329  016162  000000                          .WORD   0                    ;RPDB
3330  016164  000000                          .WORD   0                    ;RPMR
3331  016166  000000                          .WORD   0                    ;RPDT
3332  016170  000000                          .WORD   0                    ;RPSN
3333  016172  000000                          .WORD   0                    ;RPOF
3334  016174  000000                          .WORD   0                    ;RPCA
3335  016176  000000                          .WORD   0                    ;RPCC
```

```
3336  016200  000000                    .WORD   0                       ;RPER2
3337  016202  000000                    .WORD   0                       ;RPER3
3338  016204  000000                    .WORD   0                       ;RPEC1
3339  016206  000000                    .WORD   0                       ;RPEC2
3340
3341          000001                    CODE=1                          ;DPB INDEX EQUATES
3342          000002                    COMND=2
3343          000006                    BUF=6
3344          000010                    SEC=10
3345          000011                    TRK=11
3346          000012                    CYL=12
3347          000016                    STATUS=16
3348
3349                                ;;*****************************************************************
3350
3351                                .SBTTL  HEAD CODE TABLE
3352
3353                                ;;*****************************************************************
3354
3355  016210  000000                HEAD1:  .WORD   0                       ;HEAD ADDRESSES FOR CYLINDERS 245 & 496
3356  016212  000001                        .WORD   1
3357  016214  000002                        .WORD   2
3358  016216  000003                        .WORD   3
3359  016220  000004                        .WORD   4
3360  016222  000005                        .WORD   5
3361  016224  000006                        .WORD   6
3362  016226  000007                        .WORD   7
3363  016230  000010                        .WORD   10
3364  016232  000011                        .WORD   11
3365  016234  000012                        .WORD   12
3366  016236  000013                        .WORD   13
3367  016240  000014                        .WORD   14
3368  016242  000015                        .WORD   15
3369  016244  000016                        .WORD   16
3370  016246  000017                        .WORD   17
3371  016250  000020                        .WORD   20
3372  016252  000021                        .WORD   21
3373  016254  000022                        .WORD   22
3374  016256  100000                        .WORD   100000                  ;TABLE TERMINATOR
3375
3376  016260  000000                HEAD6:  .WORD   0                       ;HEAD ADDRESSES FOR CYLS 800 & 008
3377  016262  000001                        .WORD   1
3378  016264  000012                        .WORD   10.
3379  016266  000021                        .WORD   17.
3380  016270  000022                        .WORD   18.
3381  016272  100000                        .WORD   100000
3382  016274  000000                        .WORD   0
3383  016276  000001                        .WORD   1
3384  016300  000012                        .WORD   10.
3385  016302  000021                        .WORD   17.
3386  016304  000022                        .WORD   18.
3387  016306  100000                        .WORD   100000                  ;TERMINATOR
3388
3389  016310  000000                HEAD45: .WORD   0                       ;HEAD ADDRESSES FOR CYLINDERS 400 & 4
```

```
 3390  016312  000022                        .WORD   18.
 3391  016314  100000                        .WORD   100000
 3392  016316  000000                        .WORD   0
 3393  016320  000022                        .WORD   18.
 3394  016322  100000                        .WORD   100000                  ;TERMINATOR
 3395
 3396                                  ;;*****************************************************************
 3397
 3398                                  .SBTTL  OFFSET CODE TABLES
 3399
 3400                                  ;;*****************************************************************
 3401
 3402  016324                          OFTBL6:                                 ;OFFSET TABLE FOR CYLINDER 496
 3403  016324  030                             .BYTE   30
 3404  016325  027                             .BYTE   27
 3405  016326  026                             .BYTE   26
 3406  016327  025                             .BYTE   25
 3407  016330  024                             .BYTE   24
 3408  016331  023                             .BYTE   23
 3409  016332  022                             .BYTE   22
 3410  016333  021                             .BYTE   21
 3411  016334  020                             .BYTE   20
 3412  016335  017                             .BYTE   17
 3413  016336  016                             .BYTE   16
 3414  016337  015                             .BYTE   15
 3415  016340  014                             .BYTE   14
 3416  016341  013                             .BYTE   13
 3417  016342  012                             .BYTE   12
 3418  016343  011                             .BYTE   11
 3419  016344  010                             .BYTE   10
 3420  016345  007                             .BYTE   7
 3421  016346  006                             .BYTE   6
 3422  016347  005                             .BYTE   5
 3423  016350  004                             .BYTE   4
 3424  016351  003                             .BYTE   3
 3425  016352  002                             .BYTE   2
 3426  016353  001                             .BYTE   1
 3427  016354  000                             .BYTE   0
 3428  016355  201                             .BYTE   201
 3429  016356  202                             .BYTE   202
 3430  016357  203                             .BYTE   203
 3431  016360  204                             .BYTE   204
 3432  016361  205                             .BYTE   205
 3433  016362  206                             .BYTE   206
 3434  016363  207                             .BYTE   207
 3435  016364  210                             .BYTE   210
 3436  016365  211                             .BYTE   211
 3437  016366  212                             .BYTE   212
 3438  016367  213                             .BYTE   213
 3439  016370  214                             .BYTE   214
 3440  016371  215                             .BYTE   215
 3441  016372  216                             .BYTE   216
 3442  016373  217                             .BYTE   217
 3443  016374  220                             .BYTE   220
```

```
                                        .BYTE   221
                                        .BYTE   220
                                        .BYTE   223
                                        .BYTE   222
                                        .BYTE   225
                                        .BYTE   224
                                        .BYTE   227
                                        .BYTE   226
                                        .BYTE   230

OFTBL4:                                                 ;OFFSET TABLE FOR CYLINDER 245
                                        .BYTE   60
                                        .BYTE   61
                                        .BYTE   62
                                        .BYTE   63
                                        .BYTE   64
                                        .BYTE   65
                                        .BYTE   66
                                        .BYTE   67
                                        .BYTE   50
                                        .BYTE   51
                                        .BYTE   52
                                        .BYTE   53
                                        .BYTE   54
                                        .BYTE   55
                                        .BYTE   56
                                        .BYTE   57
                                        .BYTE   40
                                        .BYTE   41
                                        .BYTE   42
                                        .BYTE   43
                                        .BYTE   44
                                        .BYTE   45
                                        .BYTE   46
                                        .BYTE   47
                                        .BYTE   30
                                        .BYTE   31
                                        .BYTE   32
                                        .BYTE   33
                                        .BYTE   34
                                        .BYTE   35
                                        .BYTE   36
                                        .BYTE   37
                                        .BYTE   20
                                        .BYTE   21
                                        .BYTE   22
                                        .BYTE   23
                                        .BYTE   24
                                        .BYTE   25
                                        .BYTE   26
                                        .BYTE   27
                                        .BYTE   10
                                        .BYTE   11
                                        .BYTE   12
                                        .BYTE   13
                                        .BYTE   14
                                        .BYTE   15
                                        .BYTE   16
                                        .BYTE   17
                                        .BYTE   00
                                        .BYTE   01
                                        .BYTE   02
                                        .BYTE   03
                                        .BYTE   04
                                        .BYTE   05
                                        .BYTE   06
                                        .BYTE   07
```

```
003460  016461  004                     .BYTE   4
003462  016462  003                     .BYTE   3
003464  016463  002                     .BYTE   2
003466  016464  001                     .BYTE   1
003470  016465  000                     .BYTE   0
003472  016466  000001                  .BYTE   201
003474  016467  000002                  .BYTE   202
003476  016470  000003                  .BYTE   203
003500  016471  000004                  .BYTE   204
003502  016472  000005                  .BYTE   205
003504  016473  000006                  .BYTE   206
003506  016474  000007                  .BYTE   207
003510  016475  000010                  .BYTE   210
003512  016476  000011                  .BYTE   211
003514  016477  000012                  .BYTE   212
003516  016500  000013                  .BYTE   213
003520  016501  000014                  .BYTE   214
003522  016502  000015                  .BYTE   215
003524  016503  000016                  .BYTE   216
003526  016504  000017                  .BYTE   217
003530  016505  000020                  .BYTE   220
003532  016506  000021                  .BYTE   221
003534  016507  000022                  .BYTE   222
003536  016510  000023                  .BYTE   223
003540  016511  000024                  .BYTE   224
003542  016512  000025                  .BYTE   225
003544  016513  000026                  .BYTE   226
003546  016514  000027                  .BYTE   227
003550  016515  000030                  .BYTE   230
003552  016516  000031                  .BYTE   231
003554  016517  000032                  .BYTE   232
003556  016520  000033                  .BYTE   233
003560  016521  000034                  .BYTE   234
003562  016522  000035                  .BYTE   235
003564  016523  000036                  .BYTE   236
003566  016524  000037                  .BYTE   237
003570  016525  000040                  .BYTE   240
003572  016526  000041                  .BYTE   241
003574  016527  000042                  .BYTE   242
003576  016530  000043                  .BYTE   243
003600  016531  000044                  .BYTE   244
003602  016532  000045                  .BYTE   245
003604  016533  000046                  .BYTE   246
003606  016534  000047                  .BYTE   247
003610  016535  000050                  .BYTE   250
003612  016536  000051                  .BYTE   251
003614  016537  000052                  .BYTE   252
003616  016540  000053                  .BYTE   253
003620  016541  000054                  .BYTE   254
003622  016542  000055                  .BYTE   255
003624  016543  000056                  .BYTE   256
003626  016544  000057                  .BYTE   257
003630  016545  000060                  .BYTE   260
003632                                  .EVEN
```

```
                                        ;;*****************************************************************
                                        .SBTTL  MESSAGES
                                        ;;*****************************************************************
016546  005015  040515  047111  TITLE:  .ASCII  <CR><LF>/MAINDEC-11-DZRJC-A/<CR><LF>
016554  042504  026503  030461
016562  042055  051132  041512
016570  040455  005015
016574  050122  032060  032457        .ASCIZ  @RP04/5/6 HEAD ALIGNMENT VERIFICATION PROGRAM@<CR><LF><LF>
016602  033057  044040  040505
016610  020104  046101  043511
016616  046516  047105  020124
016624  042526  044522  044506
016632  040503  044524  047117
016640  050040  047522  051107
016646  046501  005015  000012

016654  047504  047040  052117  DDU:    .ASCII  /DO NOT SELECT DRIVE UNTIL HEAD ALIGNMENT TEST BOX HAS BEEN CONNECTED/<C
016662  051440  046105  041505
016670  020124  051104  053111
016676  020105  047125  044524
016704  020114  042510  042101
016712  040440  044514  047107
016720  042515  052116  052040
016726  051505  020124  047502
016734  020130  040510  020123
016742  042502  047105  041440
016750  047117  042516  052103
016756  042105  005015

016762  047040  052117  035105  .ASCII  / NOTE:  WITH THE ALIGNMENT PACK MOUNTED, CONSTANT INDEX ERRORS MAY OCCU
016770  020040  044527  044124
016776  052040  042510  040440
017004  044514  047107  042515
017012  052116  050040  041501
017020  020113  047515  047125
017026  042524  026104  041440
017034  047117  052123  047101
017042  020124  047111  042504
017050  020130  051105  047522
017056  051522  046440  054501
017064  047440  041503  051125
017072  006456      012

017075      040  044124  051505  .ASCII  / THESE INDEX ERRORS ARE CAUSED BY THE ALIGNMENT PACK AND MAY<CR><LF>
017102  020105  047111  042504
017110  020130  051105  047522
017116  051522  040440  042522
017124  041440  052501  042523
017132  020104  054502  052040
017140  042510  040440  044514
017146  047107  042515  052116
017154  050040  041501  020113
```

```
 3606  017162  047101  020104  040515
 3607  017170  006531  000012
 3608  017173  000040  047516  020124          .ASCII  / NOT OCCUR WITH SOME ALIGNMENT PACKS.  TO INHIBIT THE ALIGNMENT PACK/<C
 3609  017200  041517  052503  020122
 3610  017206  044527  044124  051440
 3611  017214  046517  020105  046101
 3612  017222  043511  046516  047105
 3613  017230  020124  040520  045503
 3614  017236  027123  020040  047524
 3615  017244  044440  044116  041111
 3616  017252  052111  052040  042510
 3617  017260  040440  044514  047107
 3618  017266  042515  052116  050040
 3619  017274  041501  006513  000012
 3620  017301  000040  042522  040514          .ASCII  / RELATED INDEX ERROR. GROUND THE FOLLOWING PINS:/<CR><LF>
 3621  017306  042524  020104  047111
 3622  017314  042504  020130  051105
 3623  017322  047522  026122  043440
 3624  017330  047522  047125  020104
 3625  017336  044124  020105  047506
 3626  017344  046114  053517  047111
 3627  017352  020107  044520  051516
 3628  017360  006472  000012
 3629  017363  000040  020040  020040          .ASCII  /      RP04: PIN A1A05C3 (FUNCTION '+TPNDEXER')/<CR><LF>
 3630  017370  051040  030120  035064
 3631  017376  050040  047111  040440
 3632  017404  040461  032460  034060
 3633  017412  024040  052506  041516
 3634  017420  044524  047117  023440
 3635  017426  052053  047120  042504
 3636  017434  042530  023522  006451
 3637  017442  000012
 3638  017443  000040  020040  020040          .ASCIZ  /      RP05/6:  PIN D04R19 (FUNCTION '-INDEX')/<CR><LF><LF>
 3639  017450  051040  030120  027465
 3640  017456  035066  020040  044520
 3641  017464  020116  030104  051064
 3642  017472  034461  024040  052506
 3643  017500  041516  044524  047117
 3644  017506  023440  044455  042116
 3645  017514  054105  024447  005015
 3646  017522  000012
 3647
 3648  017524  005015  051104  053111  NODRV:  .ASCIZ  <CR><LF>/DRIVE NOT AVAILABLE/<CR><LF><LF>
 3649  017532  020105  047516  020124
 3650  017540  053101  044501  040514
 3651  017546  046102  006505  005012
 3652  017554  000000
 3653
 3654  017555  000015  042412  052116  ENTERD: .ASCIZ  <CR><LF>/ENTER DRIVE NUMBER: /
 3655  017562  051105  042040  044522
 3656  017570  042526  047040  046525
 3657  017576  042502  035122  000040
 3658
 3659  017604  000015  051104  053111  WLOCK:  .ASCIZ  <CR><LF>/DRIVE NOT WRITE PROTECTED/<CR><LF><LF>
```

```
017612  020105  047516  020124
017620  051127  052111  020105
017626  051120  052117  041505
017634  042524  006504  005012
017642     000

017643     040  044040  040505  TOLRG:  .ASCIZ  /  HEAD OUT OF RANGE/
017650  020104  052517  020124
017656  043117  051040  047101
017664  042507     000

017667     015  020012  020040  HEADER: .ASCII  <CR><LF>/               TRK CENTER/<CR><LF>
017674  020040  020040  020040
017702  020040  020040  052040
017710  045522  041440  047105
017716  042524  006522     012
017723     040  020040  042510          .ASCIZ  /   HEAD  CYL (IN U INCHES)/<CR><LF><LF>
017730  042101  020040  054503
017736  020114  044450  020116
017744  020125  047111  044103
017752  051505  006451  005012
017760     000

017761     101  046050  043511  MODE:   .ASCIZ  /A(LIGN), V(ERIFY), OR E(XERCISE) ? /
017766  024516  020054  024126
017774  051105  043111  024531
020002  020054  051117  042440
020010  054050  051105  044503
020016  042523  020051  020077
020024     000

020025     126  051105  043111  VERIFY: .ASCIZ  /VERIFY/<CR><LF><LF>
020032  006531  005012     000

020037     101  044514  047107  ALIGN:  .ASCIZ  /ALIGN/<CR><LF><LF>
020044  005015  000012

020050  054105  051105  044503  EXER:   .ASCIZ  /EXERCISE/<CR><LF><LF>
020056  042523  005015  000012

020064  042510  042101  030040  HDZERO: .ASCIZ  /HEAD 0 SELECTED BY DEFAULT/<CR><LF><LF>
020072  051440  046105  041505
020100  042524  020104  054502
020106  042040  043105  052501
020114  052114  005015  000012

020122  054503  044514  042116  CYL245: .ASCIZ  /CYLINDER 245 SELECTED BY DEFAULT/<CR><LF><LF>
020130  051105  031040  032464
020136  051440  046105  041505
020144  042524  020104  054502
020152  042040  043105  052501
020160  052114  005015  000012

020166  054503  044514  042116  CYL496: .ASCIZ  /CYLINDER 496 SELECTED BY DEFAULT/<CR><LF><LF>
```

```
3714  020174  051105  032040  033071
3715  020202  051440  046105  041505
3716  020210  042524  020104  054502
3717  020216  042040  043105  052501
3718  020224  052114  005015  000012
3719
3720  020232  044504  045523  044440  ALN245: .ASCIZ  /DISK IS POSITIONED AT CYLINDER 245/<CR><LF><LF>
3721  020240  020123  047520  044523
3722  020246  044524  047117  042105
3723  020254  040440  020124  054503
3724  020262  044514  042116  051105
3725  020270  031040  032464  005015
3726  020276  000012
3727
3728  020300  044504  045523  044440  ALN496: .ASCIZ  /DISK IS POSITIONED AT CYLINDER 496/<CR><LF><LF>
3729  020306  020123  047520  044523
3730  020314  044524  047117  042105
3731  020322  040440  020124  054503
3732  020330  044514  042116  051105
3733  020336  032040  033071  005015
3734  020344  000012
3735
3736  020346  005015  047105  042524  ENTHD:  .ASCIZ  <CR><LF>/ENTER HEAD: /
3737  020354  020122  042510  042101
3738  020362  020072     000
3739
3740  020365     015  042412  052116  ENTCYL: .ASCIZ  <CR><LF>/ENTER CYLINDER ADDRESS: /
3741  020372  051105  041440  046131
3742  020400  047111  042504  020122
3743  020406  042101  051104  051505
3744  020414  035123  000040
3745
3746  020420  047111  040526  044514  BADTRK: .ASCIZ  /INVALID HEAD ADDRESS/<CR><LF>
3747  020426  020104  042510  042101
3748  020434  040440  042104  042522
3749  020442  051523  005015     000
3750
3751  020447     111  053116  046101  BADCYL: .ASCIZ  /INVALID CYLINDER ADDRESS/<CR><LF>
3752  020454  042111  041440  046131
3753  020462  047111  042504  020122
3754  020470  042101  051104  051505
3755  020476  006523  000012
3756
3757  020502  005015  042012  047117  ENDMSG: .ASCIZ  <CR><LF><LF>/DONE/<CR><LF><LF>
3758  020510  006505  005012     000
3759
3760
3761          020516                          .EVEN
3762
3763  020516  044122  030461  044440  EM1:    .ASCIZ  /RH11 INTERRUPT OCCURRED (RPAS=0)/
3764  020524  052116  051105  052522
3765  020532  052120  047440  041503
3766  020540  051125  042522  020104
3767  020546  051050  040520  036523
```

```
3768  020554  024460     000
3769
3770  020557     125  042516  050130  EM2:    .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
3771  020564  041505  042524  020104
3772  020572  052101  042524  052116
3773  020600  047511  020116  041517
3774  020606  052503  051122  042105
3775  020614     000
3776
3777  020615     115  051501  041123  EM3:    .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
3778  020622  051525  050040  051101
3779  020630  052111  020131  051105
3780  020636  047522  020122  046450
3781  020644  050103  036505  024461
3782  020652     000
3783
3784  020653     115  051501  041123  EM4:    .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
3785  020660  051525  050040  051101
3786  020666  052111  020131  051105
3787  020674  047522  020122  050050
3788  020702  051101  030475  000051
3789
3790  020710  042101  051104  051505  EM5:    .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
3791  020716  020123  046120  043525
3792  020724  041440  040510  043516
3793  020732  020105  044502  020124
3794  020740  042523  000124
3795
3796  020744  044122  030461  042040  EM6:    .ASCIZ  /RH11 DIDN'T RESPOND TO ADDRESSING/
3797  020752  042111  023516  020124
3798  020760  042522  050123  047117
3799  020766  020104  047524  040440
3800  020774  042104  042522  051523
3801  021002  047111  000107
3802
3803  021006  051104  053111  020105  EM10:   .ASCIZ  /DRIVE OR DATA ERROR/
3804  021014  051117  042040  052101
3805  021022  020101  051105  047522
3806  021030  000122
3807
3808  021032  051104  053111  020105  EM11:   .ASCIZ  /DRIVE UNSAFE ERROR/
3809  021040  047125  040523  042506
3810  021046  042440  051122  051117
3811  021054     000
3812
3813  021055     110  040505  020104  EM12:   .ASCIZ  /HEAD OUT OF ALIGNMENT/
3814  021062  052517  020124  043117
3815  021070  040440  044514  047107
3816  021076  042515  052116     000
3817
3818  021103     110  040505  020104  EM13:   .ASCIZ  /HEAD TOO FAR OUT OF ALIGNMENT/
3819  021110  047524  020117  040506
3820  021116  020122  052517  020124
3821  021124  043117  040440  044514
```

```
 3822  021132  047107  042515  052116
 3823  021140     000
 3824
 3825  021141     123  043117  053524  EM14:   .ASCIZ  /SOFTWARE TIMEOUT/
 3826  021146  051101  020105  044524
 3827  021154  042515  052517  000124
 3828
 3829  021162  047125  047503  051122  EM15:   .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
 3830  021170  041505  040524  046102
 3831  021176  020105  040515  051523
 3832  021204  052502  020123  040520
 3833  021212  044522  054524  042440
 3834  021220  051122  051117     000
 3835
 3836  021225     104  044522  042526  EM16:   .ASCIZ  /DRIVE UNAVAILABLE/
 3837  021232  052440  040516  040526
 3838  021240  046111  041101  042514
 3839  021246     000
 3840
 3841
 3842  021247     122  040520  000123  DH1:    .ASCIZ  /RPAS/
 3843
 3844  021254  051104  053111  020105  DH2:    .ASCIZ  /DRIVE   RPDS1   RPER1   RPER2   RPER3   RPAS/
 3845  021262  020040  050122  051504
 3846  021270  020061  020040  050122
 3847  021276  051105  020061  020040
 3848  021304  050122  051105  020062
 3849  021312  020040  050122  051105
 3850  021320  020063  020040  050122
 3851  021326  051501     000
 3852
 3853  021331     104  044522  042526  DH3:    .ASCIZ  /DRIVE   REG ADR  DATA/
 3854  021336  020040  051040  043505
 3855  021344  040440  051104  020040
 3856  021352  040504  040524     000
 3857
 3858  021357     104  044522  042526  DH4:    .ASCIZ  /DRIVE   REG ADR    GOOD    BAD/
 3859  021364  020040  051040  043505
 3860  021372  040440  051104  020040
 3861  021400  020040  047507  042117
 3862  021406  020040  020040  040502
 3863  021414  000104
 3864
 3865  021416  044122  040440  042104  DH6:    .ASCIZ  /RH ADDRESS/
 3866  021424  042522  051523     000
 3867
 3868  021431     104  044522  042526  DH10:   .ASCIZ  /DRIVE   RPCS1   RPCS2   RPDS1   RPER1   RPER2   RPER3/
 3869  021436  020040  051040  041520
 3870  021444  030523  020040  051040
 3871  021452  041520  031123  020040
 3872  021460  051040  042120  030523
 3873  021466  020040  051040  042520
 3874  021474  030522  020040  051040
 3875  021502  042520  031122  020040
```

```
3876  021510  051040  042520  031522
3877  021516     000
3878
3879  021517     040  020040  020040  DH12:   .ASCII  /                TRK CENTER/<CR><LF>
3880  021524  020040  020040  020040
3881  021532  020040  020040  052040
3882  021540  045522  041440  047105
3883  021546  042524  006522     012
3884  021553     040  020040  042510          .ASCIZ  /   HEAD    CYL (IN U INCHES)/
3885  021560  042101  020040  020040
3886  021566  054503  020114  044450
3887  021574  020116  020125  047111
3888  021602  044103  051505  000051
3889
3890  021610  020040  044040  040505  DH13:   .ASCIZ  /   HEAD    CYL/
3891  021616  020104  020040  041440
3892  021624  046131     000
3893
3894  021627     104  044522  042526  DH16:   .ASCIZ  /DRIVE/
3895  021634     000
3896
3897          021636                          .EVEN
3898
3899  021636  001170  000000          DT1:    .WORD   ATTN,0
3900
3901  021642  001166  007536  007540  DT2:    .WORD   DRIVE,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6,ATTN,0
3902  021650  007542  007544  001170
3903  021656  000000
3904
3905  021660  001166  014662  014664  DT3:    .WORD   DRIVE,RD.ADR,RD.WRD,0
3906  021666  000000
3907
3908  021670  001166  015102  015100  DT4:    .WORD   DRIVE,WRT.AD,WRT.WD,RD.WRD,0
3909  021676  014664  000000
3910
3911  021702  001264  000000          DT5:    .WORD   $RPADR,0
3912
3913  021706  001164  016140  016150  DT10:   .WORD   DRVSEL,REG,REG+RPCS2,REG+RPDS1,REG+RPER1,REG+RPER2,REG+RPER3,0
3914  021714  016152  016154  016200
3915  021722  016202  000000
3916
3917  021726  001172  016132  001204  DT12:   .WORD   $HEAD,DPB+CYL,PLUS,0
3918  021734  000000
3919
3920  021736  001172  016132  000000  DT13:   .WORD   $HEAD,DPB+CYL,0
3921
3922  021744  001164  000000          DT16:   .WORD   DRVSEL,0
3923
3924
3925
3926  021750     000                  DF1:    .BYTE   0
3927
3928  021751     000     000     000  DF2:    .BYTE   0,0,0,0,0,0
3929  021754     000     000     000
```

```

021757   000     000      000  DF3:     .BYTE   0,0,0

021762   000     000      000  DF4:     .BYTE   0,0,0,0
021765   000

021766   000     000      000  DF10:    .BYTE   0,0,0,0,0,0,0
021771   000     000      000
021774   000

021775   001     001      001  DF12:    .BYTE   1,1,1

022000   001     001           DF13:    .BYTE   1,1


                               .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
                               ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
                               ;OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
                               ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
                               ;REQUIRED.
                               ;NOTE: THIS ROUTINE DESTROYS R0-R4
                               ;CALL
                               ;
                               ;        JSR     PC,BUSADR
                               ;        RETURN

022002   005737  001242        BUSADR:  TST     CHGADR          ;INPUT FROM TTY REQUESTED?
022006   001450                         BEQ     7$              ;NO--BRANCH
022010   005037  001242                 CLR     CHGADR          ;YES--CLEAR THE REQUEST FLAG
022014   012700  001264        1$:      MOV     #$RPADR,R0      ;FIRST ADDRESS
022020   104401  022202                 TYPE    ,MRPCS1         ;"RPCS1="
022024   012046                         MOV     (R0)+,-(SP)     ;PRESENT RPCS1 ADDRESS
022026   104402                         TYPOC                   ;TYPE IT
022030   104401  022224                 TYPE    ,LINSP          ;2 SPACES
022034   104411                         RDLIN                   ;GET THE ENTRY
022036   012601                         MOV     (SP)+,R1        ;ADDRESS OF ASCII TEXT
022040   004537  022230                 JSR     R5,CK.NUM       ;CHECK THE NUMBER
022044   022064                         3$                      ;CARRIAGE RETURN ONLY ENTERED
022046   022130                         7$                      ;PERIOD ONLY ENTERED
022050   022014                         1$                      ;ILLEGAL INPUT
022052   022060                         2$                      ;TERMINATED WITH A CARRIAGE RETURN
022054   022014                         1$                      ;TERMINATED WITH A ","
022056   022124                         4$                      ;TERMINATED WITH A "."
022060   010260  177776        2$:      MOV     R2,-2(R0)       ;SAVE NEW RPCS1
022064   104401  022213        3$:      TYPE    ,MRHVEC         ;"RHVEC="
022070   012046                         MOV     (R0)+,-(SP)     ;PRESENT RH11 VECTOR ADDRESS ON THE STACK
022072   104402                         TYPOC                   ;TYPE IT
022074   104401  022224                 TYPE    ,LINSP          ;2 SPACES
022100   104411                         RDLIN                   ;READ THE ENTRY
022102   012601                         MOV     (SP)+,R1        ;ASCII TEXT ADDRESS
022104   004537  022230                 JSR     R5,CK.NUM       ;CHECK THE NUMBER
022110   022130                         7$                      ;CARRIAGE RETURN ONLY ENTERED
022112   022130                         7$                      ;PERIOD ONLY ENTERED
022114   022064                         3$                      ;ILLEGAL INPUT
```

```
3984  022116  022124                          4$                      ;TERMINATED WITH A CARRIAGE RETURN
3985  022120  022064                          3$                      ;TERMINATED WITH A "."
3986  022122  022124                          4$                      ;TERMINATED WITH A "'."
3987  022124  010260  177776          4$:     MOV     R2,-2(R0)               ;SAVE INPUT
3988  022130  013701  000004          7$:     MOV     ERRVEC,R1       ;SAVE THE ERROR VECTOR
3989  022134  012737  022170  000004          MOV     #8$,ERRVEC      ;SETUP FOR TRAP
3990  022142  005777  157116                  TST     @$RPADR         ;CHECK FOR RH11
3991  022146  010137  000004                  MOV     R1,ERRVEC       ;RESTORE ERROR VECTOR
3992  022152  012700  001264                  MOV     #$RPADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
3993  022156  012701  007704                  MOV     #RPADR,R1       ;FIRST ADDRESS OF WHERE TO PUT THEM
3994  022162  012021                          MOV     (R0)+,(R1)+     ;BUS ADDRESS
3995  022164  012021                          MOV     (R0)+,(R1)+     ;VECTOR ADDRESS
3996  022166  000207                          RTS     PC              ;RETURN
3997  022170  010137  000004          8$:     MOV     R1,ERRVEC       ;RESTORE ERROR VECTOR
3998  022174  022626                          CMP     (SP)+,(SP)+     ;CLEAN OFF THE STACK
3999  022176  104006                          ERROR   6               ;REPORT THE ERROR
4000  022200  000705                          BR      1$              ;ASK FOR BUS ADDRESS
4001
4002  022202  050122  051503  020061  MRPCS1: .ASCIZ  @RPCS1 = @
4003  022210  020075     000
4004  022213     122  053110  041505  MRHVEC: .ASCIZ  @RHVEC = @
4005  022220  036440  000040
4006  022224  020040     000          LINSP:  .ASCIZ  @ @
4007
4008          022230                  .EVEN
4009
4010                                  .SBTTL  CK.NUM - CHECK NUMBER (OCTAL)
4011                                  ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
4012                                  ;AND FORMS AN OCTAL NUMBER IN R2
4013                                  ;CALL:
4014                                  ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
4015                                  ;       MOV     #NUM,R2         ;MAX SIZE OF INPUT NUMBER
4016                                  ;       JSR     R5,CK.NUM       ;GO FORM THE NUMBER
4017                                  ;       RETURN  ADR1            ;"CR" ONLY ENTERED -- R2 = 0
4018                                  ;       RETURN  ADR2            ;"PERIOD" ONLY ENTERED -- R2 = 0
4019                                  ;       RETURN  ADR3            ;ILLEGAL CHARACTER IN THE INPUT STRING
4020                                  ;       RETURN  ADR4            ;"CR" ENTERED -- R2 = NUMBER
4021                                  ;       RETURN  ADR5            ;"COMMA" -- R2 = NUMBER
4022                                  ;       RETURN  ADR6            ;"PERIOD" -- R2 = NUMBER
4023
4024  022230  010446          CK.NUM: MOV     R4,-(SP)        ;SAVE R4
4025  022232  010346                  MOV     R3,-(SP)        ;SAVE R3
4026  022234  010246                  MOV     R2,-(SP)        ;SAVE R2
4027  022236  005004                  CLR     R4              ;RETURN POINTER
4028  022240  005003                  CLR     R3              ;START NUMBER AT ZERO
4029  022242  005002                  CLR     R2              ;STORE RESULT
4030  022244  004537  022444          JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
4031  022250  022346                  6$                      ;ILLEGAL CHARACTER
4032  022252  022352                  8$                      ;CARRIAGE RETURN
4033  022254  022346                  6$                      ;" "
4034  022256  022350                  7$                      ;"'"
4035  022260  022264                  1$                      ;DIGIT 0-7
4036  022262  022346                  6$                      ;DIGIT 8-9
4037  022264  062705  000004  1$:     ADD     #4,R5           ;INCREMENT RETURN PAST "CR" AND "PERIOD" ONLY RETURNS
```

```
4038  022270  006303                2$:     ASL     R3              ;FOR THE OCTAL NUMBER IN R3
4039  022272  103425                        BCS     6$              ;DON'T LET IT GET TO BIG
4040  022274  006303                        ASL     R3
4041  022276  103423                        BCS     6$
4042  022300  006303                        ASL     R3
4043  022302  103421                        BCS     6$
4044  022304  060203                        ADD     R2,R3
4045  022306  004537  022444                JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
4046  022312  022352                        8$                      ;ILLEGAL CHARACTER
4047  022314  022336                        5$                      ;CARRIAGE RETURN
4048  022316  022334                        4$                      ;"."
4049  022320  022326                        3$                      ;","
4050  022322  022270                        2$                      ;DIGIT 0-7
4051  022324  022352                        8$                      ;DIGIT 8-9
4052  022326  105711                3$:     TSTB    (R1)            ;DOES A "CR" FOLLOW THE "PERIOD"
4053  022330  001010                        BNE     8$              ;BR IF NOT
4054  022332  005724                        TST     (R4)+           ;INCREMENT THE RETURN
4055  022334  005724                4$:     TST     (R4)+           ;INCREMENT THE RETURN INDEX
4056  022336  005724                5$:     TST     (R4)+           ;INCREMENT THE RETURN INDEX
4057  022340  020316                        CMP     R3,(SP)         ;INPUT VALUE TOO LARGE ?
4058  022342  101003                        BHI     8$              ;BR IF IT IS
4059  022344  000401                        BR      7$              ;BR IF NOT
4060  022346  005725                6$:     TST     (R5)+           ;INCREMENT THE RETURN ADDRESS
4061  022350  005725                7$:     TST     (R5)+           ;INCREMENT THE RETURN ADDRESS
4062  022352  060405                8$:     ADD     R4,R5           ;SETUP FOR PROPER RETURN
4063  022354  010302                        MOV     R3,R2           ;LOAD ENTERED VALUE
4064  022356  005726                        TST     (SP)+           ;CLEAN OFF THE STACK
4065  022360  012603                        MOV     (SP)+,R3        ;RESTORE R3
4066  022362  012604                        MOV     (SP)+,R4        ;RESTORE R4
4067  022364  011505                        MOV     (R5),R5         ;GET RETURN ADDRESS
4068  022366  000205                        RTS     R5              ;RETURN
4069
4070
4071
4072                                  ;THIS ROUTINE IS USED TO CHECK IF AN
4073                                  ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
4074                                  ;CALL
4075                                  ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
4076                                  ;       JSR     R5,CK.OCT       ;CHECK THE CHARACTER
4077                                  ;       RETURN1                 ;CHARACTER IS NOT BETWEEN 0-7
4078                                  ;       RETURN2                 ;CHARACTER IS IN R2 AS A
4079                                  ;                               ;OCTAL DIGIT
4080
4081  022370  121127  000060         CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
4082  022374  103407                        BLO     1$              ;YES -- BRANCH
4083  022376  121127  000067                CMPB    (R1),#'7        ;GREATER THAN SEVEN?
4084  022402  101004                        BHI     1$              ;YES -- BRANCH
4085  022404  111102                        MOVB    (R1),R2         ;GET THE CHARACTER
4086  022406  042702  177770                BIC     #^C7,R2         ;STRIP AWAY THE ASCII
4087  022412  005725                        TST     (R5)+           ;ADJUST FOR RETURN
4088  022414  000205                1$:     RTS     R5              ;RETURN
4089
4090                                  ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
4091                                  ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
```

```
                                        ;CALL
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                        ;       JSR     R5,CK.DEC       ;CHECK THE CHARACTER
                                        ;       RETURN1                 ;NOT BETWEEN 0 AND 9
                                        ;       RETURN2                 ;BETWEEN 0 AND 9
                                        ;                               ;R2 = DIGIT

022416  121127  000060        CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
022422  103407                        BLO     1$              ;YES -- BRANCH
022424  121127  000071                CMPB    (R1),#'9        ;GREATER THAN NINE?
022430  101004                        BHI     1$              ;YES -- BRANCH
022432  111102                        MOVB    (R1),R2         ;GET THE CHARACTER
022434  042702  000060                BIC     #'0,R2          ;STRIP AWAY THE ASCII
022440  005725                        TST     (R5)+           ;ADJUST FOR RETURN
022442  000205                1$:     RTS     R5              ;RETURN

                                        ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
                                        ;DETERMINE WHAT IT IS.
                                        ;CALL
                                        ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                        ;       JSR     R5,CK.CHR       ;CHECK CHARACTER
                                        ;       RETURN  ADR1            ;UNKNOWN CHARACTER
                                        ;       RETURN  ADR2            ;CARRIAGE RETURN * (R1)=ADR+1
                                        ;       RETURN  ADR3            ;COMMA * (R1)=ADR+1
                                        ;       RETURN  ADR4            ;PERIOD * (R1)=ADR+1
                                        ;       RETURN  ADR5            ;DIGIT BETWEEN 0 AND 7.
                                        ;       RETURN  ADR6            ;DIGIT BETWEEN 8 AND 9.
                                        ;                               ;R2 = DIGIT * (R1)=ADR+1

022444  105711                CK.CHR: TSTB    (R1)            ;"CARRIAGE RETURN"?
022446  001417                        BEQ     3$              ;YES -- BRANCH
022450  121127  000054                CMPB    (R1),#',        ;"COMMA"?
022454  001413                        BEQ     2$              ;YES -- BRANCH
022456  121127  000056                CMPB    (R1),#'.        ;"PERIOD"?
022462  001407                        BEQ     1$              ;YES -- BRANCH
022464  004537  022416                JSR     R5,CK.DEC       ;"DIGIT"?
022470  000410                        BR      4$              ;NO -- BRANCH
022472  004537  022370                JSR     R5,CK.OCT       ;OCTAL ?
022476  005725                        TST     (R5)+           ;DIGIT BETWEEN 8-9
022500  005725                        TST     (R5)+           ;DIGIT BETWEEN 0-7
022502  005725                1$:     TST     (R5)+           ;PERIOD
022504  005725                2$:     TST     (R5)+           ;COMMA
022506  005725                3$:     TST     (R5)+           ;CARRIAGE RETURN
022510  005201                        INC     R1              ;MOVE POINTER TO NEXT CHARACTER
022512  011505                4$:     MOV     (R5),R5         ;UNKNOWN CHARACTER
022514  000205                        RTS     R5              ;RETURN


        000001                          .END
```

```
ABS     = 000200        ACK     = 000123        ACL     = 000040        ACTDRV  007622
ACTSTR  007623          ACU     = 100000        ALIGN   020037        ALN245  020232
ALN496  020300          ADE     = 001000        ATA     = 100000        ATABIT  007672
ATTN    001170          AT0     = 000001        AT1     = 000002        AT2     = 000004
AT3     = 000010        AT4     = 000020        AT5     = 000040        AT6     = 000100
AT7     = 000200        AT8     = 000400        AT9     = 001000        BADCYL  020447
BADTRK  020420          BAI     = 000010        BITIND  001200        BIT0    = 000001
BIT00   = 000001        BIT01   = 000002        BIT02   = 000004        BIT03   = 000010
BIT04   = 000020        BIT05   = 000040        BIT06   = 000100        BIT07   = 000200
BIT08   = 000400        BIT09   = 001000        BIT1    = 000002        BIT10   = 002000
BIT11   = 004000        BIT12   = 010000        BIT13   = 020000        BIT14   = 040000
BIT15   = 100000        BIT2    = 000004        BIT3    = 000010        BIT4    = 000020
BIT5    = 000040        BIT6    = 000100        BIT7    = 000200        BIT8    = 000400
BIT9    = 001000        BPTVEC= 000014          BUF     = 000006        BUSADR  022002
CHECK   003032          CHGADR  001242          CI1     011244          CI3     011352
CI4     011460          CI5     012036          CI6     012060          CI7     012074
CI8     012122          CIB     012174          CKSWR = 104407          CK.CHR  022444
CK.DEC  022416          CK.NUM  022230          CK.OCT  022370          CLR     = 000040
CLRQUE  015662          CODE    = 000001        COMND = 000002          CR      = 000015
CRLF    = 000200        CSF     = 000002        CSU     = 000010        CYL     = 000012
CYLDER  002646          CYL245  020122          CYL496  020166          DCK     = 100000
DCL     = 000100        DCU     = 000001        DDISP = 177570          DDU     016654
DDUALN  003530          DE1     = 000040        DFF20 = 000002          DF1     021750
DF10    021766          DF12    021775          DF13    022000          DF2     021751
DF3     021757          DF4     021762          DH1     021247          DH10    021431
DH12    021517          DH13    021610          DH16    021627          DH2     021254
DH3     021331          DH4     021357          DH6     021416          DIGB    = 000004
DISPLA  001142          DISPRE  000174          DLT     = 100000        DL64    = 000020
DMD     = 000001        DPB     016120          DPINT   007576          DPR     = 000400
DPRGS   007606          DRIVE   001166          DRIVER  003736          DRQ     = 004000
DRVACT  007546          DRVCLR= 000111          DRVINT  010134          DRVQUE  015760
DRVSEL  001164          DRVSTA  007556          DRVTYP  007566          DRY     = 000200
DSWR    = 177570        DTE     = 010000        DTSY    = 000200        DTUW    007670
DT00    = 000001        DT01    = 000002        DTC2    = 000004        DT03    = 000010
DT04    = 000020        DT05    = 000040        DT06    = 000100        DT07    = 000200
DT08    = 000400        DTI     021636          DT10    021706          DT12    021726
DT13    021736          DT16    021744          DT2     021642          DT3     021660
DT4     021670          DT6     021702          DVA     = 004000        ECH     = 000100
ECI     = 004000        EMPTYQ  015740          EMTVEC= 000030          EM1     020516
EM10    021006          EM11    021032          EM12    021055          EM13    021103
EM14    021141          EM15    021162          EM16    021225          EM2     020557
EM3     020615          EM4     020653          EM5     020710          EM6     020744
ENDMSG  020502          ENTCYL  020365          ENTERD  017555          ENTHD   020346
ERR     = 040000        ERRVEC= 000004          EXER    020050          EXT1    = 000001
EXT10   = 000010        EXT2    = 000002        EXT20 = 000020          EXT4    = 000004
EXT40   = 000040        FEN     = 000200        FER     = 000020        FMT22 = 010000
F1      = 000002        F2      = 000004        F3      = 000010        F4      = 000020
F8      = 000040        GETNUM  004270          GETREG= 000141          GETREQ  016034
GO      = 000001        GRV     = 000010        GTSWR = 104406          HCE     = 000200
HCI     = 002000        HCRC    = 000400        HDZERO  020064          HEADER  017667
HERDI   016210          HEAD45  016310          HEAD6   016260          HT      = 000011
IBE     = 002000        IE      = 000100        ILF     = 000001        ILR     = 000002
INCYL   001226          INIT    001744          INREG   001206          IOTVEC= 000020
IR      = 000100        ISR     012566          IXE     = 004000        LA      012430
```

```
LACNT   007634        LF    = 000012        LINSP   022224        LKS     001262
LAV     001356        LST   = 002000        MAXCYL  001212        MAXNEG  001216
MAXOFF  001220        MAXPOS  001214        MCLK  = 000002        MCPE  = 020000
MCPEMX  007702        MHS   = 001000        MIDCYL  001230        MIDTOL  001224
MINX  = 000004        MNDLTA  007716        MODE    017761        MOH   = 020000
MOL   = 010000        MPE   = 000400        MRD   = 000020        MRHVEC  022213
MRPCS1  022202        MSE   = 000020        MSTCK = 000010        MWR   = 000040
MXDLTA  007714        MXF   = 001000        MXLACT  007712        MXWNDW  007720
N     = 000261        NBA   = 100000        NED   = 010000        NEM   = 004000
NHS   = 002000        NODRV   017524        OCYL  = 100000        OFFDIR  001176
OFFSET= 000115        OFREV = 000200        OFTBL4  016405        OFTBL6  016324
OF100 = 000004        OF200 = 000010        OF25  = 000001        OF400 = 000020
OFSO  = 000002        OF800 = 000040        ONEHD   002444        OPE   = 020000
OPI   = 020000        OPT     010764        OR    = 000200        OUTCYL  001232
OUTTOL  001222        PAR   = 000010        PAT   = 000020        PC      =%000007
PGE   = 002000        PGM   = 001000        PIP   = 020000        PIRQ  = 177772
PIRQVE= 000240        PKB     001252        PKC     001254        PKCS    001250
PKV     001244        PLU   = 020000        PLUS    001204        POPQUE  016056
PRESET= 000121        PRO   = 000000        PR1   = 000040        PR2   = 000100
PR3   = 000140        PR4   = 000200        PR5   = 000240        PR6   = 000300
PR7   = 000340        PS    = 177776        PSEL  = 002000        PSU   = 000001
PSW   = 177776        PWRVEC= 000024        QCNT    015370        QDRV0   015462
QDRV1   015502        QDRV2   015522        QDRV3   015542        QDRV4   015562
QDRV5   015602        QDRV6   015622        QDRV7   015642        QINPT   015400
QOUTPT  015420        QSTART  015440        QSTOP   015442        QTERM = 015662
RANDOM  003640        RAW   = 000020        RDCHR = 104410        RDDAT = 000171
RDHD  = 000173        RDLIN = 104411        RDY   = 000200        RD.ADR  014662
RD.RP   014636        RD.RP1  014660        RD.RP2  015000        RD.RP3  015004
RD.RP4  015010        RD.WRD  014664        RECAL = 000107        REG     016140
REL   = 000113        RESREG= 104413        RESVEC= 000010        RMR   = 000004
RNOP  = 000101        RPADR   007704        RPAS  = 000016        RPBA  = 000004
RPCA  = 000034        RPCC  = 000036        RPCS1 = 000000        RPCS2 = 000010
RPDA  = 000006        RPDB  = 000022        RPDS1 = 000012        RPDT  = 000026
RPEC1 = 000044        RPEC2 = 000046        RPERRS  007536        RPER1 = 000014
RPER2 = 000040        RPER3 = 000042        RPINIT  007722        RPLA  = 000020
RPMR  = 000024        RPOF  = 000032        RPSN  = 000030        RPTMR   014110
RPVEC   007706        RPWC  = 000002        RPO4    010472        RTC   = 000117
R0    =%000000        R1    =%000001        R2    =%000002        R3    =%000003
R4    =%000004        R5    =%000005        R6    =%000006        R7    =%000007
SAVEFG  007644        SAVREG= 104412        SC      012772        SC1   = 000100
SC10  = 001000        SC11    013610        SC12    013700        SC13    013750
SC2   = 000200        SC20  = 002000        SC3     013036        SC4     013042
SC5     013054        SC6     013244        SC6A    013354        SC7     013502
SC8     013560        SEARCH= 000131        SEC   = 000010        SEEK  = 000105
SEEKFG  007646        SEKCNT  001240        SELDRV= 000145        SETFMT= 000143
SET.IE  015316        SINCNG  001202        SKI   = 040000        SP    =%000006
SRCHWT  007620        SRVCLK  004256        STACK = 001100        START   001450
START1  001460        START2  001464        STATUS= 000016        STKLMT= 177774
STO     014202        STO1    014232        STO2    014430        STO3    014500
STO5    014524        STO6    014532        STO7    014570        STO8    014620
STO9    014630        ST.CLK  004114        ST.LCL  004232        ST.PCL  004200
SVRH11  015200        SWR     001140        SWREG   000176        SW0   = 000001
SW00  = 000001        SW01  = 000002        SW02  = 000004        SW03  = 000010
SW04  = 000020        SW05  = 000040        SW06  = 000100        SW07  = 000200
```

```
$WD8    = 000400        $WD9    = 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        TABLE1  001234          TABLE2  001236          TAP     = 040000
TBITVE  = 000014        TD      012632          TDF     = 000040        TIMER   007650
TITLE   016546          TKVEC   = 000060        TOLER   001210          TOLRG   017643
TOPLN   001174          TPVEC   = 000064        TRAPVE= 000034          TRE     = 040000
TRK     = 000011        TRK1    = 004000        TRK10   = 040000        TRK2    = 010000
TRK20   = 100000        TRK4    = 020000        TRNSWT  007616          TRTVEC= 000014
TUF     = 000100        TYPDS   = 104405        TYPE    = 104401        TYPOC   = 104402
TYPON   = 104404        TYPOS   = 104403        ULDFLG  007624          UNLOAD= 000103
UNS     = 040000        UPE     = 020000        US1     = 000001        US2     = 000002
US4     = 000004        UWR     = 000010        VERIFY  020025          VUF     = 000002
VU30    = 010000        VV      = 000100        WAO     = 000002        WCE     = 040000
WCF     = 000040        WCHKD   = 000151        WCHKHD= 000153          WCU     = 000001
WLE     = 004000        WLOCK   017604          WRL     = 004000        WRTDAT= 000161
WRTHD   = 000163        WRT.AD  015102          WRT.RP  015012          WRT.R1  015076
WRT.R2  015120          WRT.R3  015170          WRT.R4  015174          WRT.R5  015176
WRT.WD  015100          WRU     = 000400        WSU     = 000004        $AUTOB  001134
$BDADR  001122          $BDDAT  001126          $BELL   006767          $CHARC  005110
$CKSWR  006077          $CMTAG  001100          $CM3    = 000000        $CNTLC  006773
$CNTLG  007005          $CNTLU  007000          $CRLF   001161          $DBLK   005556
$DIV    007136          $DTBL   005546          $ERFLG  001103          $ERMAX  001115
$ERROR  004412          $ERRFC  001116          $ERRTB  001270          $ERRTY  004520
$ERTTL  001112          $FILLC  001156          $FILLS  001155          $GDADR  001120
$GDDAT  001124          $GTSWR  006162          $HD     = 000001        $HEAD   001172
$HINUM  007132          $ICNT   001104          $INTAG  001135          $ITEMB  001114
$LF     001152          $LONUM  007134          $LPADR  001106          $LPERR  001110
$MNEW   007023          $MSWR   007012          $NULL   001154          $OCNT   005336
$OMODE  005340          $PASS   001100          $QUES   001160          $RAND   007034
$RDCHR  006434          $RDLIN  006524          $RDSZ   = 000007        $RESRE  007414
$RPADR  001264          $RPVEC  001266          $SAVRE  007356          $SETUP= 000146
$STUP   = 177777        $SWR    = 120000        $TKB    001146          $TKCNT  005566
$TKINT  005604          $TKQEN= 005603          $TKQIN  005570          $TKQOU  005572
$TKQSR  005574          $TKS    001144          $TKSRV  005654          $TN     = 000001
$TPB    001152          $TPFLG  001157          $TPS    001150          $TRAP   007452
$TRAP2  007474          $TRP    = 000014        $TRPAD  007506          $TSTNM  001102
$TTYIN  006760          $TYPDS  005342          $TYPE   004674          $TYPEC  005044
$TYPEX  005112          $TYPOC  005140          $TYPON  005154          $TYPOS  005114
$DFILL  005337          .       = 022616
```

:

ERRORS DETECTED: 0

```
*DZRJCA,DZRJC/SOL/LI:ME/NL:MC:MD:CND=RP0456.010,DZRJC.009
RUN-TIME:  74 51  0 SECONDS
CORE USED:  33K
```

Spooler runtime 13 Seconds, 83 KCS, 396 disk reads, 0 disk writes, 93 pages printed