RP04/05/06

HEAD ALIGN VERIFICATION

MD-11-DZRJC-A

EP-DZRJC-A-DL-A

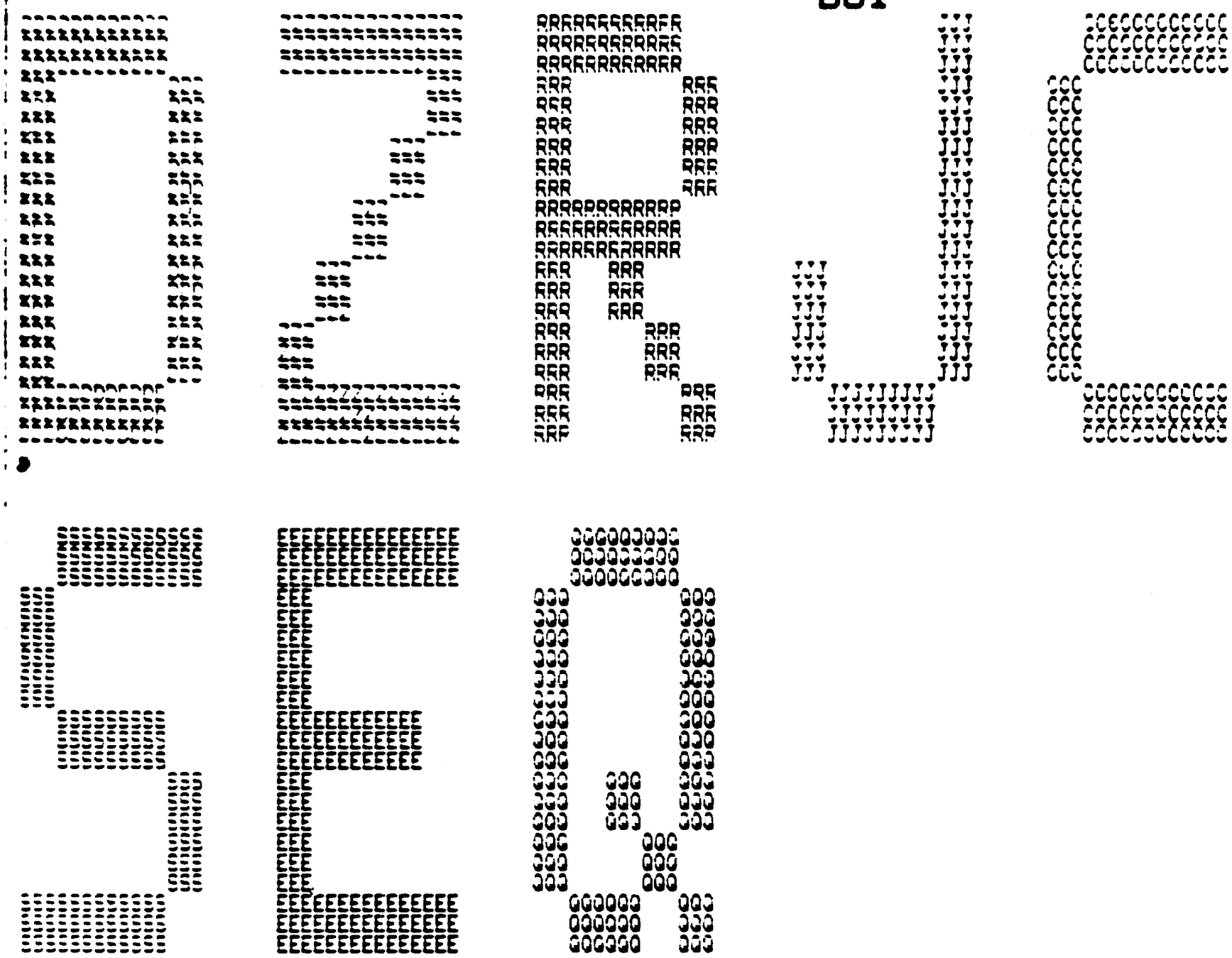
FICHE 1 OF 1

NOV 1976

digital

MADE IN USA

DZRJC
SEQ



.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

2. REQUIREMENTS

    2.1 EQUIPMENT
    2.2 PRELIMINARY PROGRAMS

3. LOADING PROCEDURES

4. STARTING PROCEDURES

    4.1 STARTING ADDRESSES
    4.2 PROGRAM & OPERATOR ACTION
    4.3 DRIVE SELECTION
    4.4 RH11 - RH70 UNIBUS ADDRESSES
    4.5 OTHER UNIBUS ADDRESSES

5. SWITCH REGISTER SETTINGS

6. ERRORS

    6.1 TEST ERRORS
    6.2 ENTRY ERRORS
    6.3 SUBSYSTEM/DEVICE ERROR MESSAGES

7. RESTRICTIONS

8. PROGRAM DESCRIPTION

    8.1 VERIFICATION MODE
        8.1.1 RP04/5 OPERATION
        8.1.2 RP06 OPERATION
    8.2 ALIGNMENT MODE
    8.3 RANDOM SEEK UTILITY

9. PROGRAM LISTING

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

5. 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.

6. ERRORS

6.1 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 19 (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 - 19. 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

16	OPERATIONAL SWITCH SETTINGS
26	BASIC DEFINITIONS
139	RH11 REGISTERS
184	RP04/5/6 REGISTERS
377	TRAP CATCHER
387	PROGRAM STARTING ADDRESS = 200
392	COMMON TAGS
469	RH11/RP04/5/6 ADDRESS & VECTOR LOCATIONS
476	ERROR POINTER TABLE
593	RP04/5/6 DRIVER COMMANDS
621	PROGRAM START AND INITIALIZATION ROUTINES
629	INITIALIZE THE COMMON TAGS
665	GET VALUE FOR SOFTWARE SWITCH REGISTER
759	SETUP TO CHECK ONLY THE SPECIFIED HEAD
800	MAIN ROUTINE - CHECK ALL HEADS AT ALL ALIGHMENT CYLNDERS
936	ROUTINE TO FIND THE TRACK CENTER
940	ROUTINE TO SELECT HEAD FOR DDU CONTROLLED HEAD ALIGNMENT
969	ROUTINE TO PERFORM 5,000 RANDOM SEEKS
994	COMMON ENTRY TO THE RP04/5/6 DRIVER
1037	SUBROUTINES
1126	MACRO ROUTINES
1128	ERROR HANDLER ROUTINE
1162	ERROR MESSAGE TYPEOUT ROUTINE
1218	TYPE ROUTINE
1288	BINARY TO OCTAL (ASCII) AND TYPE
1365	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1432	TTY INPUT ROUTINE
1690	RANDOM NUMBER GENERATOR ROUTINE
1726	INTEGER DIVIDE ROUTINE
1808	SAVE AND RESTORE R0-R5 ROUTINES
1853	TRAP DECODER
1876	TRAP TABLE
1899	SINGLE/DUAL PORT RH11/RP04/5/6 DRIVER (REV 1.0)
3296	DATA PARAMETER BLOCK
3355	HEAD CODE TABLE
3402	OFFSET CODE TABLES
3559	MESSAGES
3949	BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
4014	CK.NUM - CHECK NUMBER (OCTAL)

```
;
```

```
                        .TITLE  MD-11-DCRIC-A, RPO4 5 & 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 5 SYAC
                        ;*PACKAGE (MAINDEC-11-DCSAC-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
```

```
                        PR0=    0               ;;PRIORITY LEVEL 0
                        PR1=    40              ;;PRIORITY LEVEL 1
                        PR2=    100             ;;PRIORITY LEVEL 2
                        PR3=    140             ;;PRIORITY LEVEL 3
                        PR4=    200             ;;PRIORITY LEVEL 4
                        PR5=    240             ;;PRIORITY LEVEL 5
                        PR6=    300             ;;PRIORITY LEVEL 6
                        PR7=    340             ;;PRIORITY LEVEL 7

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
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)
```

```
000040                  CLR=    40              ;CLEAR (BIT #5)
000100                  IR=     100             ;INPUT READY (BIT #6)
000200                  OR=     200             ;OUTPUT READY (BIT #7)
000400                  MPE=    400             ;MASS BUS PARITY ERROR (BIT #8)
001000                  MXF=    1000            ;MISSED TRANSFER ERROR (BIT #9)
002000                  PGE=    2000            ;PROGRAM ERROR (BIT #10)
004000                  NEM=    4000            ;NON EXISTENT MEMORY (BIT #11)
010000                  NED=    10000           ;NON EXISTENT DRIVE (BIT #12)
020000                  UPE=    20000           ;UNIBUS PARITY ERROR (BIT #13)
040000                  WCE=    40000           ;WRITE CHECK ERROR (BIT #14)
100000                  DLT=    100000          ;DATA LATE (BIT #15)

                        ;DATA BUFFER REGISTER (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)

                        ;DFS=   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                  .V=     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 #1 (RPER1) (#02)

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

```
000002          ILR=    2               ;ILLEGAL REGISTER (BIT #1
000004          RMR=    4               ;REGISTER MODIFICATION REFUSED  BIT #2
000010          PAR=    10              ;PARITY ERROR (BIT #3)
000020          FER=    20              ;FORMAT ERROR (BIT #4)
000040          WCF=    40              ;WRITE CLOCK FAIL (BIT #5)
000100          ECH=    100             ;ECC HARD ERROR (BIT #6)
000200          HCE=    200             ;HEADER COMPARE ERROR (BIT #7)
000400          HCRC=   400             ;HEADER CRC ERROR (BIT #8)
001000          AOE=    1000            ;ADDRESS OVERFLOW ERROR (BIT #9)
002000          IAE=    2000            ;INVALID ADDRESS ERROR (BIT #10)
004000          WLE=    4000            ;WRITE LOCK ERROR (BIT #11)
010000          DTE=    10000           ;DRIVE TIMING ERROR (BIT #12)
020000          OPI=    20000           ;OPERATION INCOMPLETE (BIT #13
040000          UNS=    40000           ;DRIVE UNSAFE (BIT #14)
100000          DCK=    100000          ;DATA CHECK ERROR (BIT 15)

                ;MAINTAINABILITY REGISTER (RPMR)(#03)

000001          DMD=    1               ;DIAGNOSTIC MODE  BIT #0)
000002          MCLK=   2               ;MAINTAINABILITY CLOCK (BIT #1)
000004          MINX=   4               ;MAINTAINABILITY INDEX (BIT #2)
000010          MSTCK=  10              ;MAINTAINABILITY SECTOR CLOCK  BIT #3)
000020          MRD=    20              ;MAINTAINABILITY READ (BIT #4)
000040          MWR=    40              ;MAINTAINABILITY WRITE (BIT #5)
000200          DTSY=   200             ;MAINTAINABILITY SYNC DETECTED (BIT #7

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

000001          AT0=    1               ;DEVICE 0 (BIT #0)
000002          AT1=    2               ;DEVICE 1 (BIT #1)
000004          AT2=    4               ;DEVICE 2 (BIT #2)
000010          AT3=    10              ;DEVICE 3 (BIT #3)
000020          AT4=    20              ;DEVICE 4 (BIT #4)
000040          AT5=    40              ;DEVICE 5 (BIT #5)
000100          AT6=    100             ;DEVICE 6 (BIT #6)
000200          AT7=    200             ;DEVICE 7 (BIT #7)

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

                ;DRIVE TYPE REGISTER (RPDT) (#06)

000001          DTC0=   1               ;DRIVE TYPE NUMBER BIT 1
000002          DTC1=   2               ;DRIVE TYPE NUMBER BIT 2
000004          DTC2=   4               ;DRIVE TYPE NUMBER BIT 3
000010          DTC3=   10              ;DRIVE TYPE NUMBER BIT 4
000020          DTC4=   20              ;DRIVE TYPE NUMBER BIT 5
000040          DTC5=   40              ;DRIVE TYPE NUMBER BIT 6
000100          DTC6=   100             ;DRIVE TYPE NUMBER BIT 7
000200          DTC7=   200             ;DRIVE TYPE NUMBER BIT 8
000400          DTC8=   400             ;DRIVE TYPE NUMBER BIT 9
004000          DRQ=    4000            ;DRIVE REQUEST REQUIRED (BIT #11)
020000          MOH=    20000           ;MOVING HEAD (BIT #13)
040000          TAP=    40000           ;TAPE DRIVE (BIT #14)
```

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

                                ;LOOK-AHEAD REGISTER (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 #13)
```

```
                                ;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 (PPER3) (#15)

000001                          FSU=    1               ;PACK SPEED UNSAFE (BIT #0)
000002                          VUF=    2               ;VELOCITY UNSAFE (BIT #1)
000010                          UWR=    10              ;ANY UNSAFE EXCEPT REAS 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                          WAC=    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 PHI1 ADDRESS
000204  000137  001450          JMP     START           ;START THE PROGRAM, CHANGE THE PHI1 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
```

```
000212  000000                  MAXCYL: .WORD   0               ;CONTAINS THE MAXIMUM CYLINDER ADDRESS
                                                                ;OF DRIVE UNDER TEST
000214  000000                  MAXPOS: .WORD   0               ;CONTAINS MAXIMUM POSITIVE OFFSET
000216  000000                  MAXNEG: .WORD   0               ;CONTAINS MAXIMUM NEGATIVE OFFSET
000220  000000                  MAXOFF: .WORD   0               ;CONTAINS NUMBER OF OFFSET POSITIONS
000222  000000                  OUTTOL: .WORD   0               ;CONTAINS TOLERANCE FOR INNER AND OUTER CYLINDERS
000224  000000                  MIDTOL: .WORD   0               ;CONTAINS TOLERANCE FOR THE ALIGNMENT CYLINDER
000226  000000                  INCYL:  .WORD   0               ;INNER CYLINDER ADDRESS
000230  000000                  MIDCYL: .WORD   0               ;ALIGNMENT CYLINDER ADDRESS
000232  000000                  OUTCYL: .WORD   0               ;OUTER CYLINDER ADDRESS
000234  000000                  TABLE1: .WORD   0               ;CONTAINS HEAD TABLE ADDRESS FOR 'INCYL' & 'OUTCYL'
000236  000000                  TABLE2: .WORD   0               ;CONTAINS ADDRESS OF OFFSET CODE TABLE
000240  000000                  SEKCNT: .WORD   0               ;CONTAINS SEEK COUNT
000242  000000                  CHGADR: .WORD   0               ;CHANGE RH11 BUS ADDRESS FLAG
000244  000104  000106          PKV:    .WORD   104,106         ;KW11-P VECTOR ADDRESS
000250  172540                  PKCS:   .WORD   172540          ;KW11-P CONTROL AND STATUS REG.
000252  172542                  PKB:    .WORD   172542          ;KW11-P COUNT SET BUFFER
000254  172544                  PKC:    .WORD   172544          ;KW11-P COUNTER
000256  000100  000102          LKV:    .WORD   100,102         ;KW11-L VECTOR ADDRESS
000262  177546                  LKS:    .WORD   177546          ;KW11-L STATUS REGISTER

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

                                .SBTTL  RH11/RP04/5/6 ADDRESS & VECTOR LOCATIONS

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

001264  176700                  $RPADR: .WORD   176700          ;RH11/RP04/5/6 UNIBUS ADDRESS
001266  000254                  $RPVEC: .WORD   254             ;RH11/RP04/5/6 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 RESPCND 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
```

```
001614  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001620  012737  001452  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001626  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
001634  012737  176543  001132          MOV     #176543,$HINUM  ;;PRIME THE RANDOM NUMBER GENERATOR
001642  012737  123456  001134          MOV     #123456,$LONUM  ;;BOTH HIGH AND LOW WORDS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001650  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001654  012737  001612  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001662  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
001670  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001676  022777  177777  177332          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001704  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
001706  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
001710  012716  001720          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001714  000002                          RTI
001716  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001724  012737  000174  001142          MOV     #DISPREG,DISPLAY
001732  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

001736  005227  177777                  INC     #-1             ;FIRST START ?
001742  001006                          BNE     1$              ;BR IF NOT
001744  104401  016546                  TYPE    ,TITLE          ;TYPE PROGRAM TITLE
001750  104401  016654                  TYPE    ,DDU            ;TYPE 'DDU' SETUP MESSAGE
001754  104401  016116                  TYPE,   $CRLF           ;CR-LF
001760  004737  005604          1$:     JSR     PC,$TKINT       ;TURN ON THE TTY KEYBOARD INTERRUPT
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
001764  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP ACT?
001770  001006                          BNE     67$             ;;BRANCH IF YES
001772  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
002000  001005                          BNE     68$             ;;BRANCH IF NO
002002  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
002004  000403                          BR      68$
002006  112737  000001  001134  67$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
002014                          68$:
002014  005227  177777                  INC     #-1             ;FIRST START ?
002020  001010                          BNE     INIT            ;BR IF NOT
002022  004737  022002                  JSR     PC,BUSADR       ;CHECK THE RH11 ADDRESS
002026  013737  001264  007704          MOV     $RPADR,RPADR    ;MOVE RH11 ADDRESS TO DRIVER
002034  013737  001266  007706          MOV     $RPVEC,RPVEC    ;VECTOR ADDRRESS


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

002042  012716  001100          INIT:   MOV     #STACK,(SP)     ;SETUP THE STACK ADDRESS
002046  005037  177776                  CLR     PS              ;SET PROCESSOR PRIORITY TO ZERO
002052  005037  001174                  CLR     TOPLN           ;CLEAR HEADER INDICATOR
002056  004737  005604                  JSR     PC,$TKINT       ;INITIALIZE THE TTY KEYBOARD
002062  104401  017555                  TYPE    ,ENTERD         ;ASK FOR DRIVE ASSIGNMENT
002066  004737  004270                  JSR     PC,GETNUM       ;GET DRIVE NUMBER
002072  000410                          BR      1$              ;'CR' ENTERED
002074  011637  001164                  MOV     (SP),DRVSEL     ;DRIVE NUMBER
```

```
002002  112637  016120                  MOVB    (SP)+,DPB       ;DRIVE NUMBER TO PAR BLOCK
002006  023727  001164  000010          CMP     DRVSEL,#8.      ;CHECK DRIVE NUMBER
002014  103403                          BLO     2$              ;BR IF DRIVE NUMBER OK
002016  104401  001160          1$:     TYPE    ,$QUES          ;TYPE QUESTION MARK
002022  000750                          BR      INIT            ;TRY AGAIN
002024  004737  004114          2$:     JSR     PC,ST.CLK       ;INITIALIZE THE CLOCK
002030  004737  007722                  JSR     PC,RPINIT       ;INITIALIZE THE DRIVER
002034  012737  177777  007646          MOV     #-1,SEEKFG      ;SET SEEK ALLOW FLAG
002042  012737  177777  007644          MOV     #-1,SAVEFG      ;SAVE DRIVE REGISTERS
002050  113700  016120                  MOVB    DPB,R0          ;USE DRIVE NUMBER AS AN INDEX
002054  105760  007556                  TSTB    DRVSTA(R0)      ;SEE IF DRIVE AVAILABLE
002060  003004                          BGT     3$              ;BRANCH IF IT IS
002062  104401  017524                  TYPE    ,NODRV          ;'DRIVE NOT AVAILABLE'
002066  000137  001744                  JMP     INIT            ;TRY IT AGAIN
002072  132760  000003  007566  3$:     BITB    #BIT00!BIT01,DRVTYP(R0) ;RP04/5 ?
002100  001442                          BEQ     4$              ;BR IF NOT
002102  012737  000536  001222          MOV     #350.,OUTTOL    ;SETUP 4 & 400 TOLERANCES
002110  012737  000113  001224          MOV     #75.,MIDTOL     ;TOLERANCE AT CYLINDER 245
002116  012737  000620  001226          MOV     #400.,INCYL     ;INNER ALIGNMENT CYLINDER
002124  012737  000004  001232          MOV     #4,OUTCYL       ;OUTER ALIGNMENT CYLINDER
002132  012737  000365  001230          MOV     #245.,MIDCYL    ;ALIGNMENT CYLINDER
002140  012737  016310  001234          MOV     #HEAD45,TABLE1  ;HEAD ADDRESS TABLE
002146  012737  016405  001236          MOV     #OFTBL4,TABLE2  ;OFFSET TABLE ADDRESS
002154  012737  000633  001212          MOV     #411.,MAXCYL    ;LAST CYLINDER + 1
002162  012737  175520  001216          MOV     #-1200.,MAXNEG  ;MAXIMUM NEGATIVE OFFSET VALUE
002170  012737  002260  001214          MOV     #1200.,MAXPOS   ;MAXIMUM POSITIVE OFFSET VALUE
002176  012737  000141  001220          MOV     #97.,MAXOFF     ;NUMBER OF OFFSET POSITIONS
002204  000441                          BR      5$              ;CONTINUE WITH THE SETUP
002206  012737  000257  001222  4$:     MOV     #175.,OUTTOL    ;TOLERANCE AT CYLINDERS 8 & 800
002214  012737  000113  001224          MOV     #75.,MIDTOL     ;TOLERANCE AT ALIGNMENT CYLINDER
002222  012737  001440  001226          MOV     #800.,INCYL     ;INNER ALIGNMENT CYLINDER
002230  012737  000010  001232          MOV     #8.,OUTCYL      ;OUTER ALIGNMENT CYLINDER
002236  012737  000760  001230          MOV     #496.,MIDCYL    ;ALIGNMENT CYLINDER
002244  012737  016260  001234          MOV     #HEAD6,TABLE1   ;HEAD ADDRESS TABLE
002252  012737  016324  001236          MOV     #OFTBL6,TABLE2  ;OFFSET CODE TABLE
002260  012737  001457  001212          MOV     #815.,MAXCYL    ;MAXIMUM CYLINDER ADDRESS + 1
002266  012737  176650  001216          MOV     #-600.,MAXNEG   ;MAXIMUM NEGATIVE OFFSET VALUE
002274  012737  001130  001214          MOV     #600.,MAXPOS    ;MAXIMUM POSITIVE OFFSET VALUE
002302  012737  000061  001220          MOV     #49.,MAXOFF     ;NUMBER OF OFFSET POSITIONS
002310  004537  003736          5$:     JSR     R5,DRIVER       ;RECALIBRATE
002314  000107                          .WORD   RECAL           ;DRIVER CALL
002316  032737  004000  016152          BIT     #BIT11,REG+RPDS1  ;SEE IF WRITE LOCK SET
002324  001003                          BNE     6$              ;BR IF WRITE LOCKED
002326  104401  017604                  TYPE    ,WLOCK          ;REPORT NOT WRITE LOCKED
002332  000604                          BR      INIT            ;TRY IT AGAIN
002334  104401  017761          6$:     TYPE    ,MODE           ;SEE IF ALIGN, VERIFY, OR EXERCISE
002340  104410                          RDCHR                   ;GET THE ENTRY
002342  012637  002442                  MOV     (SP)+,10$       ;SAVE THE ENTRY
002346  122737  000126  002442          CMPB    #'V,10$         ;WAS A 'V' ENTERED ?
002354  001422                          BEQ     9$              ;BR IF IT WAS
002356  022737  000101  002442          CMP     #'A,10$         ;WAS AN 'A' ENTERED ?
002364  001002                          BNE     7$              ;BR IF NOT
002366  000137  003530                  JMP     DDUALN          ;HEADS TO BE ALIGNED WITH DDU
002372  022737  000105  002442  7$:     CMP     #'E,10$         ;SEE IF EXERCISE
```

```
002400  001004                          BNE     8$              ;BR IF 'E' NOT ENTERED
002402  000137  003640                  JMP     RANDOM          ;GO TO EXERCISE ROUTINE
002406  104401  002442                  TYPE    .10$            ;ECHO THE INVALID CHARACTER
002412  104401  001160          8$:     TYPE    .$QUES          ;TYPE A QUESTION MARK
002416  000137  001744                  JMP     INIT            ;GO ALL THE WAY BACK TO DRIVE ENTR
002422  104401  020035          9$:     TYPE    .VERIFY         ;TYPE 'VERIFY'
002426  032777  000004  176504          BIT     #SW2,@SWR       ;SWITCH 2 SET ?
002434  001504                          BEQ     CYLDER          ;BR IF NOT SET - CHECK ALL HEADS
002436  000137  002444                  JMP     ONEHD           ;CHECK ONLY THE SPECIFIED HEAD
002442  000000          10$:    .WORD   0               ;OPERATOR ENTRY GOES HERE

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

                        .SBTTL  SETUP TO CHECK ONLY THE SPECIFIED HEAD

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

002444  104401  003365          ONEHD:  TYPE    ,ENTCYL         ;ASK FOR CYLINDER ADDRESS
002450  004737  004270                  JSR     PC,GETNUM       ;GET THE CYLINDER ADDRESS
002454  000421                          BR      1$              ;'CR' ENTERED
002456  012637  016132                  MOV     (SP)+,DPB+CYL   ;MOVE IT TO PARAMETER BLOCK
002462  013737  001210  001210          MOV     OUTTOL,TOLER    ;SET INITIAL ALIGNMENT TOLERANCE
002470  023737  001232  016132          CMP     OUTCYL,DPB+CYL  ;SEE IF OUTER CYLINDER
002476  001420                          BEQ     4$              ;BR IF IT IS
002500  023737  001226  016132          CMP     INCYL,DPB+CYL   ;SEE IF INNER CYLINDER
002506  001426                          BEQ     4$              ;BR IF IT IS
002510  013737  001224  001210          MOV     MIDTOL,TOLER    ;CHANGE TOLERANCE
002516  023737  001230  016132          CMP     MIDCYL,DPB+CYL  ;SEE IF ALIGNMENT CYLINDER
002524  001414                          BEQ     3$              ;BR IF YES
002526  104401  020447                  TYPE    ,BADCYL         ;REPORT INVALID CYLINDER
002532  000744                          BR      ONEHD           ;TRY IT AGAIN
002534  023727  001212  000633  1$:     CMP     MAXCYL,#411.    ;RP04/5 ?
002542  001403                          BEQ     2$              ;BR IF YES
002544  104401  020166                  TYPE    ,CYL496         ;CYLINDER 496 DEFAULT MESSAGE
002550  000402                          BR      3$              ;CONTINUE
002552  104401  020122          2$:     TYPE    ,CYL245         ;CYLINDER 245 DEFAULT MESSAGE
002556  013737  001230  016132  3$:     MOV     MIDCYL,DPB+CYL  ;CYLINDER ADDRESS
002564  104401  020346          4$:     TYPE    ,ENTHD          ;ASK FOR THE HEAD ADDRESS
002570  004737  004270                  JSR     PC,GETNUM       ;GET THE HEAD ADDRESS
002574  000411                          BR      5$              ;'CR' ENTERED
002576  012637  001172                  MOV     (SP)+,$HEAD     ;SAVE THE HEAD ADDRESS
002602  022737  000022  001172          CMP     #19.,$HEAD      ;ADDRESS VALID ?
002610  002007                          BGE     6$              ;BR IF YES
002612  104401  020420                  TYPE    ,BADTRK         ;INVALID HEAD ADDRESS MESSAGE
002616  000762                          BR      4$              ;TRY AGAIN
002620  104401  020064          5$:     TYPE    ,HDZERO         ;HEAD DEFAULT MESSAGE
002624  005037  001172                  CLR     $HEAD           ;CLEAR HEAD STORAGE
002630  004537  003736          6$:     JSR     R5,DRIVER       ;SEEK TO THE SPECIFIED CYLINDER
002634  000105                          .WORD   SEEK            ;SEEK CODE
002636  004737  003032                  JSR     PC,CHECK        ;CHECK THE HEAD
002642  000137  001744                  JMP     INIT            ;GET THE NEXT DRIVE HEAD

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

```
                                .SBTTL  MAIN ROUTINE - CHECK ALL HEADS AT ALL ALIGHMENT CYLNDERS

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

002646  013737  001230  016138  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  001146                  JMP     INIT            ;GET NEXT DRIVE OR HEAD
003014  012403  000000          1$:     MOV     (R4)+,$HEAD     ;MOVE HEAD ADDRESS FROM TABLE
003020  100004                          BMI     2$              ;BR IF END OF TABLE
003022  004737  003036                  JSR     PC,CHECK        ;CHECK THE ALIGNMENT
003026  000771                          BR      1$              ;GET NEXT HEAD VALUE
003030  000207                  2$:     RTS     PC              ;RETURN

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

                                .SBTTL  ROUTINE TO FIND THE TRACK CENTER

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

                        CHECK:  CLR     OFFDIR          ;CLEAR THE OFFSET DIRECTION INDICATOR
                                CLR     BITING          ;CLEAR THE SIGN BIT INDICATOR
                                MOV     MAXPOS,R2       ;MAXIMUM POSITIVE OFFSET VALUE
                                MOV     MAXOFF,R3       ;MAXIMUM NUMBER OF OFFSETS
                                MOV     TABLE2,R1       ;OFFSET CODE TABLE ADDRESS
                                MOV     $HEAD,-(SP)     ;HEAD VALUE TO STACK
                                MTPS    #SP             ;SWITCH FOR RPDA LOAD
                                MOVB    R0,@#RPDA       ;LOAD RPDA WITH HEAD ADDRESS
                                                        ;REGISTER ADDRESS
                                                        ;UNCORRECTABLE PARITY ERROR RETURN
                                MOV     R1,@DPB+CODE    ;OFFSET VALUE TO PARAMETER BLOCK
                                DEC     R2              ;DECREMENT OFFSET VALUE
                                BR      3$              ;BYPASS REVERSE DIRECTION SETUP
                                NEG     @DPB+CODE       ;REVERSE OFFSET VALUE IN PARM BLOCK
```

```
                                                ;INCREMENT OFFSET VALUE
                                                ;OFFSET THE DRIVE
                                                ;OFFSET OP CODE
                                                ;INITIAL SIGN VALUE STORED ?
                                                ;BR IF STORED
                                                ;SET THE SIGN BIT INDICATOR
                                                ;LEAVE THE SIGN BIT
                                                ;STORE THE SIGN CHANGE BIT

                                                ;KEEP ONLY THE SIGN VALUE
                                                ;SIGN CHANGED ?
                                                ;BRANCH IF IT DID
                                                ;DECREMENT THE OFFSET ATTEMPT COUNTER
                                                ;BRANCH IF OFFSETS NOT COMPLETED
                                                ;CHECK SWR0
                                                ;BR IF SET
                                                ;REPORT OUT OF LIMIT CONDITION
                                                ;SEE IF LOOP ON ERROR
                                                ;TYPE THE HEADING LINE ?
                                                ;BR IF NOT
                                                ;TYPE THE HEADER
                                                ;SET THE BYPASS SWITCH

                                                ;;SAVE $HEAD FOR TYPEOUT
                                                ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                                ;;SAVE DPB+CYL FOR TYPEOUT
                                                ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                                ;HEAD OUT OF LIMIT
                                                ;CR-LF
                                                ;HALT IF ERROR ?
                                                ;BR IF NOT
                                                ;HEAD JUST TYPED OUT OF TOLERANCE
                                                ;CHECK FOR LOOP
                                                ;SEE WHICH DIRECTION
                                                ;BRANCH IF FORWARD
                                                ;REVERSE DIRECTION
                                                ;FINISHED WITH REVERSE ?
                                                ;BRANCH IF YES
                                                ;SAVE THE SIGN CHANGE VALUE
                                                ;MAXIMUM NEGATIVE OFFSET VALUE
                                                ;NUMBER OF OFFSETS
                                                ;NUMBER OF OFFSETS
                                                ;BEGINNING OF REVERSE DIRECTION VALUES
                                                ;SET REVERSE DIRECTION INDICATOR
                                                ;RESET SIGN BIT INDICATOR
                                                ;START REVERSE OFFSETS
                                                ;ADD THE TWO SIGN CHANGE VALUES
                                                ;'DIVIDE' BY 2
                                                ;SAVE THE RESULT
                                                ;SEE IF VALUE NEGATIVE
                                                ;BRANCH IF POSITIVE
                                                ;RECOMPLEMENT THE VALUE
                                                ;SEE IF SWR0 SET
                                                ;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          000012  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  175474  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$:    BIT     #SW9,@SWR       ;SEE IF SWITCH 9 SET
003506  001004                  BNE     15$             ;BR IF SET
003510  032777  000002  175422  14$:    BIT     #SW1,@SWR       ;SWITCH 1 SET ?
003516  001402                  BEQ     16$             ;NOT SET
003520  000737  003032          15$:    JMP     CHECK           ;DO THE HEAD AGAIN
003524  000207                  16$:    RTS     PC              ;RETURN

                                ;;*****************************************************************
                                .SBTTL  ROUTINE TO SELECT HEAD FOR DDU CONTROLLED HEAD ALIGNMENT
                                ;;*****************************************************************
003530  104401  020027          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
```

```
                                        JSR     R0,WRT.RP       ;LOAD RPDA
                                        RPDA                    ;REGISTER ADDRESS
                                        INIT                    ;UNCORRECTABLE PARITY ERROR RETURN
                                        BR      3$              ;WAIT FOR NEXT HEAD ENTRY

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

                                .SBTTL  ROUTINE TO PERFORM 5,000 RANDOM SEEKS

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

                        RANDOM: TYPE    ,EXER                   ;TYPE 'EXERCISE'
                                MOV     #5000.,SEKCNT           ;NUMBER OF SEEKS
                        1$:     JSR     PC,$RAND                ;CYCLE THE RANDOM NUMBER GENERATOR
                                MOV     $HINUM,-(SP)            ;PUT UPPER RANDOM NUMBER ON THE STACK
                                CLR     -(SP)                   ;UPPER DIVIDEND
                                MOV     MAXCYL,-(SP)            ;PUT DIVISOR ON STACK
                                JSR     PC,$DIV                 ;GET THE REMAINDER (RANDOM CYLINDER ADDRESS)
                                CMP     (SP),DPB+CYL            ;SEE IF SAME RANDOM CYLINDER AS LAST
                                BNE     2$                      ;BR IF DIFFERENT CYLINDER
                                ADD     #4,SP                   ;ADJUST THE STACK POINTER
                                BR      1$                      ;TRY AGAIN
                        2$:     MOV     (SP)+,DPB+CYL           ;PUT CYL ADDR IN THE DPB
                                TST     (SP)+                   ;CORRECT THE STACK POINTER
                                JSR     R5,DRIVER               ;DO THE SEEK
                                .WORD   SEEK                    ;SEEK CODE
                                DEC     SEKCNT                  ;DECREMENT THE SEEK COUNT
                                BNE     1$                      ;BR IF NOT DONE
                                JMP     INIT                    ;RETURN TO COMMAND ROUTINE

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

                                .SBTTL  COMMON ENTRY TO THE RP04 5 6 DRIVER

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

                        DRIVER: MOV     (R5)+,DPB+COMND         ;COMMAND CODE
                        1$:     JSR     R0,RP04                 ;DRIVER ENTRY
                                DPB                             ;DATA PARAMETER BLOCK ADDRESS
                                BR      1$                      ;REQUEST NOT ACCEPTED
                        2$:     TST     DPB+STATUS              ;SEE IF ORDER COMPLETE
                                BEQ     2$                      ;BR IF NOT COMPLETE
                                BMI     4$                      ;BRANCH IF ERROR
                        3$:     RTS     R5                      ;RETURN
                        4$:     BIT     #BIT7,DPB+STATUS        ;DID THE ORDER TERMINATE
                                BNE     5$                      ;BRANCH IF IT DID
                                BIT     #BIT14!BIT13!BIT02!BIT01,DPB+STATUS  ;DRIVE OFFLINE OR
                                                                ;NON-EXISTENT
                                BNE     6$                      ;BR IF IT IS
                                BIT     #BIT12,DPB+STATUS       ;DRIVE UNSAFE
                                BNE     7$                      ;BR IF UNSAFE
                                BIT     #BIT11!BIT10,DPB+STATUS ;UNCORRECTABLE PARITY ERROR
                                BNE     8$                      ;BR IF YES
                                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,@PIB         ;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$:     MOV     (SP)+,2(SP)     ;SAVE THE RESULT
1111  004362  062716  000002          ADD     #2,(SP)         ;INCREMENT THE RETURN ADDRESS
1112  004366  000410                  BR      9$              ;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    .$QLES          ;'?' 'CR' 'LF'
1118  004404  000732                  BR      1$              ;TRY AGAIN
1119  004406  012616          7$:     MOV     (SP)+,(SP)      ;RESTORE THE PC
1120  004410  000207          9$:     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
                                        ;*      ERPOR   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
```

```
                                        :*      MESADR
                                        :*
                                        :*
004674  105737  001157        $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
004700  100002                        BPL     1$              ;;BR IF YES
004702  000000                        HALT                    ;;HALT HERE IF NO TERMINAL
004704  000407                        BR      3$              ;;LEAVE
004706  010046                1$:     MOV     R0,-(SP)        ;;SAVE R0
004710  017600  000002                MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
004714  112046                2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
004716  001005                        BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
004720  005726                        TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
004722  012600                60$:    MOV     (SP)+,R0        ;;RESTORE R0
004724  062716  000002        3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
004730  000002                        RTI                     ;;RETURN
004732  122716  000011        4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
004736  001430                        BEQ     8$
004740  122716  000200                CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
004744  001005                        BNE     5$
004746  005726                        TST     (SP)+           ;;POP  <CR><LF> EQUIV
004750  104401                        TYPE                    ;;TYPE A CR AND LF
004752  001161                        $CRLF
004754  105037  005110                CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
004760  000755                        BR      2$              ;;GET NEXT CHARACTER
004762  004737  005044        5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
004766  123726  001156        6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
004772  001350                        BNE     2$              ;;IF NO GO GET NEXT CHAR.
004774  013746  001154                MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                              ;;AND THE NULL CHAR.
005000  105366  000001        7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
005004  002770                        BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
005006  004737  005044                JSR     PC,$TYPEC       ;;GO TYPE A NULL
005012  105337  005110                DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
005016  000770                        BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

005020  112716  000040        8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
005024  004737  005044        9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
005030  132737  000007  005110        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
005036  001372                        BNE     9$              ;;TAB STOP
005040  005726                        TST     (SP)+           ;;POP SPACE OFF STACK
005042  000724                        BR      2$              ;;GET NEXT CHARACTER
005044  105777  174100        $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
005050  100375                        BPL     $TYPEC
005052  116677  000002  174072        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
005060  122766  000015  000002        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
005066  001003                        BNE     1$              ;;BRANCH IF NO
005070  105037  005110                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
005074  000406                        BR      $TYPEX          ;;EXIT
005076  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
005104  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
005106  105227                        INCB    (PC)+           ;;COUNT THE CHARACTER
005110  000000                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
```

```
1294  005112  000207                  $TYPEX: RTS     PC
1295
1286                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1287
1288                                  ;;**************************************************************
1289                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1290                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
1291                                  ;*$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
```

```
                                                BIC     #177770,R3      ;;GET RID OF JUNK
                                                BNE     4$              ;;TEST FOR 0
                                                TST     R4              ;;SUPPRESS THIS 0?
                                                BEQ     5$              ;;BR IF YES
                                                INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
                                        4$:     BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
                                        5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
                                                MOVB    R3,B$           ;;SAVE FOR TYPING
                                                TYPE    ,B$             ;;GO TYPE THIS DIGIT
                                        7$:     DECB    $OCNT           ;;COUNT BY 1
                                                BGT     2$              ;;BR IF MORE TO DO
                                                BLT     6$              ;;BR IF DONE
                                                INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
                                                BR      2$              ;;GO DO THE LAST DIGIT
                                        6$:     MOV     (SP)+,R5        ;;RESTORE R5
                                                MOV     (SP)+,R4        ;;RESTORE R4
                                                MOV     (SP)+,R3        ;;RESTORE R3
                                                MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
                                                MOV     (SP)+,SP
                                                RTI                     ;;RETURN
                                        B$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
                                                .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
                                        $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
                                        $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
                                        $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
                                        .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

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

                                        $TYPDS:
                                                MOV     R0,-(SP)        ;;PUSH R0 ON STACK
                                                MOV     R1,-(SP)        ;;PUSH R1 ON STACK
                                                MOV     R2,-(SP)        ;;PUSH R2 ON STACK
                                                MOV     R3,-(SP)        ;;PUSH R3 ON STACK
                                                MOV     R5,-(SP)        ;;PUSH R5 ON STACK
                                                MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
                                                MOV     20(SP),R5       ;;GET THE INPUT NUMBER
                                                BPL     1$              ;;BR IF INPUT IS POS.
                                                NEG     R5              ;;MAKE THE BINARY NUMBER POS.
                                                MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
                                        1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
                                                MOV     #$OBLK,R3       ;;SETUP THE OUTPUT POINTER
                                                MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
                                        2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
                                                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
                                        3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
```

```
                                                SBLT    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
                                        ;
005564  005037  005566                  $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
005570  012737  005574  005572                  MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
005576  013737  005574  005600                  MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
005624  012737  005654  000060                  MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
005632  012737  000200  000062                  MOV     #200,@#TKVEC+2  ;;"BR" LEVEL 4
005640  005  177560                  TST     @$TKB           ;;CLEAR DONE FLAG
005644  012  000100  177572                  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 (START).
                                        ;
005654  117746  177266                  $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
005660  042716  177600                          BIC     #^C177,(SP)     ;;STRIP THE JUNK
005664  022627  000003                          CMP     (SP)+,#3        ;;IS IT A CONTROL C?
005670  001007                          BNE     1$              ;;BRANCH IF NO
005672  104401  006773                          TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
005676  004737  005624                          JSR     PC,$TKINT       ;;INIT THE KEYBOARD
005702  005726                          TST     (SP)+           ;;CLEAN UP STACK
005704  000137  001460                          JMP     START1          ;;CONTROL C RESTART
005710  022627  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$:
005730  022737  000007  005566                  CMP     #7,$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  022627  000023                  3$:     CMP     (SP)+,#23       ;;IS IT A CONTROL-S?
005752  001021                          BNE     33$             ;;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,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
006014  000002                          RTI                     ;;RETURN
006016  005237  005566                  33$:    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?
```

```
                                                ;;BRANCH IF YES
                                                ;;MAKE IT UPPER CASE
                                                ;;AND PUT IT IN QUEUE
                                                ;;UPDATE THE POINTER
                                                ;;GO OFF THE END?
                                                ;;BRANCH IF NO
                                                ;;RESET THE POINTER
                                                ;;RETURN

                                                ;;*****************************************************************
                                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                                ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
                                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
                                                ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
                                                ;;IS THE SOFT-SWR SELECTED
                                                ;;EXIT IF NOT
                                                ;;IS A CHAR WAITING?
                                                ;;IF NOT, EXIT
                                                ;;YES
                                                ;;MAKE IT  7-BIT ASCII
                                                ;;IS IT A CONTROL-G?
                                                ;;IF NOT, PUT IT IN THE TTY QUEUE
                                                ;;AND EXIT

                                                ;;*****************************************************************
                                                ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
                                                ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
                                                ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
                                                ;;ARE WE RUNNING IN AUTO-MODE?
                                                ;;BRANCH IF YES
                                                ;;CLEAR CONTROL-G OFF STACK
                                                ;;FLUSH THE TTY INPUT QUEUE
                                                ;;DISABLE TTY KEYBOARD INTERRUPTS
                                                ;;SET INTERRUPT MODE INDICATOR

                                                ;;ECHO THE CONTROL-G (^G)
                                                ;;TYPE CURRENT CONTENTS
                                                ;;SAVE SWREG FOR TYPEOUT
                                                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                                ;;PROMPT FOR NEW SWR
                                                ;;CLEAR COUNTER
                                                ;;THE NEW SWR
                                                ;;CHAR THERE?
                                                ;;IF NOT TRY AGAIN

                                                ;;PICK UP CHAR
                                                ;;MAKE IT 7-BIT ASCII

                                                ;;IS IT A CONTROL-C?
                                                ;;BRANCH IF NOT
                                                ;;YES, ECHO CONTROL-C (^C)
                                                ;;CLEAN UP STACK
                                                ;;REENABLE TTY KEYBOARD INTERRUPTS
                                                ;;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     19$             ;;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
                                      .DSABL  LSB


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

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   $CNT_G: .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  063700  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$:     ROL     R0              ;;POSITION MSB'S
1775  007250  010003                        MOV     R0,R3           ;;COPY
1776  007252  060203                        ADD     R2,R3           ;;COMPARE DIVIDEND & DIVISOR
1777  007254  103001                        BCC     6$              ;;SK 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  000032                  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

000000  000000  000000  000000  RPERRS: .WORD   0,0,0,0
000000  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

000000          000             DRVACT: .BYTE   0               ;DRIVE 0
000000          000                     .BYTE   0               ;DRIVE 1
000000          000                     .BYTE   0               ;DRIVE 2
000000          000                     .BYTE   0               ;DRIVE 3
000000          000                     .BYTE   0               ;DRIVE 4
000000          000                     .BYTE   0               ;DRIVE 5
000000          000                     .BYTE   0               ;DRIVE 6
000000          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.
```

```
                                TRNSWT: .WORD   0

                                ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
                                        ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
                                        ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
                                        ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
                                        ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.

                                SRCHWT: .WORD   0

                                ;RP04/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
                                        ;ACTDRV=0 IF DRIVER IS INACTIVE
                                        ;ACTDRV>0 IF DRIVER IS ACTIVE

                                ACTDRV: .BYTE   0

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

                                ACTSTR: .BYTE   0

                                ;UNLOAD FLAG (ULDFLG=8 BYTES)
                                        ;ULDFLG=0 IF NO UNLOAD COMMAND
                                        ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
                                        ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE

                                ULDFLG: .BYTE   0               ;DRIVE 0
                                        .BYTE   0               ;DRIVE 1
                                        .BYTE   0               ;DRIVE 2
                                        .BYTE   0               ;DRIVE 3
                                        .BYTE   0               ;DRIVE 4
                                        .BYTE   0               ;DRIVE 5
                                        .BYTE   0               ;DRIVE 6
                                        .BYTE   0               ;DRIVE 7

                                ;LOOK AHEAD COUNT (LACNT=8 BYTES)
                                        ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED

                                LACNT:  .BYTE   0               ;DRIVE 0
                                        .BYTE   0               ;DRIVE 1
                                        .BYTE   0               ;DRIVE 2
                                        .BYTE   0               ;DRIVE 3
                                        .BYTE   0               ;DRIVE 4
                                        .BYTE   0               ;DRIVE 5
                                        .BYTE   0               ;DRIVE 6
                                        .BYTE   0               ;DRIVE 7

                                ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
                                        ;SAVEFG <0 IF SAVE THE RH11/RP04 5/6 REGISTERS WHEN THE
                                        ;OPERATION IS COMPLETED AS PER .DPB+14).
                                        ;SAVEFG=0 IF SAVE THE RH11/RP04 5/6 REGISTERS, AS PER
                                        ;(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 (ATA2IT=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)
```

```
                        MXLACT: .WORD   4
                        ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)

                        MXDLTA: .WORD   8.*64.
                        ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)

                        MNDLTA: .WORD   2*64.
                        ;MAXIMUM SEARCH FOR I O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)

                        MXWNDW: .WORD   5

                        ;DEFINITIONS OF THE RH11 RP04/5/6 ADDRESS INDEXES

                        RPCS1=0                         ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
                        RPWC=2                          ;WORD COUNT REGISTER (NOT A DRIVE REG)
                        RP8A=4                          ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
                        RPDA=6                          ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
                        RPCS2=10                        ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
                        RPDS1=12                        ;DRIVE STATUS REGISTER (DRIVE REG 01)
                        RPER1=14                        ;ERROR REGISTER #1 (DRIVE REG. 02)
                        RPAS=16                         ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
                        RPLA=20                         ;LOOK AHEAD REGISTER (DRIVE REG. 07)
                        RPDB=22                         ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
                        RPMR=24                         ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
                        RPDT=26                         ;DRIVE TYPE REGISTER (DRIVE REG. 06)
                        RPSN=30                         ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
                        RPOF=32                         ;OFFSET REGISTER (DRIVE REG. 11)
                        RPDA=34                         ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
                        RPCC=36                         ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
                        RPER2=40                        ;ERROR REGISTER #2 (DRIVE REG. 14)
                        RPER3=42                        ;ERROR REGISTER #3 (DRIVE REG. 15)
                        RPEC1=44                        ;ECC POSITION REGISTER (DRIVE REG. 16)
                        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
                        ;
                        RPINIT: SAVREG                          ;SAVE R0 - R5
                                MOV     @#PS,-(SP)              ;SAVE THE PRESENT PROCESSOR STATUS
                                MOV     #(5*32.),@#PS           ;CHANGE THE PRIORITY TO 5
                                JSR     PC,CLRQUE               ;CLEAR ALL REQUEST QUEUES
                                MOV     #RPERRS,R1              ;FIRST ADDRESS TO BE CLEARED
                                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/RPC4/5/6 VECTOR
                                        MOV     #ISR,(R3)+
                                        MOV     RPVEC+2,(R3)
                                        MOV     RPADR,R4        ;FIRST ADDRESS OF RH11/RPC4
                                        MOV     #BIT05,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     #^C07,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 RPC4/5/6. 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 RPC4 5 6 (RPCS1)
                                ;       JSR     R0,DRVINT       ;CALLED BY A JSR
                                ;       RETURN1                 ;ERROR OCCURRED (PARITY)
                                ;       RETURN2                 ;NORMAL RETURN
                                ;

                                DRVINT: MOV     R5,-(SP)        ;SAVE R5
                                        CLRB    DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
                                        CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
```

```
                                                CLRB    ULDFLG(R1)          ;CLEAR THE UNLOAD FLAG
                                                MOV     R1,RPCS2(R4         ;SELECT A DRIVE
                                                MOVB    #111,RPCS1(R4)      ;DO A DRIVE CLEAR COMMAND :3 SEIZE DRIVE
                                                BIT     #BIT12,RPCS2(R4)    ;NONEXISTENT DRIVE?
                                                BEQ     1$                  ;NO---BRANCH
                                                JSR     PC,SET.IE           ;GO SET "IE" WITHOUT A "TRE"
                                                SR      6$                  ;LEAVE THIS ROUTINE
                                        1$:     CLRB    DRVSTA(R1)          ;SET DRIVE STATUS TO OFFLINE
                                                BIT     #BIT11,RPCS1(R4)    ;SEE IF DRIVE AVAILABLE
                                                BEQ     7$                  ;BR IF DRIVE NOT AVAILABLE
                                                JSR     R0,RD.RP            ;READ THE DRIVE TYPE REG.
                                                RPDT
                                                8$                          ;ERROR RETURN ADDRESS
                                                MOV     (SP)+,R5            ;PUT DRIVE TYPE IN R5
                                                MOVB    #1,DRVTYP(R1)       ;SET RP04 INDICATOR
                                                CMP     #20020,R5           ;IS IT A SINGLE PORT RP04?
                                                BEQ     2$                  ;BRANCH IF YES
                                                CMP     #24020,R5           ;IS IT A DUAL PORT RP04?
                                                BEQ     2$                  ;BR IF YES
                                                MOVB    #2,DRVTYP(R1)       ;SET RP05 INDICATOR
                                                CMP     #20021,R5           ;SINGLE PORT RP05 ?
                                                BEQ     2$                  ;BR IF YES
                                                CMP     #24021,R5           ;DUAL PORT RP05 ?
                                                BEQ     2$                  ;BR IF YES
                                                MOVB    #4,DRVTYP(R1)       ;SET RP06 INDICATOR
                                                CMP     #20022,R5           ;SINGLE PORT RP06 ?
                                                BEQ     2$                  ;BR IF YES
                                                CMP     #24022,R5           ;DUAL PORT RP06 ?
                                                BEQ     2$                  ;BR IF YES
                                                MOVB    #-1,DRVTYP(R1)      ;SET INDICATOR TO 'OTHER'
                                                BR      6$                  ;EXIT
                                        2$:     MOV     #121,-(SP)          ;DO A "READ-IN PRESET"
                                                JSR     R0,WRT.RP
                                                RPCS1
                                                8$
                                                MOV     #BIT12,-(SP)        ;SET FMT22=1
                                                JSR     R0,WRT.RP
                                                RPOF
                                                8$
                                                JSR     R0,RD.RP            ;READ RPCS1
                                                RPDS1
                                                9$
                                                MOV     (SP)+,R5            ;AND SAVE IT IN R5
                                                BPL     4$                  ;BRANCH IF ATA=0
                                                MOVB    ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
                                                JSR     R0,RD.RP            ;FIND OUT WHY ATA=1
                                                RPER1
                                                9$
                                                ROL     (SP)+               ;IS IT UNSAFE?
                                                BPL     4$                  ;BR IF NOT
                                                MOVB    #-1,DRVSTA(R1)      ;SET UNSAFE INDICATOR
                                                BR      6$                  ;EXIT
                                        4$:     COM     R5                  ;CHECK MOL, DPR, DRY, AND VV
                                                BIC     #^C BIT12!BIT08!BIT07!BIT06),R5
```

```
                                                BNE     6$                      ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
                                                MOVB    #1,DRVSTA(R1)           ;SET DRIVE STATUS TO ONLINE
                                        6$:     TST     (R0)+                   ;STEP OVER THE ERROR RETURN
                                                BR      8$                      ;EXIT
                                        7$:     ASL     R1                      ;CHANGE INDEX TO ADDRESS WORDS
                                                MOV     #2000.,TIMER(R1)        ;START 2 SEC TIMER
                                                ASR     R1                      ;RESTORE R1
                                                MOVB    #-1,DPINT(R1)           ;SET PORT INITIALIZE INDICATOR
                                        8$:     MOV     (SP)+,R5                ;RESTORE R5
                                                RTS     R0                      ;EXIT

                                        ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
                                        ;
                                        ;CALL
                                        ;
                                        ;       JSR     R0,@#RPC4               ;CALL THE RPC4/5/6 DRIVER
                                        ;       PNTADR                          ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
                                        ;       RETURN1                         ;RETURN HERE IF QUEUE IS FULL
                                        ;       RETURN2                         ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
                                        ;                                       ;IS AN ERROR CONDITION

                                        RPC4:   MOV     @#PS,-(SP)              ;SAVE THE CALLING STATUS
                                                MOV     RPVEC+2,@#PS            ;DON'T ALLOW ANY RPC4/5/6 INTERRUPTS
                                                MOVB    #1,ACTDRV               ;SET "ACTIVE DRIVER" FLAG
                                                SAVREG                          ;SAVE R0 - R5
                                                MOV     (R0),R2                 ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
                                                CLR     16(R2)                  ;CLEAR THE STATUS/ERROR INDICATOR
                                                MOVB    (R2),R1                 ;PICKUP THE DRIVE NUMBER
                                                MOV     RPADR,R4                ;UNIBUS ADDRESS OF RPCS1
                                                TSTB    DRVSTA(R1)              ;CHECK DRIVES STATUS
                                                BGT     1$                      ;BRANCH IF ONLINE
                                                TSTB    ULDFLG(R1)              ;UNLOAD COMMAND IN QUEUE?
                                                BNE     3$                      ;BRANCH IF YES
                                                TSTB    DPINT(R1)               ;TRYING TO INIT THE DRIVE
                                                BNE     5$                      ;BR IF YES
                                                JSR     R0,DRVINT               ;GO INIT. THE DRIVE
                                                BR      4$                      ;ERROR RETURN
                                                TSTB    DRVSTA(R1)              ;IS DRIVE STATUS ONLINE?
                                                BLE     6$                      ;BR IF NOT
                                        1$:     TSTB    DPRQS(R1)               ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
                                                BNE     5$                      ;BR IF YES
                                                MOV     R1,RPCS2(R4)            ;SELECT THE DRIVE
                                                JSR     R0,DRVQUE               ;PUT THIS REQUEST IN QUEUE
                                                BR      9$                      ;QUEUE IS FULL
                                                CMPB    #103.,2(R2)             ;IS THIS REQ. FOR AN UNLOAD?
                                                BNE     2$                      ;BR IF NO
                                                MOVB    #-1,ULDFLG(R1)          ;SET THE "UNLOAD IN QUEUE" FLAG
                                        2$:     TSTB    DRVACT(R1)              ;IS THIS DRIVE ACTIVE?
                                                BNE     8$                      ;BR IF YES
                                                JSR     PC,OPT                  ;CALL THE OPTIMIZER
                                                BR      8$
                                        3$:     MOV     #BIT15!BIT13,16(R2)     ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
                                                BR      8$                      ;EXIT
                                        4$:     JSR     PC,DI7                  ;GO HANDLE THE PARITY ERROR
```

```
                                        BR      9$
                                5$:     JSR     R0,DRVQJE       ;PUT REQUEST IN QUEUE
                                        BR      9$              ;QUEUE IS FULL
                                        BIT     #BIT06,(R4)     ;IS 'IE' SET ALREADY ?
                                        BNE     8$              ;BR IF IT IS
                                        JSR     PC,SET.IE       ;SET INTERRUPT
                                        BR      6$              ;RETURN, REQUEST IN QUELE
                                6$:     TSTB    DRVSTA(R1)      ;SEE IF DRIVE OFFLINE OR UNSAFE
                                        BLT     7$              ;BR IF UNSAFE
                                        MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE ERROR INDICATOR
                                        TSTB    DRVTYP(R1)      ;SEE IF OFFLINE OR NONEXISTENT
                                        BNE     8$              ;BR IF OFFLINE
                                        MOV     #BIT15!BIT01,16(R2)  ;REPORT DRIVE NONEXISTENT
                                        BR      8$              ;GO TO EXIT
                                7$:     MOV     #BIT15!BIT12,16(R2)  ;DRIVE IS UNSAFE
                                8$:     RESREG                  ;RESTORE R0 - R5
                                        TST     (R0)+           ;SETUP FOR NORMAL RETURN
                                        BR      10$             ;FINISH UP, THEN EXIT
                                9$:     RESREG                  ;RESTORE R0 - R5
                                10$:    TST     (R0)+           ;CORRECT THE RETURN ADDRESS
                                        CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
                                        MOV     (SP)+,@#PS      ;RETURN "PS" TO USER LEVEL
                                        RTS     R0              ;RETURN TO CALLER

                                ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
                                ;
                                ;CALL
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                ;       JSR     PC,OPT          ;SETUP A COMMAND
                                ;
                                OPT:    SAVREG                  ;SAVE R0 - R5
                                        MOV     @#PS,-(SP)      ;SAVE PROC. STATUS
                                        BICB    ATABIT(R1),SRCHW?       ;CLEAR "SEARCH WAIT" KEY
                                        JSR     PC,GETREQ       ;GET "DPB" POINTER OF REQUEST
                                        TST     R2              ;IS THERE A REQUEST IN QUEUE?
                                        BEQ     7$              ;NO--BRANCH TO EXIT
                                        BIT     #BIT12,RPDS1(R4)  ;IS MOL STILL SET ?
                                        BEQ     9$              ;BR IF NOT
                                        BIT     #BIT6,RPDS1(R4)  ;IS VV SET ?
                                        BNE     10$             ;BR IF IT IS
                                3$:     JSR     R0,DRVINT       ;SEE IF DRIVE STILL ONLINE ?
                                        BR      6$              ;PARITY OR 'DVA' NOT SET
                                10$:    TSTB    DRVSTA(R1)      ;IS DRIVE ONLINE?
                                        BGT     1$              ;YES--BRANCH
                                        JSR     PC,POPQUE       ;NO--REMOVE REQUEST FROM QUELE
                                        MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE STATUS ERROR INDICATOR
                                        TSTB    DRVSTA(R1)      ;IS DRIVE UNSAFE ?
                                        BPL     8$              ;BR TO EXIT IF NOT
                                        MOV     #BIT15!BIT12,16(R2)  ;SET UNSAFE STATUS ERROR INDICATOR
                                        BR      8$              ;BRANCH TO EXIT
                                1$:     MOV     #111,-(SP)      ;LOAD COMMAND ONTO THE STACK
                                        JSR     R0,WRT.RP       ;LOAD THE REGISTER
                                        RPCS1                   ;REGISTER INCREMENT
                                        6$                      ;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 RFQUEST 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  016C34                    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"
```

```
2698  012776  001012                    BNE     2$              ;BRANCH IF ANY 'ATA' BITS SET
2699  013000  004037  014636            JSR     R0,RD.RP        ;READ CONTROL AND STATUS REGISTER
2700  013004  000000                    RPCS1
2701  013006  012174                    CIB
2702  013010  106126                    ROLB    (SP)+           ;IS "IE"=1?
2703  013012  100403                    BMI     1$              ;YES, NO DRIVES TO CHECK
2704  013014  104001                    ERROR   1               ;REPORT AN ILLEGAL INTERRUPT
2705  013016  004737  015316            JSR     PC,SET.IE       ;SET INTERRUPT ENABLE
2706  013022  000207            1$:     RTS     PC              ;RETURN
2707  013024  005046            2$:     CLR     -(SP)           ;PROCESS ALL DRIVES THAT HAVE
2708  013026  110316                    MOVB    R3,(SP)         ;AN "ATA"=1
2709  013030  012703  000001            MOV     #1,R3
2710  013034  005001                    CLR     R1
2711  013036  030316            SC3:    BIT     R3,(SP)         ;ATA=1?
2712  013040  001005                    BNE     SC5             ;YES--BRANCH
2713  013042  005201            SC4:    INC     R1              ;MOVE TO THE NEXT DRIVE
2714  013044  106303                    ASLB    R3
2715  013046  001373                    BNE     SC3             ;BRANCH IF MORE TO CHECK?
2716  013050  005726                    TST     (SP)+           ;CLEAN OFF THE STACK
2717  013052  000207                    RTS     PC              ;RETURN TO USER
2718  013054  105761  007576    SC5:    TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
2719  013060  001402                    BEQ     1$              ;BR IF NOT
2720  013062  000137  013750            JMP     SC13            ;PROCESS THE DRIVE
2721  013066  105761  007606    1$:     TSTB    DPRQS(R1)       ;PORT REQUEST OUTSTANDING ?
2722  013072  001402                    BEQ     2$              ;BR IF NOT
2723  013074  000137  013750            JMP     SC13            ;START THE OUTSTANDING COMMAND
2724  013100  105761  007556    2$:     TSTB    DRVSTA(R1)      ;CHECK THE DRIVE STATUS
2725  013104  003025                    BGT     5$              ;BRANCH IF ONLINE
2726  013106  105761  007624            TSTB    ULDFLG(R1)      ;UNLOAD IN PROGRESS?
2727  013112  003422                    BLE     5$              ;BRANCH IF NOT
2728  013114  004737  016034            JSR     PC,GETREQ       ;GET DPB POINTER
2729  013120  004737  015200            JSR     PC,SVRH11       ;SAVE THE RH11/RP04/5/6 REGISTERS
2730  013124  004737  013700            JSR     PC,SC12         ;SAVE RPDS1, RPER1, RPER2, AND RPER3
2731                                                            ;ALSO DO A DRIVE INIT (DRVINT)
2732  013130  105761  007556            TSTB    DRVSTA,R1       ;DID DRIVE COME ONLINE?
2733  013134  003416                    BLE     6$              ;NO---BRANCH
2734  013136  032737  040000  007536    BIT     #BIT14,RPERRS   ;WAS THERE AN ERROR?
2735  013144  001002                    BNE     3$              ;BR IF ERROR
2736  013146  000137  013610            JMP     SC11            ;NO ERROR
2737  013152  013705  007540    3$:     MOV     RPERRS+2,R5     ;YES -- PICKUP RPER1 AND
2738  013156  000476                    BR      SC6A            ;GO PROCESS THE ERROR
2739  013160  105761  007546    5$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
2740  013164  001027                    BNE     SC6             ;BR IF EITHER
2741  013166  004737  013700            JSR     PC,SC12         ;SAVE RPDS1, RPER1, RPER2, AND RPER3
2742                                                            ;ALSO DO A DRVINT
2743  013172  105761  007576    6$:     TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE ?
2744  013176  001321                    BNE     SC4             ;BR IF YES, CHECK ON MORE DRIVES
2745  013200  105761  007556            TSTB    DRVSTA(R1)      ;CHECK ON DRIVE'S STATUS
2746  013204  100412                    BMI     7$              ;BR IF UNSAFE
2747  013206  032737  020000  007544    BIT     #BIT13,RPERRS+6 ;ADDRESS PLUG CHANGED ?
2748  013214  001011                    BNE     8$              ;BR IF YES
2749  013216  012746  000113            MOV     #113,-(SP)      ;RELEASE COMMAND
2750  013222  004037  015012            JSR     R0,SET.GP       ;WRITE THE COMMAND INTO RPCS1
2751  013226  000000                    RPCS1                   ;REGISTER INDEX
```

```
                                ;PARITY EXIT ADDRESS
                        7$:     MOV     (SP),R5         ;PICKUP (R5)AS  BEFORE THE ERROR CALL
                                ERROR   3               ;REPORT THE UNEXPECTED ATTENTION
                                BR      SC4             ;GO CHECK FOR MORE ATA'S
                        8$:
                                ERROR   5               ;REPORT THE ADDRESS PLUG CHANGE
                                BR      SC4             ;CHECK FOR MORE DRIVES
                        SC6:    MOV     R1              ;SETUP TO ADDRESS WORDS
                                MOV     #-1,TIMER(R1)   ;STOP THE TIMER
                                MOVB    R1              ;RESTORE THE DRIVE ADDRESS
                                JSR     PC,GETREG       ;GET THE DPB POINTER FROM THE QUEUE
                                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
                                JSR     RC,RD.RP        ;READ THE RP04'S STATUS REG.
                                RPCS1
                                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
                                BEQ     SC11            ;BR IF DRIVE ACTIVE WITH ORDER
                                BIS     #BIT15!BIT07!BIT03,16(R2)  ;INFORM USER OF ERROR RECOVER COMPLETION
                                BR      SC7
                        1$:     JSR     RC,RD.RP        ;READ ERROR REGISTER #1
                                RPER1
                                SC9
                                MOV     (SP)+,R5        ;AND SAVE IT IN R5
                                JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
                                MOV     #11.,-(SP)      ;ISSUE A DRIVE CLEAR
                                JSR     RC,WRT.RP
                                RPCS1
                                SC8
                        SC6A:   ROL     R5              ;WAS "UNSAFE" CONDITION =1?
                                BMI     1$              ;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$:     JSR     RC,RD.RP        ;READ DRIVE STATUS REG. #1
                                RPDS1
                                SC8
                                MOV     (SP),R5         ;SAVE RPDS1 IN R5
                                ROL     (SP)+           ;"ERR"=1?
                                BPL     2$              ;BR IF NO--UNSAFE CLEARED
                                MOVB    #-1,DRVSTA(R1)  ;DRIVE IS UNSAFE
                                JSR     PC,SVRH11       ;SAVE RH11/RP04/5 & 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
```

```
013470  000137  013042                SC4:    JMP     SC4
013474  052762  100220  000016        3$:     BIS     #BIT15!BIT07!BIT04,16(R2)       ;INFORM USER OF ERROR
013502  105061  007546                SC7:    CLRB    DRVACT(R1)              ;DRIVE IS IDLE
013506  004737  015240                        JSR     PC,EMPTYQ               ;DUMP THE QUEUE
013512  105761  007624                        TSTB    ULDFLG(R1)              ;UNLOAD IN PROGRESS OR QUEUE?
013516  003002                                BGT     1$                      ;BR IF NOT
013520  105061  007624                        CLRB    ULDFLG(R1)              ;CLEAR UNLOAD FLAG
013524  116164  007672  000016        1$:     MOVB    ATABIT(R1),RPAS(R4)     ;CLEAR ATTENTION BIT
013532  105761  007556                        TSTB    DRVSTA(R1)              ;IS THE DRIVE UNSAFE ?
013536  100406                                BMI     2$                      ;BR IF IT IS
013540  012746  000113                        MOV     #113,-(SP)              ;RELEASE COMMAND
013544  004037  015012                        JSR     R0,WRT.RP               ;WRITE THE COMMAND INTO RPCS1
013550  000000                                RPCS1                           ;REGISTER INDEX
013552  013560                                SC8                             ;PARITY EXIT ADDRESS
013554  000137  013042                2$:     JMP     SC4                     ;CHECK FOR MORE DRIVES
013560  105761  007546                SC8:    TSTB    DRVACT(R1)              ;IS DRIVE IDLE?
013564  001405                                BEQ     1$                      ;YES--BRANCH
013566  004737  016034                        JSR     PC,GETREQ               ;GET DPB POINTER
013572  004737  012074                        JSR     PC,CI7                  ;PROCESS THE PARITY ERROR
013576  000402                                BR      2$                      ;CONTINUE
013600  004737  012122                1$:     JSR     PC,CI7B                 ;PROCESS THE UNCORRECTABLE PARITY ERROR
013604  000137  013042                2$:     JMP     SC4                     ;CHECK MORE DRIVES
013610  105761  007624                SC11:   TSTB    ULDFLG(R1)              ;"UNLOAD IN PROGRESS"?
013614  003402                                BLE     1$                      ;BRANCH IF NO
013616  105061  007624                        CLRB    ULDFLG(R1)              ;CLEAR UNLOAD FLAG
013622  105061  007546                1$:     CLRB    DRVACT(R1)              ;SET DRIVE IDLE
013626  136137  007672  007620                BITB    ATABIT(R1),SRCHW!       ;DOING A SEARCH OPERATION FOR
                                                                              ;AN I/O COMMAND?
013634  001012                                BNE     2$                      ;BRANCH IF YES
013636  004737  016056                        JSR     PC,POPQUE               ;REMOVE REQUEST FROM QUEUE
013642  052762  000200  000016                BIS     #BIT07,16(R2)           ;SET "DONE" BIT
013650  005737  007644                        TST     SAVEFG                  ;SAVE THE REGISTERS?
013654  100002                                BPL     2$                      ;BRANCH IF NO
013656  004737  015200                        JSR     PC,SVRH11               ;YES--SAVE ALL OF THE RH11 RP04 5 6 REGS
013662  116164  007672  000016        2$:     MOVB    ATABIT(R1),RPAS(R4)     ;CLEAR ATTENTION BIT
013670  004737  010764                        JSR     PC,OPT                  ;START A REQUEST
013674  000137  013042                        JMP     SC4                     ;CHECK FOR MORE DRIVES
013700  010164  000010                SC12:   MOV     R1,RPCS2(R4)            ;SELECT DRIVE
013704  016437  000012  007536                MOV     RPDS1(R4),RPERRS        ;SAVE THE FOUR REGISTERS THAT
013712  016437  000014  007540                MOV     RPER1(R4),RPERRS+2      ;WILL TELL US SOMETHING
013720  016437  000040  007542                MOV     RPER2(R4),RPERRS+4
013726  016437  000042  007544                MOV     RPER3(R4),RPERRS+6
013734  004037  010134                        JSR     R0,DRVINT               ;INIT. THE STATE OF THE DRIVE
013740  000401                                BR      1$                      ;TAKE ERROR EXIT
013742  000207                                RTS     PC                      ;RETURN
013744  005726                        1$:     TST     (SP)+                   ;POP PC OFF OF THE STACK
013746  000704                                BR      SC8                     ;PROCESS THE PARITY ERROR
013750  006301                        SC13:   ASL     R1                      ;SETUP TO ADDRESS WORDS
013752  012761  177777  007650                MOV     #-1,TIMER(R1)           ;STOP THE TIMER
013760  006201                                ASR     R1                      ;
013762  010164  000010                        MOV     R1,RPCS2(R4)            ;SELECT THE DRIVE
013766  116164  007672  000016                MOVB    ATABIT(R1),RPAS(R4)     ;CLEAR THE ATTENTION BIT
013774  032714  004000                        BIT     #BIT11,(R4)             ;DRIVE AVAILABLE ?
014000  001006                                BNE     1$                      ;BR IF AVAILABLE
```

```
                                                ASL     R1
                                                MOV     #10000.,TIMER(R1)  ;START 10 SEC TIMER AGAIN
                                                ASR     R1
                                                BR      3$                 ;EXIT
007576                                  1$:     TSTB    DPINT(R1)          ;INITIALIZING THE DRIVE ?
                                                BEQ     2$                 ;BR IF NOT
007576                                          CLRB    DPINT(R1)          ;CLEAR THE INIT INDICATOR
                                                JSR     R0,DRVINT          ;GO INIT THE DRIVE
                                                NOP                        ;DUMMY PARITY ERROR RETURN
007556                                          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
000016                                          BIS     #BIT15!BIT14,16(R2)  ;INFORM USER THAT DRIVE OFFLINE
                                                JSR     PC,S.RH11          ;SAVE THE REGISTERS
                                                JSR     PC,EMPTYQ          ;EMPTY THE REQUEST QUEUE
                                                BR      3$
007606                                  2$:     CLRB    DPRQS(R1)          ;CLEAR THE PORT REQUEST INDICATOR
                                                JSR     PC,SPT             ;START THE PENDING REQUEST
                                                JMP     SC4                ;PROCESS OTHER DRIVES

                                        ;RPC4 5 6 TIMER ROUTINE
                                        ;CALL
                                        ;       MOV     #TIME,-(SP)        ;ELAPSED TIME IN MILLISECONDS ON THE STACK
                                        ;       JSR     PC,RPTMR           ;CALL RPC4/5/6 TIME ROUTINE

007623                                  RPTMR:  TST     ACTDRV             ;CHECK "ACTDRV & ACTSTR"
                                                BNE     4$                 ;IF NON ZERO EXIT
000001  007623                                  MOVB    #1,ACTSTR          ;SET "ACTSTR"
                                                SAVREG                     ;SAVE R0 - R5
                                                CLR     R1                 ;START WITH DRIVE 0
                                                CLR     R3
007650                                  1$:     TST     TIMER(R3)          ;IS THE TIMER RUNNING?
                                                BLT     2$                 ;BRANCH IF NO
000002  007650                                  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$:     INC     R1                 ;MOVE TO NEXT DRIVE
                                                TST     (R3)+
000010                                          CMP     #8.,R1             ;OUT OF DRIVES?
                                                BGT     1$                 ;BRANCH IF NO
                                        3$:     RESREG                     ;RESTORE R0 - R5
007623                                          CLRB    ACTSTR             ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
                                        4$:     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
014202  010146                          STO:    MOV     R1,-(SP)        ;SAVE R1
014204  010346                                  MOV     R3,-(SP)        ;SAVE R3
014206  013704  007704                          MOV     RPADR,R4        ;GET ADDRESS OF "RPCS1"
014212  010164  000010                          MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
014216  004037  014636                          JSR     R0,RD.RP        ;READ "DRIVE STATUS REG"
014222  000012                                  RPDS1
014224  014524                                  STDS
014226  105726                                  TSTB    (SP)+           ;IS "DRY"=1?
014230  100477                                  BMI     STO2            ;BR IF YES
014232  105761  007576                  STO1:   TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
014236  001074                                  BNE     STO2            ;BR IF YES
014240  105761  007606                          TSTB    CPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
014244  001071                                  BNE     STO2            ;BR IF YES
014246  013702  007616                          MOV     TRNSWT,R2       ;PICKUP TRANSFER WAIT QUEUE
014252  020137  007670                          CMP     R1,DTUW         ;TRANSFER UNDERWAY ON THIS DRIVE?
014256  001402                                  BEQ     1$              ;BRANCH IF YES
014260  004737  016034                          JSR     PC,GETREQ       ;GET DPB ADDRESS
014264  052762  101000  000016          1$:     BIS     #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
014272  004737  015200                          JSR     PC,S.RH11       ;SAVE RH11/RP04,5/6 REGISTERS
014276  012764  000040  000010                  MOV     #BIT05,RPCS2(R4) ;"INIT" THE MASS BUS
014304  105061  007546                          CLRB    DRVACT(R1)      ;DRIVE IS IDLE
014310  105061  007624                          CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
014314  005001                                  CLR     R1              ;START WITH DRIVE 0
014316  005003                                  CLR     R3
014320  004037  010134                  2$:     JSR     R0,DRVINT       ;INIT. THIS DRIVE
014324  000477                                  BR      ST05            ;PARITY ERROR RETURN
014326  105761  007546                          TSTB    DRVACT(R1)      ;DRIVE IDLE BEFORE THE INIT.?
014332  001414                                  BEQ     4$              ;YES--BRANCH
014334  013702  007616                          MOV     TRNSWT,R2       ;GET TRANSFER WAIT QUEUE
014340  023701  007670                          CMP     DTUW,R1         ;WAS THERE I/O ON THIS DRIVE?
014344  001402                                  BEQ     3$              ;YES--BRANCH
014346  004737  016034                          JSR     PC,GETREQ       ;GET THE DPB POINTER FROM QUEUE
014352  052762  100400  000016          3$:     BIS     #BIT15!BIT08,16(R2) ;INFORM USER OF INIT.
014360  105061  007546                          CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE TO IDLE
014364  105061  007624                  4$:     CLRB    ULDFLG(R1)      ;NO UNLOAD
014370  012763  177777  007650                  MOV     #-1,TIMER(R3)   ;STOP THE TIMER
014376  005723                                  TST     (R3)+           ;UPDATE THE INDEX
014400  005201                                  INC     R1              ;INCREMENT THE DRIVE NUMBER
014402  022701  000010                          CMP     #8.,R1          ;LAST DRIVE BEEN CHECKED?
014406  003344                                  BGT     2$              ;NO--LOOP
014410  012737  177777  007670                  MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
014416  005037  007616                          CLR     TRNSWT          ;CLEAR TRANSFER WAIT QUEUE
014422  004737  015662                          JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
014426  000500                                  BR      ST09            ;EXIT
014430  116405  000016                  ST02:   MOVB    RPAS(R4),R5     ;READ ATTENTION REG
014434  136105  007672                          BITB    ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
014440  001017                                  BNE     ST03            ;YES--BRANCH
014442  105761  007576                          TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
014446  001031                                  BNE     ST06            ;BR IF YES - DRIVE NOT ONLINE
014450  105761  007606                          TSTB    CPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
```

```
                                                BNE     STO7            ;BR IF YES - NO RESPONSE TO REQUEST
                                                CMP     R1,DTUW         ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
                                                BNE     STO1            ;BR IF NO
                                                JSR     R0,RD.RP        ;YES--CHECK "RDY"
                                                RPCS1
                                                STC5
                                                TSTB    (SP)+
                                                BPL     STO1            ;BR IF "RDY"=0
                                        ST03:   TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
                                                BNE     1$              ;BR IF INIT PENDING
                                                TSTB    DPRQS(R1)       ;PORT REQUEST PENDING ?
                                                BEQ     ST09            ;BR IF NOT
                                        1$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                                BR      ST09            ;EXIT
                                        ST05:   JSR     PC,CIB          ;GO HANDLE THE PARITY ERROR
                                                BR      ST09
                                        ST06:   CLRB    DPINT(R1)       ;CLEAR THE INITIALIZE INDICATOR
                                                CLRB    DRVSTA(R1)      ;SET UNIT OFFLINE
                                                MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                                JSR     PC,GETREQ       ;GET THE DPB ADDRESS
                                                TST     R2              ;REQUEST IN QUEUE ?
                                                BEQ     ST09            ;BR IF NOT
                                                BIS     #BIT15!BIT14,16(R2)  ;INFORM THE USER DRIVE NOT AVAILABLE
                                                BR      ST08            ;FINISH
                                        ST07:   MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                                CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST INDICATOR
                                                JSR     PC,GETREQ       ;GET DPB ADDRESS
                                                TST     R2              ;QUEUE ENTRY FOR DRIVE ?
                                                BEQ     ST09            ;BR IF NONE
                                                MOV     #BIT15!BIT2,16(R2)  ;INFORM USER OF PORT REQUEST ERROR
                                        ST08:   JSR     PC,EMPTYQ       ;CLEAR THE QUEUE FOR THE DRIVE
                                                JSR     PC,SVRH11       ;SAVE THE REGISTERS
                                        ST09:   MOV     (SP)+,R3        ;RESTORE R3
                                                MOV     (SP)+,R1        ;RESTORE R1
                                                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
                                        ;
                                        RD.RP:  MOV     MCPEMX,RD.RP2   ;MAX. RETRYS ALLOWED
                                                MOV     (SP),-(SP)      ;SAVE R0 FOR RETURN
                                                MOV     RPADR,RD.ADR    ;FORM THE DESIRED ADDRESS
                                                ADD     (R0)+,RD.ADR    ;USING THE BASE AND THE INDEX
                                        RD.RP1: MOV     @(PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RP04
                                        RD.ADR: .WORD   0               ;ADDRESS IS FORMED HERE
                                        RD.WRD: .WORD   0               ;REG. CONTENTS PUT HERE
                                                MOV     RD.WRD,2(SP)    ;RETURN IT TO THE USER
                                                MOV     RPADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
```

```
                                                ADD     #RPCS2,(SP)     ;FORM THE ADDRESS OF RPCS2
                                                BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
                                                BNE     RD.RP3          ;BR IF DRIVE NON-EXISTENT
                                                MOV     @RPADR,-(SP)    ;READ RPCS1
                                                BIT     #BIT13,(SP)     ;DID MCPE SET?
                                                BNE     1$              ;BRANCH IF YES
                                                CMP     (SP)+,(R0)+     ;ADJUST FOR RETURN
                                                BR      RD.RP4          ;EXIT
                                        1$:
                                                ERROR   3               ;REPORT "MCPE" ERROR
                                                TST     DTUW            ;DATA TRANSFER UNDERWAY?
                                                BMI     2$              ;NO--BRANCH
                                                BIT     #BIT14,(SP)     ;NO--"TRE"=1?
                                                BEQ     2$              ;NO--BRANCH
                                                TST     (SP)+           ;YES--CLEAN OFF THE STACK AND
                                                BR      RD.RP3          ;TAKE THE FATAL ERROR EXIT
                                        2$:     BIS     #BIT14,(SP)     ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
                                                SWAB    (SP)            ;POSITION BEFORE WRITING
                                                MOV     RPADR,3$        ;FORM ADDRESS OF HIGH BYTE
                                                INC     3$
                                                MOVB    (SP)+,@(PC)+    ;WRITE THE HIGH BYTE OF RPCS1
                                        3$:     .WORD   0               ;ADDRESS STORAGE
                                                DEC     (PC)+           ;EXCEEDED MAX. RETRYS
                                        RD.RP2: .WORD   3
                                                BGE     RD.RP1          ;BRANCH IF NO
                                        RD.RP3: MOV     (R0),R0         ;FATAL ERROR EXIT
                                                MOV     (SP)+,(SP)
                                        RD.RP4: RTS     R0

                                        ;ROUTINE TO WRITE A REGISTER
                                        ;
                                        ;CALL
                                        ;       MOV     DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
                                        ;       JSR     R0,WRT.RP       ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
                                        ;       INDEX                   ;INDEX OF THE REGISTER TO BE LOADED
                                        ;       ERRADR                  ;ADDRESS TO RETURN TO ON AN ERROR
                                        ;       RETURN                  ;ERROR FREE RETURN
                                        ;

015012  013737  007702  015162  WRT.RP: MOV     MCPEMX,WRT.R2   ;MAX RETRYS ALLOWED
015020  016637  000002  015130          MOV     2(SP),WRT.WD    ;SAVE THE WORD TO WRITE
015026  012616                          MOV     (SP)+,(SP)      ;ADJUST THE STACK
015030  012037  015132                  MOV     (R0)+,WRT.AD    ;GET INDEX OF REGISTER TO BE WRITTEN
015034  001015                          BNE     1$              ;BRANCH IF NOT RPCS1
015036  122737  000150  015100          CMPB    #150,WRT.WD     ;IS THE COMMAND FOR DATA TRANSFERS?
015044  002411                          BLT     1$              ;YES--DON'T GET THE OLD A16 & A17, & PSEL
015046  004037  014636                  JSR     R0,RD.RP        ;NO---COMBINE A16&A17, & PSEL WITH
015052  000000                          RPCS1                   ;THE COMMAND BEFORE SENDING IT TO
015054  015170                          WRT.R3                  ;THE RH11/RP04
015056  000316                          SWAB    (SP)
015060  042716  177770                  BIC     #^C7,(SP)
015064  112637  015101                  MOVB    (SP)+,WRT.WD+1
015070  063737  007704  015102  1$:     ADD     RPADR,WRT.AD    ;FORM THE ADDRESS OF THE DISK REG.
015076  012737                  WRT.R1: MOV     (PC)+,@(PC)+    ;LOAD THE DESIRED REG.
015102  000000                  WRT.WD: .WORD   0               ;WORD TO WRITE GOES HERE
```

```
 3056  015102  000000                    .RT.AD: .WORD   0               ;ADDRESS IS FORMED HERE
 3057  015104  013746  007704                    MOV     RPADR,-(SP      ;PUT THE ADDRESS ON THE STACK
 3058  015110  062716  000010                    ADD     #RPCS2,(SP      ;FORM THE ADDRESS OF RPCS2
 3059  015114  032736  010000                    BIT     #BIT12,2(SP)+   ;CHECK THE 'NED' BIT
 3060  015120  001023                            BNE     WRT.R3          ;BR IF DRIVE NON-EXISTENT
 3061  015122  004037  014636                    JSR     R0,RD.RP        ;CHECK FOR PARITY ERROR ON WRITE
 3062  015126  000014                            RPER1
 3063  015130  015170                            WRT.R3
 3064  015132  032726  000010                    BIT     #BIT03,(SP)+
 3065  015136  001416                            BEQ     WRT.R4          ;BRANCH IF "PAR=0"
 3066  015140  016037  177776  015152            MOV     -2(R0),1$       ;PICKUP THE INDEX
 3067  015146  004037  014636                    JSR     R0,RD.RP        ;READ THE REG.
 3068  015152  000000                    1$:     .WORD   0               ;REG. INDEX
 3069  015154  015170                            WRT.R3                  ;RETURN TO THIS ADDRESS ON ERROR
 3070  015156  104004                            ERROR   4               ;REPORT THE PARITY ON WRITE ERROR
 3071  015160  005327                            DEC     (PC)+           ;DECREMENT THE ERROR COUNT
 3072  015162  000003                    WRT.R2: .WORD   3               ;RETRY COUNTER
 3073  015164  002344                            BGE     WRT.R1          ;TRY AGAIN IF NOT FINISHED
 3074  015166  005726                            TST     (SP)+           ;CLEAN OFF THE STACK
 3075  015170  011000                    WRT.R3: MOV     (R0),R0         ;TAKE THE "PARITY ON WRITE" ERROR EXIT
 3076  015172  000401                            BR      WRT.R5          ;EXIT
 3077  015174  005720                    WRT.R4: TST     (R0)+           ;ADJUST FOR ERROR FREE EXIT
 3078  015176  000200                    WRT.R5: RTS     R0
 3079
 3080                                    ;ROUTINE TO SAVE THE RH11/RP04/5 6 REGISTERS AS PER DPB+14
 3081                                    ;
 3082                                    ;CALL
 3083                                    ;       MOV     #DPBNUM,R2      ;DPB POINTER TO R2
 3084                                    ;       JSR     PC,SVRH11       ;SAVE THE DRIVES REG'S
 3085
 3086  015200  104412                    SVRH11: SAVREG                  ;SAVE R0 - R5
 3087  015202  005702                            TST     R2              ;QUEUE ENTRY FOR THE DRIVE ?
 3088  015204  001430                            BEQ     4$              ;BR IF NONE
 3089  015206  013704  007704                    MOV     RPADR,R4
 3090  015212  111264  000010                    MOVB    (R2),RPCS2(R4)  ;SELECT DRIVE
 3091  015216  016203  000014                    MOV     14(R2),R3       ;GET THE ERROR TABLE POINTER
 3092  015222  001433                            BEQ     6$              ;EXIT IF NO ADDRESS
 3093  015224  005037  015260                    CLR     3$              ;COUNTER & POINTER
 3094  015230  023727  015260  000022    1$:     CMP     3$,#RPDB        ;REACHED THE BUFFER REGISTER ?
 3095  015236  001006                            BNE     2$              ;BR IF NOT
 3096  015240  032764  000200  000010            BIT     #BIT07,RPCS2(R4)  ;'CR' SET ?
 3097  015246  001002                            BNE     2$              ;BR IF SET
 3098  015250  005023                            CLR     (R3)+           ;STORE RPDB AS ZEROES
 3099  015252  000405                            BR      4$              ;CONTINUE
 3100  015254  004037  014636            2$:     JSR     R0,RD.RP        ;READ THE SELECTED REGISTER
 3101  015260  000000                    3$:     .WORD   0               ;REGISTER INDEX
 3102  015262  015306                            5$                      ;ERROR RETURN ADDRESS
 3103  015264  012623                            MOV     (SP)+,(R3)+     ;STORE THE REGISTER CONTENTS
 3104  015266  023727  015260  000046    4$:     CMP     3$,#RPEC2       ;REACHED THE END ?
 3105  015274  001406                            BEQ     6$              ;BR IF YES
 3106  015276  062737  000002  015260            ADD     #2,3$           ;INCREMENT THE REGISTER INDEX
 3107  015304  000751                            BR      1$              ;CONTINUE READING THE REGISTERS
 3108  015306  004737  012074            5$:     JSR     PC,???          ;PROCESS THE UNCORRECTABLE PARITY ERROR
 3109  015312  104413                    6$:     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  015430          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                                  ;       RETURN1                 ;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
```

```
3292  016076  026161  015420  015442          CMP     QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
3293  016104  001003                          BNE     1$                  ;NO--BRANCH TO EXIT
3294  016106  016161  015440  015420          MOV     QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
3295  016114  006201                  1$:     ASR     R1
3296  016116  000207                          RTS     PC                  ;RETURN TO USER
3297
3298
3299
3300                                  ;;*****************************************************************
3301
3302                                  .SBTTL  DATA PARAMETER BLOCK
3303
3304                                  ;;*****************************************************************
3305
3306  016120     000                  DPB:    .BYTE 0                     ;DRIVE NUMBER
3307  016121     000                          .BYTE 0                     ;OFFSET VALUE OR FMT22, ECI, AND HCI
3308  016122     000                          .BYTE 0                     ;COMMAND
3309  016123     000                          .BYTE 0                     ;PSEL AND A17 AND A16
3310  016124  000000                          .WORD 0                     ;WORD COUNT (MUST BE NEG)
3311  016126  001206                          .WORD INREG                 ;BUFFER ADDRESS OR
3312                                                                      ;REGISTER TABLE POINTER
3313  016130     000                          .BYTE 0                     ;SECTOR ADDRESS OR
3314                                                                      ;FIRST REGISTER INDEX
3315  016131     000                          .BYTE 0                     ;TRACK ADDRESS OR
3316                                                                      ;LAST REGISTER INDEX
3317  016132  000000                          .WORD 0                     ;CYLINDER ADDRESS
3318  016134  016140                          .WORD REG                   ;ERROR TABLE POINTER
3319                                                                      ;POINTS TO THE FIRST OF TWENTY
3320                                                                      ;LOCATIONS WHERE THE DRIVER IS
3321                                                                      ;TO STORE THE RH11/RP04 REGISTERS
3322                                                                      ;ON AN ERROR.  IF ZERO, REGISTERS
3323                                                                      ;ARE NOT SAVED.
3324  016136  000000                          .WORD 0                     ;STATUS/ERROR INDICATOR
3325                                                                      ;BIT15 = 1 => ERROR OCCURRED
3326                                                                      ;BIT07 = 1 => DONE
3327                                                                      ;BIT 14 - BIT10 AND BIT06 - BIT03
3328                                                                      ;INDICATE TYPE OF ERROR
3329
3330  016140  000000          REG:    .WORD   0                           ;STORE RP04 REGISTERS HERE
3331  016142  000000                  .WORD   0                           ;RPWC
3332  016144  000000                  .WORD   0                           ;RPBA
3333  016146  000000                  .WORD   0                           ;RPDA
3334  016150  000000                  .WORD   0                           ;RPCS2
3335  016152  000000                  .WORD   0                           ;RPDS1
3336  016154  000000                  .WORD   0                           ;RPER1
3337  016156  000000                  .WORD   0                           ;RPAS
3338  016160  000000                  .WORD   0                           ;RPLA
3339  016162  000000                  .WORD   0                           ;RPDB
3340  016164  000000                  .WORD   0                           ;RPMR
3341  016166  000000                  .WORD   0                           ;RPDT
3342  016170  000000                  .WORD   0                           ;RPSN
3343  016172  000000                  .WORD   0                           ;RPOF
3344  016174  000000                  .WORD   0                           ;RPCA
3345  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 & 809
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
```

```
016312  000022                  .WORD   18.
016314  100000                  .WORD   100000
016316  000000                  .WORD   0
016320  000022                  .WORD   18.
016322  100000                  .WORD   100000                  ;TERMINATOR

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

                        .SBTTL  OFFSET CODE TABLES

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

016324                  OFTBL6:                                 ;OFFSET TABLE FOR CYLINDER 496
016324     030                  .BYTE   30
016325     027                  .BYTE   27
016326     026                  .BYTE   26
016327     025                  .BYTE   25
016330     024                  .BYTE   24
016331     023                  .BYTE   23
016332     022                  .BYTE   22
016333     021                  .BYTE   21
016334     020                  .BYTE   20
016335     017                  .BYTE   17
016336     016                  .BYTE   16
016337     015                  .BYTE   15
016340     014                  .BYTE   14
016341     013                  .BYTE   13
016342     012                  .BYTE   12
016343     011                  .BYTE   11
016344     010                  .BYTE   10
016345     007                  .BYTE   7
016346     006                  .BYTE   6
016347     005                  .BYTE   5
016350     004                  .BYTE   4
016351     003                  .BYTE   3
016352     002                  .BYTE   2
016353     001                  .BYTE   1
016354     000                  .BYTE   0
016355     201                  .BYTE   201
016356     202                  .BYTE   202
016357     203                  .BYTE   203
016360     204                  .BYTE   204
016361     205                  .BYTE   205
016362     206                  .BYTE   206
016363     207                  .BYTE   207
016364     210                  .BYTE   210
016365     211                  .BYTE   211
016366     212                  .BYTE   212
016367     213                  .BYTE   213
016370     214                  .BYTE   214
016371     215                  .BYTE   215
016372     216                  .BYTE   216
016373     217                  .BYTE   217
016374     220                  .BYTE   220
```

```
OFTBL4:                                 ;OFFSET TABLE FOR CYLINDER 245
```

```
016461  004     .BYTE   4
016462  003     .BYTE   3
016463  002     .BYTE   2
016464  001     .BYTE   1
016465  000     .BYTE   0
016466  201     .BYTE   201
016467  202     .BYTE   202
016470  203     .BYTE   203
016471  204     .BYTE   204
016472  205     .BYTE   205
016473  206     .BYTE   206
016474  207     .BYTE   207
016475  210     .BYTE   210
016476  211     .BYTE   211
016477  212     .BYTE   212
016500  213     .BYTE   213
016501  214     .BYTE   214
016502  215     .BYTE   215
016503  216     .BYTE   216
016504  217     .BYTE   217
016505  220     .BYTE   220
016506  221     .BYTE   221
016507  222     .BYTE   222
016510  223     .BYTE   223
016511  224     .BYTE   224
016512  225     .BYTE   225
016513  226     .BYTE   226
016514  227     .BYTE   227
016515  230     .BYTE   230
016516  231     .BYTE   231
016517  232     .BYTE   232
016520  233     .BYTE   233
016521  234     .BYTE   234
016522  235     .BYTE   235
016523  236     .BYTE   236
016524  237     .BYTE   237
016525  240     .BYTE   240
016526  241     .BYTE   241
016527  242     .BYTE   242
016530  243     .BYTE   243
016531  244     .BYTE   244
016532  245     .BYTE   245
016533  246     .BYTE   246
016534  247     .BYTE   247
016535  250     .BYTE   250
016536  251     .BYTE   251
016537  252     .BYTE   252
016540  253     .BYTE   253
016541  254     .BYTE   254
016542  255     .BYTE   255
016543  256     .BYTE   256
016544  257     .BYTE   257
016545  260     .BYTE   260
                .EVEN
```

```
                                        ;;*****************************************************************
                                        .SBTTL  MESSAGES
                                        ;;*****************************************************************

016546  020015  042515  047111  TITLE:  .ASCII  <CR><LF>/MAINDEC-11-DZRJC-A/<CR><LF>
016546  042015  042515  030461
016546  042055  051132  041512
016546  042055  020015
016546  046015  032060  032457        .ASCII  ?RP04 5 6 HEAD ALIGNMENT VERIFICATION PROGRAM? <CR><LF><LF>
016602  033057  044040  040505
016610  020104  046101  043511
016616  046516  047105  020124
016624  042020  044522  044506
016632  040503  044524  047111
016640  050040  047522  051103
016646  046501  020015  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  051506  020124  047502
016734  020130  040520  020123
016742  042502  047105  041440
016750  047117  042516  052103
016756  042105  000015
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 NOR <LF>
017102  020105  047111  042504
017110  020130  051105  047522
017116  051522  040440  042522
017124  041440  052501  042523
017132  020104  054502  052040
017140  042515  040440  044514
017146  047107  042515  052116
017154  050040  041501  020113
```

```
 3636  017162  047101  020104  040515
 3637  017170  006531      012
 3640  017173      040  047516  020124          .ASCII    NOT OCCUR WITH SOME ALIGNMENT PACKS.  TO INHIBIT THE ALIGNMENT PAC
 3641  017201  041511  052503  020133
 3642  017206  044527  044124  051440
 3643  017214  046517  020105  046101
 3644  017222  043511  046516  047105
 3645  017230  020124  040520  045503
 3646  017236  027123  020040  047524
 3647  017244  044440  044116  041111
 3650  017252  052111  052040  042510
 3651  017260  040440  044514  047105
 3652  017266  042515  052116  050040
 3653  017274  041501  006513      012
 3654  017301      040  042522  040514          .ASCII    RELATED INDEX ERROR. GROUND THE FOLLOWING PINS:<CR><LF>
 3655  017306  042524  020104  047111
 3656  017314  042504  020130  051105
 3657  017322  047522  026122  043440
 3660  017330  047522  047125  020104
 3661  017336  044124  020105  047506
 3662  017344  046114  053517  047111
 3663  017352  020107  044520  051516
 3664  017360  006472      012
 3665  017363      040  020040  020040          .ASCII    '       RP04: PIN A1AC5C3 (FUNCTION '+INDEX'<CR><LF>
 3666  017370  051040  030120  035064
 3667  017376  050040  047111  040440
 3670  017404  040461  032460  034060
 3671  017412  024040  052506  041516
 3672  017420  044524  047117  023440
 3673  017426  052053  047120  042504
 3674  017434  042530  023522  006451
 3675  017442      012
 3676  017443      040  020040  020040          .ASCIZ  @       RP05 6:  PIN D04R19 (FUNCTION '-INDEX'@<CR><LF><LF>
 3677  017450  051040  030120  027465
 3700  017456  035066  020040  044520
 3701  017464  020116  030104  051064
 3702  017472  034461  024040  052506
 3703  017500  041516  044524  047117
 3704  017506  023440  044455  042116
 3705  017514  054105  024447  005015
 3706  017522  000012
 3707
 3710  017524  005015  051104  053111  NOCRV:  .ASCIZ   <CR><LF>'DRIVE NOT AVAILABLE'<CR><LF><LF>
 3711  017532  020105  047516  020124
 3712  017540  053101  044501  040514
 3713  017546  046102  006505  005012
 3714  017554      000
 3715
 3716  017555      015  042412  052116  ENTERD: .ASCIZ  <CR><LF>/ENTER DRIVE NUMBER:
 3717  017562  051105  042040  044522
 3720  017570  042526  047040  046525
 3721  017576  042502  035122  000040
 3722
 3723  017604  005015  051104  053111  WLOCK:  .ASCIZ  <CR><LF>' DRIVE NOT WRITE PROTECTED '<CR><LF><LF>
```

```
                                                                                
003660  017612  020105  047516  020124
003661  017620  051122  052111  020105
003662  017626  051120  052117  041505
003663  017634  042524  006504  005012
003664  017642      000
003665
003666  017643      042  044040  040505  TOLRG:  .ASCIZ    HEAD OUT OF RANGE
003667  017650  020104  052517  020124
003668  017656  043117  051040  047101
003669  017664  042507      000
003670
003671  017667      015  020012  020040  HEADER: .ASCII  <CR><LF>                TRK CENTER<CR><LF>
003672  017674  020040  020040  020040
003673  017702  020040  020040  052040
003674  017710  045522  041440  047105
003675  017716  042524  006522      012
003676  017723      040  020040  042510          .ASCII    HEAD CYL (IN 3 INCHES)<CR><LF><LF>
003677  017730  042101  020040  054503
003678  017736  020114  044450  020116
003679  017744  020125  047111  044103
003680  017752  051505  006451  005012
003681  017760      000
003682
003683  017761      101  046050  043511  MODE:   .ASCII  /A(LIGN), V(ERIFY), OR E(XERCISE) /
003684  017766  024516  020054  024126
003685  017774  051105  043111  024531
003686  020002  020054  051117  042440
003687  020010  054050  051105  044503
003688  020016  042523  020051  020077
003689  020024      000
003690
003691  020025      126  051105  043111  VERIFY: .ASCIZ  /VERIFY/<CR><LF><LF>
003692  020032  006531  005012      000
003693
003694  020037      101  044514  047107  ALIGN:  .ASCIZ  /ALIGN/<CR><LF><LF>
003695  020044  005015  000012
003696
003697  020050  054105  051105  044503  EXER:   .ASCIZ  /EXERCISE/<CR><LF><LF>
003698  020056  042523  005015  000012
003699
003700  020064  042510  042101  030040  HDZERO: .ASCIZ  /HEAD 0 SELECTED BY DEFAULT<CR><LF><LF>
003701  020072  051440  046105  041505
003702  020100  042524  020104  054502
003703  020106  042040  043105  052501
003704  020114  052114  005015  000012
003705
003706  020122  054503  044514  042116  CYL245: .ASCIZ  /CYLINDER 245 SELECTED BY DEFAULT<CR><LF><LF>
003707  020130  051105  031040  032464
003708  020136  051440  046105  041505
003709  020144  042524  020104  054502
003710  020152  042040  043105  052501
003711  020160  052114  005015  000012
003712
003713  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  052124  005015  000012
                        3719
ALN245: .ASCIZ  DISK IS POSITIONED AT CYLINDER 245<CR><LF><LF>
                        3720  020232  044504  045523  044440
                        3721  020240  020123  047520  044523
                        3722  020246  044524  047117  042105
                        3723  020254  040440  020124  054503
                        3724  020262  044514  042116  051105
                        3725  020270  033040  032464  005015
                        3726  020276  000012
                        3727
ALN496: .ASCIZ  <DISK IS POSITIONED AT CYLINDER 496  CR><LF><LF>
                        3728  020300  044504  045523  044440
                        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
ENTHD:  .ASCIZ  <CR><LF>ENTER HEAD: <
                        3736  020346  005015  047105  042524
                        3737  020354  020122  042510  042101
                        3738  020362  020072     000
                        3739
ENTCYL: .ASCIZ  <CR><LF>ENTER CYLINDER ADDRESS: <
                        3740  020365     015  042412  052116
                        3741  020372  051105  041440  046131
                        3742  020400  047111  042504  020122
                        3743  020406  042101  051104  051505
                        3744  020414  035123  000040
                        3745
BADTRK: .ASCIZ  <INVALID HEAD ADDRESS<CR><LF>
                        3746  020420  047111  040526  044514
                        3747  020426  020104  042510  042101
                        3748  020434  040440  042104  042522
                        3749  020442  051523  005015     000
                        3750
BADCYL: .ASCIZ  <INVALID CYLINDER ADDRESS<CR><LF>
                        3751  020447     111  053116  046101
                        3752  020454  042111  041440  046131
                        3753  020462  047111  042504  020122
                        3754  020470  042101  051104  051505
                        3755  020476  006523  000012
                        3756
ENDMSG: .ASCIZ  <CR><LF><LF><DONE><CR><LF><LF>
                        3757  020502  005015  042012  047117
                        3758  020510  006505  005012     000
                        3759
                        3760
                        3761          020516                .EVEN
                        3762
EM1:    .ASCIZ  <RH11 INTERRUPT OCCURRED (RPAS=0)>
                        3763  020516  044122  030461  044440
                        3764  020524  052116  051105  052522
                        3765  020532  052120  047440  041503
                        3766  020540  051125  042522  020104
                        3767  020546  051050  040520  036523
```

```
3769  020554  024460     000
3769
3770  020557     125  042516  050130  EM2:    .ASCIZ  UNEXPECTED ATTENTION OCCURRED/
3771  020564  041505  043524  020104
3772  020572  052101  043524  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  000300
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+RPCS1,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
```

```
                                                                                
                                                                                
022757  000     000     000  DF3:    .BYTE   0,0,0
                                                                                
022762  000     000     000  DF4:    .BYTE   0,0,0,0
022765  000
                                                                                
022766  000     000     000  DF10:   .BYTE   0,0,0,0,0,0,0
022771  000     000     000
022774  000
                                                                                
022775  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
```

```
3994  022116  022124                        4$                      ;TERMINATED WITH A CARRIAGE RETURN
3995  022120  022064                        3$                      ;TERMINATED WITH A ","
3996  022122  022124                        4$                      ;TERMINATED WITH A "."
3997  022124  010260  177776        4$:     MOV     R2,-2(R0)               ;SAVE INPUT
3998  022130  013701  000004        7$:     MOV     ERRVEC,R1       ;SAVE THE ERROR VECTOR
3999  022134  012737  022170  000004        MOV     #8$,ERRVEC      ;SETUP FOR TRAP
4000  022142  005777  157116                TST     @$RPADR         ;CHECK FOR RH11
4001  022146  010137  000004                MOV     R1,ERRVEC       ;RESTORE ERROR VECTOR
4002  022152  012700  001264                MOV     #$RPADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
4003  022156  012701  007704                MOV     #RPADR,R1       ;FIRST ADDRESS OF WHERE TO PUT THEM
4004  022162  012021                        MOV     (R0)+,(R1)+     ;BUS ADDRESS
4005  022164  012021                        MOV     (R0)+,(R1)+     ;VECTOR ADDRESS
4006  022166  000207                        RTS     PC              ;RETURN
4007  022170  010137  000004        8$:     MOV     R1,ERRVEC       ;RESTORE ERROR VECTOR
4008  022174  022626                        CMP     (SP)+,(SP)+     ;CLEAN OFF THE STACK
4009  022176  104006                        ERROR   6               ;REPORT THE ERROR
4010  022200  000705                        BR      1$              ;ASK FOR BUS ADDRESS
4011
4012  022202  050122  051503  020061  MRPCS1: .ASCIZ  @RPCS1 = @
4013  022210  020075     000
4014  022213     122  053110  041505  MRHVEC: .ASCIZ  @RHVEC = @
4015  022220  036440  000040
4016  022224  020040     000          LINSP:  .ASCIZ  @ @
4017
4018          022230                  .EVEN
4019
4020                                  .SBTTL  CK.NUM - CHECK NUMBER (OCTAL)
4021                                  ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
4022                                  ;AND FORMS AN OCTAL NUMBER IN R2
4023                                  ;CALL:
4024                                  ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
4025                                  ;       MOV     #NUM,R2         ;MAX SIZE OF INPUT NUMBER
4026                                  ;       JSR     R5,CK.NUM       ;GO FORM THE NUMBER
4027                                  ;       RETURN  ADR1            ;"CR" ONLY ENTERED -- R2 = 0
4028                                  ;       RETURN  ADR2            ;"PERIOD" ONLY ENTERED -- R2 = 0
4029                                  ;       RETURN  ADR3            ;ILLEGAL CHARACTER IN THE INPUT STRING
4030                                  ;       RETURN  ADR4            ;"CR" ENTERED -- R2 = NUMBER
4031                                  ;       RETURN  ADR5            ;"COMMA" -- R2 = NUMBER
4032                                  ;       RETURN  ADR6            ;"PERIOD" -- R2 = NUMBER
4033
4034  022230  010446          CK.NUM: MOV     R4,-(SP)        ;SAVE R4
4035  022232  010346                  MOV     R3,-(SP)        ;SAVE R3
4036  022234  010246                  MOV     R2,-(SP)        ;SAVE R2
4037  022236  005004                  CLR     R4              ;RETURN POINTER
4038  022240  005003                  CLR     R3              ;START NUMBER AT ZERO
4039  022242  005002                  CLR     R2              ;STORE RESULT
4040  022244  004537  022444          JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
4041  022250  022346                  6$                      ;ILLEGAL CHARACTER
4042  022252  022352                  8$                      ;CARRIAGE RETURN
4043  022254  022346                  6$                      ;" "
4044  022256  022350                  7$                      ;","
4045  022260  022264                  1$                      ;DIGIT 0-7
4046  022262  022346                  6$                      ;DIGIT 8-9
4047  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                        6$                      ;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
```

```
                    000012          LINSP   022224          LKS     001252
                    000000          MAXCYL  001312          MAYNEG  000216
                    001214          MICLK   = 000002          MCPE    = 020000
                    001000          MIDCYL  001230          MIDTOL  001224
                    000016          MODE    017761          MOH     = 020000
                    000400          MRD     = 000020          MRHVEC  022213
                    000200          MSTCK   = 000010          MWR     = 000040
                    001000          MXLACT  007712          MXWNDW  007720
                    100000          NED     = 010000          NEM     = 040000
                    017554          OCYL    = 100000          OFFDIR  001176
                    000200          OFTBL4  016405          OFTBL6  016324
                    000100          OFS25   = 000001          OF400   = 000020
                    000040          ONELD   002444          OPE     = 020000
                    010754          OUTCYL  001232          PC      =%000007
                    000010          PKT     = 000020          PIRQ    = 177772
                    001552          PKTS    001254          PKTSIZE 001250
                    020000          PLS     001204          POPSIZE 016056
                    000000          PR1     = 000040          PR2     = 000100
                    000200          PR5     = 000240          PR6     = 000300
                    177776          PSEL    = 002000          PSL     = 000001
                    000024          QCNT    015370          QDRVQ   015462
                    015520          QDRV3   015542          QDRV4   015562
                    015540          QDRV7   015642          QINPT   015400
                    015440          QSTOP   015442          QTERM   = 015662
                    000020          RDCHR   = 104410          RDDAT   = 000177
                    104412          RDY     = 000200          RD.RDR  014666
                    014664          RD.RP2  015000          RD.RP3  015004
                    014664          RECNO   = 000107          REG     016140
                    104413          RESVEC  = 000010          RMR     = 000004
                    007704          RPAS    = 000016          RPBA    = 000004
                    000036          RPCS1   = 000000          RPCS2   = 000010
                    000022          RPDS1   = 000012          RPDT    = 000026
                    000046          RPERRS  007536          RPER1   = 000014
                    000042          RPINIT  007722          RPLA    = 000020
                    000032          RPSN    = 000030          RPTMR   014110
                    000002          RPWC    000002          RTC     = 000112
                    =%000001          R2      =%000002          R3      =%000003
                    =%000005          R6      =%000006          R7      =%000007
                    104412          SC      012772          SC1     = 000100
                    013610          SC12    013700          SC13    013750
                    002000          SC3     013036          SC4     013042
                    013244          SC6A    013354          SC7     013502
                    000131          SEC     = 000010          SEEK    = 000105
                    001240          SELDRV= 000145          SETFMT= 000143
                    001202          SKI     = 040000          SP      =%000006
                    004256          STACK   = 001100          START   001450
                    001464          STATUS= 000016          STK_MT= 177774
                    014232          STO2    014430          STO3    014500
                    014532          STO7    014570          STO8    014620
                    004114          ST.LCL  004232          ST.PCL  004200
                    001140          SWREG   000176          SW0     = 000001
                    000004          SW02    = 000004          SW03    = 000010
                    000340          SW06    = 000100          SW07    = 000200
```

```
[illegible]  000000  $ECO    = 001000  SW1    = 000002  $W10    = 002000
[illegible]  000000  $FIS    = 010000  SW13   = 020000  SW14   = 040000
[illegible]  000000  $W2     = 000004  SW3    = 000010  SW4    = 000020
[illegible]  000000  $W5     = 000100  SW6    = 000200  SW8    = 000400
[illegible]  000000  $ABLE1  = 001234  TABLE2  = 001236  TAP    = 040000
[illegible]  000000  TCB     = 012632  TDF    = 000040  TIMER  = 007650
[illegible]  000000  TKVEC   = 000060  TQ.ER   = 001210  TQLRG  = 001643
[illegible]  000000  TPVEC   = 000064  TRADVE  = 000034  TRE    = 040000
[illegible]  000000  TRK1    = 004000  TRK10   = 040000  TRK2   = 010000
[illegible]  000000  TRK4    = 020000  TRNSWT  = 007616  TRTVEC = 000014
[illegible]  000000  TYPOS   = 104405  TYPE   = 104401  TYPOT  = 104402
[illegible]  000000  TYPOS   = 104403  ULDFLG  = 007624  UNLOAD = 000103
[illegible]  000000  UFE     = 020000  US1    = 000001  US2    = 000002
[illegible]  000000  UXB     = 000010  VERIFY  = 020025  VUF    = 000002
[illegible]  000000  VV      = 000100  WRC    = 000002  WTE    = 040000
[illegible]  000000  XCHAD   = 000154  XCHMD  = 000153  XSU    = 000001
[illegible]  000000  XLOCK   = 017604  XRL    = 004000  XRTCNT = 000161
[illegible]  000000  XRT.RO  = 015100  XRT.R2  = 015012  XRT.R1 = 015076
[illegible]  000000  XRT.R3  = 015114  XRT.R4  = 015134  XRT.R5 = 015176
[illegible]  000000  XSU     = 000004  $AUTOB  = 001134  $BADDAT = 001126
[illegible]  000000  $BELL   = 006767  $CHARC  = 005110  $CMTBG  = 001100
[illegible]  000000  $CM3    = 000000  $CNTLC  = 006773  $CNTLO  = 007000
[illegible]  000000  $CRLF   = 001161  $DBLK   = 005556  $CTBL   = 005546
[illegible]  000000  $ERRFLG = 001103  $ERMAX  = 001115  $ERRFC  = 001116
[illegible]  000000  $ERRTB  = 001270  $ERRTY  = 004520  $FILLC  = 001156
[illegible]  000000  $FILLS  = 001155  $GDADR  = 001120  $GTSWR  = 006162
[illegible]  000000  $HD     = 000001  $HEAD   = 001172  $ICNT   = 001104
[illegible]  000000  $INTAG  = 001135  $ITEMB  = 001114  $LONUM  = 007134
[illegible]  000000  $LPADR  = 001106  $LPERR  = 001110  $MSWR   = 007012
[illegible]  000000  $NULL   = 001154  $OCNT   = 005336  $PASS   = 001100
[illegible]  000000  $QUES   = 001160  $RAND   = 007034  $RDLIN  = 006524
[illegible]  000000  $RDSZ   = 000007  $RESRE  = 007414  $RPVEC  = 001266
[illegible]  000000  $SAVRE  = 007355  $SETUP  = 000146  $SWR    = 120000
[illegible]  000000  $TKB    = 001146  $TKCNT  = 005566  $TKQEN  = 005603
[illegible]  000000  $TKQIN  = 005570  $TKQOU  = 005572  $TKSRV  = 005654
[illegible]  000000  $TN     = 000001  $TPS    = 001150  $TPFLG  = 001157
[illegible]  000000  $TRAP   = 007453  $TRPAD  = 007506  $TSTNM  = 001102
[illegible]  000000  $TYPE   = 004674  $TYPEC  = 005044  $TYPOS  = 005342
[illegible]  000000  $TYPON  = 005154  $TYPOS  = 005114  $TYPOC  = 005140
[illegible]  000000  .       = 022516
```

ERRORS DETECTED: 0

```
*DZRSOA,DZRSO/SOL/-I:ME/NL:MC:MD:CND=RPC456.010.DZRSO.009
[illegible] TIME: 24 51 0 SECONDS
[illegible] CORE USED: 30K
```

Spooler runtime 13 Seconds, 83 KCS, 3% disk reads, 0 disk writes, 93 pages printed