

TU60

TA11 MOTION TEST
CZTADD0.

AH-9358D-MC

JUN 1978

COPYRIGHT © 73-78

digital

FICHE 1 OF 1

MADE IN USA

B01

EOF1CZRKIFSEQ
SEQ 000PDP10 411
CZTADD.P11

00010000 780519
TALL MOTION TEST

23-MAR-78 08:34

PDP10 411
CZTAD-D MACY11 30A(1052)

0 HDR1CZTADDSEQ
23-MAR-78 08:36 PAGE 1

00010000

780519

SEQ 0001

.REM 2

IDENTIFICATION

PRODUCT CODE: AC-9357D-MC
PRODUCT NAME: CZTADD TALL MOTION TST
DATE REVISED: FEB 1978
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: JIM LACEY

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1973, 1978 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
3. LOADING PROCEDURE
4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESS
 - 4.3 PROGRAM & OPERATOR ACTION
5. OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 SUBROUTINE ABSTRACTS
6. ERRORS
7. RESTRICTIONS
8. MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 STACK POINTER
 - 8.3 PASS COUNTER
 - 8.4 ITERATIONS
 - 8.5 SPECIAL REGISTERS
9. PROGRAM DESCRIPTION
 1. ABSTRACT

THIS PROGRAM CONTAINS A SERIES OF TESTS THAT CHECK THE TU60 DRIVE FOR PROPER OPERATION.
 2. REQUIREMENTS
 - 2.1 EQUIPMENT

PDP-11 COMPUTER WITH OR WITHOUT HARDWARE SWITCH REGISTER WITH CONSOLE TELETYPE, AND A TALI CASSETTE
 - 2.2 STORAGE

THIS PROGRAM REQUIRES APPROX. 4K STORAGE.
 - 2.3 PRELIMINARY PROGRAMS

CZTAA

CZTAB
CZTAC

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR LOADING .ABS TAPES
OR A CASSETTE TAPE.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE 5.1.

4.2 STARTING ADDRESSES

| | |
|-----|--|
| 200 | NORMAL STARTING ADDRESS |
| 204 | SELECT DRIVE(S) BEFORE STARTING TEST |
| 210 | SELECT DRIVE(S) AND ADDRESSES BEFORE STARTING TEST |
| 214 | SETUP FOR MANUAL LOOPING |
| 220 | WRITE FILE GAP FROM BOT TO EOT |
| 224 | WRITE CONTINUOUS BLOCKS OF DATA |
| 230 | READ CONTINUOUS BLOCKS OF DATA |
| 234 | WRITE FILE GAP AND A BLOCK OF DATA |
| 240 | READ BLOCK OF DATA AND INTO A FILE GAP |
| 244 | SPACE FWD FILE GAP FROM BOT TO EOT |
| 250 | BACK SPACE FILE GAPS |

4.3 PROGRAM & OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3.)
2. LOAD A WRITE ENABLED CASSETTE IN BOTH DRIVES
3. REWIND BOTH DRIVES
4. LOAD ADDRESS 200.
5. SET SWITCHES (SEE SECTION 5.1)
6. PRESS START.
7. THE PROGRAM WILL LOOP & TTY BELL WILL RING ONCE EVERY PASS, IF SW<10>=0.

*** NOTE: IF USING THE SOFTWARE SWITCH REGISTER THE PROGRAM WILL TYPE "SWR=XXXXXX NEW=" AFTER TYPING THE NAME OF THE PROGRAM.

4.3.1 DRIVE SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECTION OF DRIVES "A" AND "B" TO BE TESTED.
NOTE: IF LOAD MEDIUM IS CASSETTE WITH STANDARD VECTOR PROGRAM WILL RESPOND AS IF STARTED AT 210.

STARTING THE PROGRAM AT 204, 210, OR 214 ALLOWS THE OPERATOR TO SELECT THE DRIVE(S) TO BE TESTED.

THE PROGRAM WILL TYPE "DRIVE(S)?".

EITHER OR BOTH DRIVES CAN BE SELECTED BY TYPING "A" AND/OR "B" FOLLOWED BY A CARRIAGE RETURN.

4.3.1.1 DRIVE SELECTION EXAMPLES

DRIVE(S)? A,B
DRIVE(S)? AB
DRIVE(S)? B,A
DRIVE(S)? B

4.3.2 ADDRESS SELECTION

STARTING THE PROGRAM AT 210 OR 214 ALLOWS THE OPERATOR TO CHANGE THE "CONTROL AND STATUS" AND "DATA BUFFER" REGISTER ADDRESSES, THE VECTOR ADDRESS AND THE PRIORITY LEVEL.

THE PROGRAM WILL ASK FOR THE DRIVES TO BE TESTED AS PER 4.3.1. AFTER THE DRIVES HAVE BEEN SELECTED IT WILL ASK FOR:

1. BUS ADDRESS OF THE CONTROL AND STATUS REGISTER (TACS)
2. VECTOR ADDRESS
3. PRIORITY LEVEL

AND THE OPERATOR MUST RESPOND WITH THE DESIRED PARAMETER OR A CARRIAGE RETURN (WHICH IMPLIES LEAVE AS IS). WHEN ALL PARAMETERS HAVE BEEN DEFINED THE PROGRAM WILL TYPE THEM BACK OUT AND ASK IF THEY ARE OK AT WHICH TIME THE OPERATOR RESPONDES WITH A "Y" OR A "CARRIAGE RETURN" FOR "YES" ANYTHING ELSE IS A "NO".

4.3.2.1 ADDRESS SELECTION EXAMPLES

```
DRIVES(S) A
TACS? 177500
VECTOR? 260
PRIORITY? 6
TACS=177500 TADB=177502 VECTOR=000260 PRIORITY=000300
OK?
```

```
DRIVES(S) A,B
TACS? 470
VECTOR?
PRIORITY?
TACS=177470 TADB=177472 VECTOR=000260 PRIORITY=000300
OK?
```


4.3.3 SUBTEST LOOPING

THE SCOPE ROUTINE (REFER 5.2.1) PROVIDES A MEANS BY WHICH THE OPERATOR CAN SPECIFY THE FIRST ADDRESS OF A SCOPE LOOP.

THE OPERATOR TYPES A "CONTROL C" (↑C) AND WHEN THE NEXT SCOPE STATEMENT IS EXECUTED THE PROGRAM WILL ASK FOR:

1. TEST PC--- THE FIRST ADDRESS OF THE TEST (MUST BE A SCOPE)
2. LOOP PC--- THE ADDRESS TO LOOP BACK TO

4.3.3.1 SUBTEST LOOPING EXAMPLES

```
:OPERATOR TYPES "↑C"
↑TEST PC? 2242
LOOP PC?
```

```
:OPERATOR TYPES "2242""CARRIAGE RETURN "(CR)"
:OPERATOR TYPES "CR" WHICH
:IMPLIES "LOOP PC"="TEST PC"
```

```
:OPERATOR TYPES "↑C"
↑TEST PC? 3000
LOOP PC? 3020
```

```
:OPERATOR TYPES "3000""CR"
:OPERATOR TYPES "3020""CR"
:PROGRAM USES THIS AS THE
:FIRST ADDRESS OF THE SCOPE
:LOOP
```

```
:OPERATOR TYPES "↑C"
↑TEST PC?
```

```
:OPERATOR TYPES "CR"
:PROGRAM CONTINUES
:FROM THIS POINT
```


- 5. OPERATING PROCEDURE
- 5.1 OPERATIONAL SWITCH SETTINGS

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G (<↑G>): THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: SWR=XXXXXXNEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE 'NEW=' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
 - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED) IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
 - B) IF A CONTROL U (<↑U>) IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

WITH SW<15:08>=0 THE PROGRAM WILL PRINT OUT ON ERRORS AND CONTINUE IN TEST. BELL WILL RING AT COMPLETION OF A PASS. THE SWITCH SETTINGS ARE:

SW<15>=1...HALT ON ERROR
 SW<14>=1...LOOP ON TEST
 SW<13>=1...INHIBIT ERROR TYPEOUTS
 SW<11>=1...INHIBIT ITERATIONS
 SW<10>=1...RING BELL ON ERROR
 SW<10>=0...RING BELL ON PASS COMPLETE
 SW<09>=1...LOOP ON ERROR
 SW<07>=1...LOCK ON CURRENT DRIVE
 SW<06>=1...DELAY AT END OF EACH FUNCTION
 SW<05>=1...RUN WITHOUT INTERRUPTS
 SW<04>=1...IGNORE BLOCK CHECK ERRORS
 SW<03>=1...INHIBIT DATA COMPARE

5.2 SUBROUTINE ABSTRACTS

5.2.1 SCOPE

THIS SUBROUTINE CALL (VIA AN IOT INSTRUCTION) IS PLACED BETWEEN EACH TEST IN THE INSTRUCTION SECTION. IT RECORDS THE STARTING ADDRESS OF EACH TEST IN LOCATION "\$LPADR" AND "\$LPERR" AS IT IS BEING ENTERED.

THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
NOTE: THIS ROUTINE CHECKS THE TTY INPUT BUFFER FOR A "CONTROL C" (REFER TO 4.3.3)

5.2.2 TRAPCATCHER

A ".+2" - "HALT" SEQUENCE IS REPEATED FROM LOC. 0 TO LOC. 776 TO CATCH ANY UNEXPECTED TRAPS. THUS, ANY UNEXPECTED TRAPS WILL HALT AT THE DEVICE TRAP VECTOR +2.

5.2.3 ERROR

THIS SUBROUTINE CALL (VIA A EMT INSTRUCTION) IS USED TO REPORT ALL ERRORS. (REFER TO 6.)

*** THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
*** IF THE PROCESSOR HALTS (BIT 15=1), OPERATOR CAN RESET S/W SWITCH REGISTER BY HITTING A "CONTROL G" (<G>) BEFORE HITTING CONTINUE.

5.2.4 TRAP

A NUMBER OF SUBROUTINES ARE CALLED BY THE TRAP INSTRUCTION. FOLLOWING IS THE CALLS USED AND THE STARTING ADDRESS OF THE ROUTINE.

5.2.4.1 TYPE (\$TYPE)

TYPE AN ASCIZ STRING ON THE TTY

5.2.4.2 RDCHR (\$RDCHR)

READ A SINGLE ASCII CHARACTER FROM THE TTY

5.2.4.3 RDLIN (\$RDLIN)

READ AN ASCII STRING FROM THE TTY

5.2.4.4 SETLOOP (T.SETLOOP)

SETUP TO LOOP AS PER THE TTY

5.2.5 THE FOLLOWING "TRAP" CALLS ARE WHAT ARE USED TO PERFORM THE TESTS.

THE ROUTINES THAT ARE CALLED IS WHAT MAKES UP THE HEART OF THIS PROGRAM.

5.2.5.1 WFG (T.WFG) WRITE A FILE GAP

5.2.5.2 WRITE (T.WRIT) WRITE A BLOCK OF DATA

5.2.5.3 READ (T.READ) READ A BLOCK OF DATA

5.2.5.4 BSFG (T.BSFG) BACK SPACE A FILE GAP

5.2.5.5 BSBG (T.BSBG) BACK SPACE A BLOCK GAP

5.2.5.6 SFFG (T.SFFG) SPACE FORWARD A FILE GAP

5.2.5.7 SFBG (T.SFBG) SPACE FORWARD A BLOCK GAP

5.2.5.8 REWIND (T.RWIND) REWIND THE TAPE TO BOT

5.2.5.9 SELDRV (T.SELDRV) SELECT A DRIVE

5.2.5.10 BLKCMP (T.BLKCMP) COMPARE READ AND WRITE BUFFERS

5.2.6 THE FOLLOWING SUBROUTINES ARE CALLED BY A JSR

5.2.6.1 DO.CMD

THIS ROUTINE WILL LOAD THE LOW BYTE OF THE "TACS" WITH THE FIRST BYTE FOLLOWING THE CALL.

WHEN THE FUNCTION IS TO BE PERFORMED WITH "INTERRUPT ENABLE"=1 THE TAIL VECTOR IS SET TO "SERV". WHEN THE FUNCTION IS TO BE PERFORMED WITH "INTERRUPT ENABLE"=0 THE VECTOR IS SET TO "BADINT"

NOTE SWR<5> PROVIDES OVERRIDE CAPABILITIES OF "INTERRUPT ENABLE"=1.

5.2.6.2 WAITFLAG

THIS ROUTINE IS CALLED AFTER A COMMAND HAS BEEN SENT TO THE TAIL. IT WAITS A PREDETERMINED AMOUNT OF TIME AND TAKES ONE OF THREE EXITS.

THE EXITS FOLLOW THE CALL AND ARE:

1. ERROR NO FLAGS OCCURRED
2. ERROR NO INTERRUPT OCCURRED
3. NORMAL RETURN

5.2.6.3 FLAGS

THIS ROUTINE IS CALLED TO DETERMINE WHAT FLAGS ARE UP. THIS ROUTINE WILL TAKE ONE OF FOUR RETURNS DEPENDING ON THE FLAGS.

THE RETURNS FOLLOW THE CALL AND ARE:

1. "TRANSFER REQUEST"=0 AND "READY"=0
2. "TRANSFER REQUEST"=1
3. "READY"=1 AND "ERROR"=0
4. "READY"=1 AND "ERROR"=1

5.2.6.3 DO.CRC

THIS ROUTINE IS USED TO CALCULATE "CRC". IT WORKS ON ONE BYTE AT A TIME WHICH MUST BE IN R0 WHEN CALLED.

5.2.6.4 A2OCT

THIS ROUTINE CHANGES AN ASCII STRING TO AN OCTAL NUMBER.

5.2.6.5 GETDRV

THIS ROUTINE IS USED TO ASK THE OPERATOR WHICH DRIVE(S)
ARE TO BE TESTED

5.2.6.6 ASKADR

THIS ROUTINE IS USED TO INPUT THE ADDRESSES FOR THE "TACS",
"TADB" AND THE VECTOR AND THE PRIORITY LEVEL TO USE.

5.2.6.7 TYPERR

THIS ROUTINE IS USED TO TYPE OUT THE "ERROR" DATA

5.2.6.8 EXAM

THIS ROUTINE IS USED TO DETERMINE WHICH DRIVE(S) ARE AVAILABLE
FOR TESTING.

5.2.7 THE FOLLOW ROUTINES ARE USED TO MAKE ADJUSTMENTS TO
THE TUGO. BEFORE USING ANY OF THEM LOAD AND START 214.

5.2.7.1 WFGSUB

WRITE FILE GAPS FROM "BOT" TO "EOT"
START AT 220
THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND
THE "WRITE DELAY MONO".

5.2.7.2 WRTSUB

WRITE CONTINUOUS BLOCKS OF DATA
START AT 224
THE PROGRAM WILL HALT THREE(3) TIMES
AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
HALT 2 ---SWR<7:0> = PATTERN DESIRED
HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"
** IF USING SOFTWARE SWITCH REGISTER, AFTER
EACH HALT OPERATOR WILL BE PROMPTED
FOR THE VALUE WITH "SWR=XXXXXX NEW="

5.2.7.3 RDSUB

READ CONTINUOUS BLOCKS OF DATA
START AT 230
THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
AND THE "THRESHOLD POT"

5.2.7.4 WGPBLK

WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO ECT
START AT 234
THE PROGRAM WILL HALT THREE (3) TIMES
AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
HALT 2 --- SWR<7:0> = PATTERN DESIRED
HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
AND THE "GAP TIME MONO".

** IF USING SOFTWARE SWITCH REGISTER, AFTER
EACH HALT OPERATOR WILL BE PROMPTED
FOR THE VALUE WITH "SWR=XXXXXX NEW="

5.2.7.5 RGBLK

READ A BLOCK OF DATA AND A FILE GAP
START AT 240
THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
IT CAN BE USED TO ADJUST THE "SIGNAL MON". THE THRESHOLD POT"
AND THE "TAPE BLANK MONO".

5.2.7.6 SFFGSB

SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
START AT 244
THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
SPACE FOWARD (TAPE BLANK MONO CAN BE ADJUSTED). OR AFTER READ OR
WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
(SIGNAL MONO CAN BE CHECKED).

5.2.7.7 BSFGSB

BACK SPACE FILE GAP
START AT 250
THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".

6. ERRORS

THERE ARE A NUMBER OF ERRORS THAT CAN OCCUR IN THIS PROGRAM. WHEN AN ERROR IS ENCOUNTERED THE CALL TO THE ERROR (ERROR) ROUTINE IS MADE AND IF SW(13) IS NOT SET AN ERROR MESSAGE PERTAINING TO THE ERROR WILL BE TYPED. EACH ERROR TYPE OUT WILL CONTAIN THE FOLLOWING:

1. THE FUNCTION BEING PERFORM
2. AN ERROR MESSAGE
3. A DATA HEADER
4. A DATA STRING

REFER TO THE LISTING UNDER SEARTEB FOR THE DIFFERENT ERRORS THAT CAN OCCUR.

7. RESTRICTIONS

BEFORE STARTING THE PROGRAM THE OPERATOR MUST INSURE THAT A CASSETTE IS LOADED IN THE DRIVE(S) TO BE TESTED AND IS WRITE ENABLED.

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE FIRST PASS TAKES APPROXIMATELY 4 MINUTES ALL SUBSEQUENT PASSES TAKE APPROXIMATELY 8 MINUTES

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100.

8.3 PASS COUNT

A PROGRAM PASS THRU COUNT IS KEPT IN "\$PASS".

8.4 ITERATIONS

THE FIRST PASS OF THE PROGRAM WILL AUTOMATICALLY INHIBIT ITERATIONS. ALL SUBSEQUENT PASSES WILL PERFORM FULL, (ONE PER DRIVE), ITERATIONS.

8.5 SPECIAL REGISTERS

R4 AND R5 ARE RESERVED FOR "TACS" AND "TADB" THROUGH OUT THE PROGRAM.

| | | | | |
|-----|----|---------------------------|---|-------------|
| 620 | 09 | OFF LINE | ? | READ ONLY |
| 621 | 08 | UNIT SELECT | 0 | READ/WRITE |
| 622 | 07 | TRANSFER REQUEST | 0 | READ ONLY |
| 623 | 06 | INTERRUPT ENABLE | 0 | READ/WRITE |
| 624 | 05 | READY | 1 | READ ONLY |
| 625 | 04 | ILBS | 0 | READ/WRITE |
| 626 | 03 | FUNCTION BIT 02 | 0 | READ/WRITE |
| 627 | 02 | FUNCTION BIT 01 | 0 | READ/WRITE |
| 628 | 01 | FUNCTION BIT 00 | 0 | READ/WRITE |
| 629 | | 0=WRITE-FILE-GAP | | |
| 630 | | 1=WRITE | | |
| 631 | | 2=READ | | |
| 632 | | 3=BACK SPACE FILE GAP | | |
| 633 | | 4=BACK SPACE BLOCK GAP | | |
| 634 | | 5=SPACE FORWARD FILE GAP | | |
| 635 | | 6=SPACE FORWARD BLOCK GAP | | |
| 636 | | 7=REWIND | | |
| 637 | 00 | GO BIT | 0 | WRITE ONLY! |

638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693

.SBTTL OPERATIONAL SWITCH SETTINGS

| SWITCH | USE |
|--------|-------------------------------|
| 15 | HALT ON ERROR |
| 14 | LOOP ON TEST |
| 13 | INHIBIT ERROR TYPEOUTS |
| 11 | INHIBIT ITERATIONS |
| 10 | BELL ON ERROR |
| 9 | LOOP ON ERROR |
| 7 | LOCK ON CURRENT DRIVE |
| 6 | DELAY AT END OF EACH FUNCTION |
| 5 | RUN WITHOUT INTERRUPTS |
| 4 | IGNORE BLOCK CHECK ERRORS |
| 3 | INHIBIT DATA COMPARE |

.SBTTL BASIC DEFINITIONS

```

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

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

```

```

**GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0                    ;;GENERAL REGISTER
R1= %1                    ;;GENERAL REGISTER
R2= %2                    ;;GENERAL REGISTER
R3= %3                    ;;GENERAL REGISTER
R4= %4                    ;;GENERAL REGISTER
R5= %5                    ;;GENERAL REGISTER
R6= %6                    ;;GENERAL REGISTER
R7= %7                    ;;GENERAL REGISTER
SP= %6                    ;;STACK POINTER
PC= %7                    ;;PROGRAM COUNTER

```

```

**PRIORITY LEVEL DEFINITIONS
PRO= 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

```


694 000300
695 000340

PR6= 300 ::PRIORITY LEVEL 6
PR7= 340 ::PRIORITY LEVEL 7

696
697
698 100000
699 040000
700 020000
701 010000
702 004000
703 002000
704 001000
705 000400
706 000200
707 000100
708 000040
709 000020
710 000010
711 000004
712 000002
713 000001

:"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
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

725 100000
726 040000
727 020000
728 010000
729 004000
730 002000
731 001000
732 000400
733 000200
734 000100
735 000040
736 000020
737 000010
738 000004
739 000002
740 000001

:"DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
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

741
742
743
744
745
746
747
748
749

```

750      .EQUIV BIT01,BIT1
751      .EQUIV BIT00,BIT0
752
753      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
754      000004 ERRVEC= 4      ;: TIME OUT AND OTHER ERRORS
755      000010 RESVEC= 10     ;: RESERVED AND ILLEGAL INSTRUCTIONS
756      000014 TBITVEC=14    ;: "T" BIT
757      000014 TRTVEC= 14     ;: TRACE TRAP
758      000014 BPTVEC= 14     ;: BREAKPOINT TRAP (BPT)
759      000020 IOTVEC= 20     ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
760      000024 PWRVEC= 24     ;: POWER FAIL
761      000030 EMTVEC= 30     ;: EMULATOR TRAP (EMT) **ERROR**
762      000034 TRAPVEC=34    ;: "TRAP" TRAP
763      000060 TKVEC= 60      ;: TTY KEYBOARD VECTOR
764      000064 TPVEC= 64      ;: TTY PRINTER VECTOR
765      000240 PIRQVEC=240    ;: PROGRAM INTERRUPT REQUEST VECTOR
766      ;:*****
767
768      ;*****TAIL FUNCTIONS*****
769      000000 XWFG= 0        ;: WRITE FILE GAP FUNCTION
770      000002 XWRITE= 2      ;: WRITE FUNCTION
771      000004 XREAD= 4       ;: READ FUNCTION
772      000006 XBSFG= 6       ;: BACK SPACE FILE GAP FUNCTION
773      000010 XBSBG= 10      ;: BACK SPACE BLOCK GAP FUNCTION
774      000012 XSFFG= 12      ;: SPACE FWD FILE GAP FUNCTION
775      000014 XSFBG= 14      ;: SPACE FWD BLOCK GAP FUNCTION
776      000016 XRWND= 16      ;: REWIND FUNCTION
777
778      ;*****TAIL BIT ASSIGNMENT*****
779      100000 ERROR= BIT15
780      040000 CRCERR= BIT14
781      020000 LEADER= BIT13
782      010000 WRTLOCK=BIT12
783      004000 FGAP= BIT11
784      002000 TIMERR= BIT10
785      001000 OFFLINE=BIT09
786      000400 UNIT= BIT08
787      000200 TR.REQ= BIT07
788      000100 INT.EN= BIT06
789      000040 READY= BIT05
790      000020 ILBS= BIT04
791      000010 FUNC2= BIT03
792      000004 FUNC1= BIT02
793      000002 FUNC0= BIT01
794      000001 GO= BIT00
795      000016 FUNCTION=      FUNC2+FUNC1+FUNC0
796
797      ;:////////////////////////////////////
798      ;:////////////////////////////////////
799      ;SPECIAL REGISTERS
800
801      000004 TACS= %4      ;:R4 IS USED AS A POINTER TO THE TACS REGISTER
802      000005 TADB= %5     ;:R5 IS USED AS A POINTER TO THE TADB REGISTER.
803      ;:////////////////////////////////////

```



```

804 .SBTTL TRAP CATCHER
805
806         000000
807         .=0
808         ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
809         ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
810         ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
811         .=174
812         DISPREG: .WORD 0           ;; SOFTWARE DISPLAY REGISTER
813         SWREG:   .WORD 0           ;; SOFTWARE SWITCH REGISTER
814         .SBTTL STARTING ADDRESS(ES)
815         JMP      @#BEGIN1          ;; JUMP TO STARTING ADDRESS OF PROGRAM
816         JMP      @#BEGIN2          ;; SELECT DRIVE(S) BEFORE STARTING TEST
817         JMP      @#BEGIN3          ;; SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
818         JMP      @#BEGINX         ;; SETUP FOR MANUAL LOOPING
819         JMP      @#WFGSUB          ;; WRITE FILE GAP FROM BOT TO EOT
820         JMP      @#WRTSUB          ;; WRITE CONTINUOUS BLOCKS OF DATA
821         JMP      @#RDSUB           ;; READ CONTINUOUS BLOCKS OF DATA
822         JMP      @#WGPBLK          ;; WRITE FILE GAP AND A BLOCK OF DATA
823         JMP      @#RGPBLK          ;; READ BLOCK OF DATA AND INTO A FILE GAP
824         JMP      @#SFFGSB          ;; SPACE FWD FILE GAP FROM BOT TO EOT
825         JMP      @#BSFGSB          ;; BACK SPACE FILE GAPS
826         ;; *****

```

.SBTTL COMMON TAGS

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

| | | | | | | | | | |
|-----|--------|--------|--------|--|--|--|--|--|--|
| 827 | | | | | | | | | |
| 828 | | | | | | | | | |
| 829 | | | | | | | | | |
| 830 | | | | | | | | | |
| 831 | | | | | | | | | |
| 832 | | | | | | | | | |
| 833 | | 001100 | | | | | | | |
| 834 | 001100 | | | | | | | | |
| 835 | 001100 | 000000 | | | | | | | |
| 836 | 001102 | 000 | | | | | | | |
| 837 | 001103 | 000 | | | | | | | |
| 838 | 001104 | 000000 | | | | | | | |
| 839 | 001106 | 000000 | | | | | | | |
| 840 | 001110 | 000000 | | | | | | | |
| 841 | 001112 | 000000 | | | | | | | |
| 842 | 001114 | 000 | | | | | | | |
| 843 | 001115 | 001 | | | | | | | |
| 844 | 001116 | 000000 | | | | | | | |
| 845 | 001120 | 000000 | | | | | | | |
| 846 | 001122 | 000000 | | | | | | | |
| 847 | 001124 | 000000 | | | | | | | |
| 848 | 001126 | 000000 | | | | | | | |
| 849 | 001130 | 000000 | | | | | | | |
| 850 | 001132 | 000000 | | | | | | | |
| 851 | 001134 | 000 | | | | | | | |
| 852 | 001135 | 000 | | | | | | | |
| 853 | 001136 | 000000 | | | | | | | |
| 854 | 001140 | 177570 | | | | | | | |
| 855 | 001142 | 177570 | | | | | | | |
| 856 | 001144 | 177560 | | | | | | | |
| 857 | 001146 | 177562 | | | | | | | |
| 858 | 001150 | 177564 | | | | | | | |
| 859 | 001152 | 177566 | | | | | | | |
| 860 | 001154 | 000 | | | | | | | |
| 861 | 001155 | 002 | | | | | | | |
| 862 | 001156 | 012 | | | | | | | |
| 863 | 001157 | 000 | | | | | | | |
| 864 | 001160 | 000000 | | | | | | | |
| 865 | | | | | | | | | |
| 866 | 001162 | 000000 | | | | | | | |
| 867 | 001164 | 000000 | | | | | | | |
| 868 | 001166 | 000000 | | | | | | | |
| 869 | 001170 | 000000 | | | | | | | |
| 870 | 001172 | 000000 | | | | | | | |
| 871 | 001174 | 177607 | 000377 | | | | | | |
| 872 | 001200 | 077 | | | | | | | |
| 873 | 001201 | 015 | | | | | | | |
| 874 | 001202 | 000012 | | | | | | | |
| 875 | | | | | | | | | |
| 876 | 001204 | 000000 | | | | | | | |
| 877 | 001206 | 000000 | | | | | | | |
| 878 | 001210 | 000000 | | | | | | | |
| 879 | 001212 | 000000 | | | | | | | |
| 880 | 001214 | 000000 | | | | | | | |
| 881 | | | | | | | | | |
| 882 | 001216 | 000001 | | | | | | | |

. =1100

```

SCMTAG: .WORD 0          : START OF COMMON TAGS
SPASS: .WORD 0           : CONTAINS PASS COUNT
STSTNM: .BYTE 00         : CONTAINS THE TEST NUMBER
SERFLG: .BYTE 000       : CONTAINS ERROR FLAG
$ICNT: .WORD 0000       : CONTAINS SUBTEST ITERATION COUNT
SLPADR: .WORD 0000      : CONTAINS SCOPE LOOP ADDRESS
SLPERR: .WORD 0000      : CONTAINS SCOPE RETURN FOR ERRORS
SERTTL: .WORD 0000      : CONTAINS TOTAL ERRORS DETECTED
$ITEMB: .BYTE 00        : CONTAINS ITEM CONTROL BYTE
SERMAX: .BYTE 1         : CONTAINS MAX. ERRORS PER TEST
$ERRPC: .WORD 0000      : CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR: .WORD 0000      : CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR: .WORD 0000      : CONTAINS ADDRESS OF 'BAD' DATA
$GDDAT: .WORD 0000      : CONTAINS 'GOOD' DATA
$BDDAT: .WORD 0000      : CONTAINS 'BAD' DATA
          .WORD 0000     : RESERVED--NOT TO BE USED
$AUTOB: .BYTE 00        : AUTOMATIC MODE INDICATOR
$INTAG: .BYTE 00        : INTERRUPT MODE INDICATOR
          .WORD 0
SWR: .WORD DSWR         : ADDRESS OF SWITCH REGISTER
DISPLAY: .WORD DDISP   : ADDRESS OF DISPLAY REGISTER
$TKS: 177560           : TTY KBD STATUS
$TKB: 177562           : TTY KBD BUFFER
$TPS: 177564           : TTY PRINTER STATUS REG. ADDRESS
$TPB: 177566           : TTY PRINTER BUFFER REG. ADDRESS
$NULL: .BYTE 0         : CONTAINS NULL CHARACTER FOR FILLS
$FILLS: .BYTE 2        : CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC: .BYTE 12       : INSERT FILL CHARS. AFTER A "LINE FEED"
$STPFLG: .BYTE 00      : "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
$REGAD: .WORD 0        : CONTAINS THE ADDRESS FROM WHICH ($REGO) WAS OBTAINED
$REGO: .WORD 0         : CONTAINS ((REGAD)+0)
$REG1: .WORD 0         : CONTAINS ((REGAD)+2)
$TMPO: .WORD 0         : USER DEFINED
$TIMES: 0              : MAX. NUMBER OF ITERATIONS
$ESCAPE: 0             : ESCAPE ON ERROR ADDRESS
$BELL: .ASCIZ <207><377><377> : CODE FOR BELL
$QUES: .ASCIZ /?/?     : QUESTION MARK
$CRLF: .ASCII <15>     : CARRIAGE RETURN
$LF: .ASCIZ <12>       : LINE FEED
*****
SAVPC: .WORD 0         : STORAGE FOR THE PC
BYTNUM: .WORD 0        : THE NUMBER OF BYTES "READ" OR "WRITTEN"
RCRC0: .WORD 0         : CRC CALCULATED FOR "WRITE"
RCRC1: .WORD 0         : CRC CALCULATED FOR "READ"
RCRC2: .WORD 0         : CRC FROM THE TAII FOR "READ"
$MXCNT: 1              : MAX. NUMBER OF ITERATIONS

```


| | | | |
|-----|--------|--------|--------|
| 883 | 001220 | 177500 | |
| 884 | 001222 | 177501 | |
| 885 | 001224 | 177502 | |
| 886 | 001226 | 177503 | |
| 887 | 001230 | 000260 | 000262 |
| 888 | 001234 | 000300 | |
| 889 | 001236 | 000000 | |
| 890 | 001240 | 000000 | 000000 |
| 891 | 001244 | 001240 | |
| 892 | 001246 | 000000 | |
| 893 | 001250 | 177777 | 000000 |
| 894 | 001254 | 000000 | |
| 895 | 001256 | 001260 | |
| 896 | 001260 | 000000 | |
| 897 | 001262 | 000000 | |
| 898 | 001264 | 000000 | 000000 |

TACSL: 177500
TACSH: 177501
TADBL: 177502
TADBH: 177503
TAVEC: 260,262
TAPRIO: 300
DRIVE: 0
DRVKEY: 0,0
DRVPT: DRVKEY
ASKKEY: 0
CURDRV: -1,0
WAITKEY: 0
RWDFLG: RWDA
RWDA: 0
RWDB: 0
STALL: 0,0

;LOW BYTE ADDRESS OF TACS
;HIGH BYTE ADDRESS OF TACS
;LOW BYTE ADDRESS OF TADG
;HIGH BYTE ADDRESS OF TADB
;TA11 VECTOR ADDRESS
;TA11 BR LEVEL 6
;NEXT DRIVE TO TEST
;DRIVE SELECT KEY:

;CURRENT DRIVE BEING TESTED
;WAIT ON INTERRUPT KEY
;REWIND FLAG POINTER
;DRIVE "A" REWIND FLAG
;DRIVE "B" REWIND FLAG
;STALL TIME

899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954

.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 SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 ;;*NOTE1: IF SITEMB 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

\$ERRTB:
 ;NOTE: ALL NUMBERS WILL BE TYPED AS 6 DIGIT OCTAL NUMBERS

;ITEM 1
 EM1 ;:IMPROPER FLAG SETTING
 DH1 ;:TEST ERROR
 ;PC PC TACS TADB
 DT1 ;:SAVPC \$ERRPC \$REGO \$REG1
 0

;ITEM 2
 EM2 ;:IMPROPER FLAG OCCURRED
 DH1 ;:TEST ERROR
 ;PC PC TACS TADB
 DT1 ;:SAVPC \$ERRPC \$REGO \$REG1
 0

;ITEM 3
 EM3 ;:MISSED A FLAG
 DH1 ;:TEST ERROR
 ;PC PC TACS TADB
 DT1 ;:SAVPC \$ERRPC \$REGO \$REG1
 0

;ITEM 4
 EM4 ;:INTERRUPT FAILED
 DH1 ;:TEST ERROR
 ;PC PC TACS TADB
 DT1 ;:SAVPC \$ERRPC \$REGO \$REG1
 0

;ITEM 5
 EM5 ;:PREMATURE READY OCCURRED
 DH1 ;:TEST ERROR
 ;PC PC TACS TADB
 DT1 ;:SAVPC \$ERRPC \$REGO \$REG1
 0

;ITEM 6
 EM6 ;:DIDN'T STOP IN A FILE GAP
 DH1 ;:TEST ERROR
 ;PC PC TACS TADB

001270

001270 015540
 001272 016102
 001274 015044
 001276 000000

001300 015566
 001302 016102
 001304 015044
 001306 000000

001310 015615
 001312 016102
 001314 015044
 001316 000000

001320 015633
 001322 016102
 001324 015044
 001326 000000

001330 015654
 001332 016102
 001334 015044
 001336 000000

001340 015705
 001342 016102


```

1011 : ////////////////////////////////////////////////////////////////////
1012 : ////////////////////////////////////////////////////////////////////
1013 : *****
1014 :
1015 : BEGIN1 IS FOR NORMAL START
1016 : BEGIN2 IS FOR DRIVE SELECTION
1017 : BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
1018 : BEGIN4 IS FOR MANUAL OPERATION
1019 :
1020 : *****
1021 :
1022 001450 005005 BEGIN1: CLR R5 ;NORMAL START
1023 001452 012737 041101 001240 MOV #AB, @DRVKEY
1024 001460 122737 000005 000041 CMPB #5, @41 ;CASSETTE DDP?
1025 001466 001015 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
1026 001470 022737 000260 001230 CMP #260, @TAVEC ;STANDARD VECTOR?
1027 001476 001011 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
1028 001500 000403 BR BEGIN3 ;GET DRIVES AND ADDRESSES
1029 001502 012705 000001 BEGIN2: MOV #1, R5 ;ASK FOR DRIVES FLAG
1030 001506 000405 BR BGNCMN ;BEGIN COMMON CODE
1031 001510 012705 000002 BEGIN3: MOV #2, R5 ;ASK FOR DRIVES AND ADDRESSES
1032 001514 000402 BR BGNCMN
1033 001516 012705 000003 BEGIN4: MOV #3, R5
1034 001522 BGNCMN:
1035 .SBTTL INITIALIZE THE COMMON TAGS
1036 ;; CLEAR THE COMMON TAGS (SCMTAG) AREA
1037 001522 012706 001100 MOV #SCMTAG, R6 ;; FIRST LOCATION TO BE CLEARED
1038 001526 005026 CLR (R6)+ ;; CLEAR MEMORY LOCATION
1039 001530 022706 001140 CMP #SWR, R6 ;; DONE?
1040 001534 001374 BNE -6 ;; LOOP BACK IF NO
1041 001536 012706 001100 MOV #STACK, SP ;; SETUP THE STACK POINTER
1042 ;; INITIALIZE A FEW VECTORS
1043 001542 012737 006570 000020 MOV #SCOPE, @IOTVEC ;; IOT VECTOR FOR SCOPE ROUTINE
1044 001550 012737 000340 000022 MOV #340, @IOTVEC+2 ;; LEVEL 7
1045 001556 012737 007046 000030 MOV #ERROR, @EMTVEC ;; EMT VECTOR FOR ERROR ROUTINE
1046 001564 012737 000340 000032 MOV #340, @EMTVEC+2 ;; LEVEL 7
1047 001572 012737 014554 000034 MOV #TRAP, @TRAPVEC ;; TRAP VECTOR FOR TRAP CALLS
1048 001600 012737 000340 000036 MOV #340, @TRAPVEC+2; LEVEL 7
1049 001606 012737 014662 000024 MOV #SPWRDN, @PWRVEC ;; POWER FAILURE VECTOR
1050 001614 012737 000340 000026 MOV #340, @PWRVEC+2; LEVEL 7
1051 001622 013737 006406 006400 MOV SENDCT, SEOPCT ;; SETUP END-OF-PROGRAM COUNTER
1052 001630 005037 001170 CLR STIMES ;; INITIALIZE NUMBER OF ITERATIONS
1053 001634 005037 001172 CLR SESCAPE ;; CLEAR THE ESCAPE ON ERROR ADDRESS
1054 001640 112737 000001 001115 MOVB #1, SERMAX ;; ALLOW ONE ERROR PER TEST
1055 001646 012737 001646 001106 MOV #., SLPADR ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
1056 001654 012737 001654 001110 MOV #., SLPERR ;; SETUP THE ERROR LOOP ADDRESS
1057 ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1058 ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1059 001662 013746 000004 MOV @ERRVEC, -(SP) ;; SAVE ERROR VECTOR
1060 001666 012737 001722 000004 MOV #645, @ERRVEC ;; SET UP ERROR VECTOR
1061 001674 012737 177570 001140 MOV #DSWR, SWR ;; SETUP FOR A HARDWARE SWICH REGISTER
1062 001702 012737 177570 001142 MOV #DDISP, DISPLAY ;; AND A HARDWARE DISPLAY REGISTER
1063 001710 022777 177777 177222 CMP #-1, @SWR ;; TRY TO REFERENCE HARDWARE SWR
1064 001716 001012 BNE 66$ ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
1065 AND THE HARDWARE SWR IS NOT = -1
1066 001720 000403 BR 65$ ;; BRANCH IF NO TIMEOUT

```



```

1067 001722 012716 001730      64$:  MOV    #65$, (SP)      ;;SET UP FOR TRAP RETURN
1068 001726 000002                RTI
1069 001730 012737 000176 001140 65$:  MOV    #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1070 001736 012737 000176 001140 66$:  MOV    #DISPREG,DISPLAY
1071 001744 012637 000004                (SP)+, #ERRVEC ;;RESTORE ERROR VECTOR
1072
1073
1074      .SBTTL  TYPE PROGRAM NAME
      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1075 001750 005227 177777      INC    #-1      ;;FIRST TIME?
1076 001754 001030                BNE    HERE     ;;BRANCH IF NO
1077 001756 022737 006432 000042  CMP    #SENDAD, #42  ;;ACT-11?
1078 001764 001424                BEQ    HERE     ;;BRANCH IF YES
1079 001766 104401 002024      TYPE    MSGID     ;;TYPE ASCIZ STRING
1080      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1081 001772 005737 000042      TST    #42      ;;ARE WE RUNNING UNDER XXDP/ACT?
1082 001776 001006                BNE    #67$     ;;BRANCH IF YES
1083 002000 023727 001140 000176  CMP    SWR, #SWREG  ;;SOFTWARE SWITCH REG SELECTED?
1084 002006 001005                BNE    #68$     ;;BRANCH IF NO
1085 002010 104405                GTSWR          ;;GET SOFT-SWR SETTINGS
1086 002012 000403                BR
1087 002014 112737 000001 001134 67$:  MOVB   #1, SAUTOB  ;;SET AUTO-MODE INDICATOR
1088 002022
1089 002022 000405                68$:  BR    HERE     ;;GET OVER THE ASCIZ
1090      ;;MSGID:  .ASCIZ <CRLF>/CZTAD-D/<CRLF>
1091 002036      HERE:

```

1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119

002036 010504
002040 005305
002042 002406
002044 004737 007422
002050 005305
002052 002402
002054 004737 007532

: THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
: R5=3 MANUAL OPERATIONS
: R5=2 ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
: R5=1 ASK FOR DRIVE(S)
: R5=0 DON'T ASK FOR ANYTHING

BEGINX: MOV R5,R4 ; COPY R5
DEC R5 ; ASK FOR DRIVES?
BLT CHKADR ; BR IF NO
JSR PC,@ASKDRV ; GO GET DRIVES TO BE TESTED
DEC R5 ; ASK FOR ADDRESSES?
BLT CHKADR ; BR IF NO
JSR PC,@ASKADR ; GO GET TA11 ADDRESSES

: CHECK THAT "TACS" WILL RESPOND TO ADDRESSING

- I. TIMEOUT OCCURRED
 - A. TYPE ERROR MESSAGE
 - B. EXAMINE R4
 - 1. R4>0 GOTO BEGINX
 - 2. R4=0 EXAMINE (42)
 - A. (42)=0 GOTO BEGINX
 - B. (42)>0 GOTO SENDAD
- II. TIMEOUT DIDN'T OCCUR
 - A. CONTINUE

1127 002060 012737 002076 000004
1128 002066 005000
1129 002070 005777 177124
1130 002074 000402
1131 002076 005200
1132 002100 022626
1133 002102 012737 000006 000004
1134 002110 005700
1135 002112 001412
1136 002114 104201
1137 002116 012705 000002
1138 002122 005704
1139 002124 001344
1140 002126 013700 000042
1141 002132 001741
1142 002134 000137 006432
1143 002140

CHKADR: MOV #1\$,@ERRVEC ; IN CASE OF TIMEOUTS
CLR R0 ; USE AS A SWITCH
TST @TACSL ; SEE IF TA11 RESPONDS
BR 2\$; BR IF NO TIMEOUT
INC R0 ; COME HERE ON TIMEOUT
CMP (SP)+,(SP)+ ; CLEANUP THE STACK
MOV #ERRVEC+2,@ERRVEC ; RESTORE TIMEOUT VECTOR
TST R0 ; DID A TIMEOUT OCCUR?
BEQ 3\$; BR IF NO
ERROR 201 ; TA11 FAILED TO RESPOND
MOV #2,R5 ; DRIVES & ADDRESSES
TST R4 ; OPERATOR INPUTS?
BNE BEGINX ; BR IF YES
MOV @#42,R0 ; GET MONITOR RETURN ADDRESS
BEQ BEGINX ; BR IF NO MONITOR
JMP @SENDAD ; GO TO END
3\$:


```

1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165 002140 012700 001240
1166 002144 004737 006464
1167 002150 000410
1168 002152 116010 000001
1169 002156 001412
1170 002160 004737 006464
1171 002164 000407
1172 002166 005010
1173 002170 000405
1174 002172 005200
1175 002174 004737 006464
1176 002200 000401
1177 002202 105010
1178 002204 012700 001240
1179 002210 010037 001244
1180 002214 121060 000001
1181 002220 001002
1182 002222 105060 000001
1183 002226 005710
1184 002230 001401
1185 002232 000412
1186 002234 104202
1187 002236 012705 000002
1188 002242 005704
1189 002244 001274
1190 002246 013700 000042
1191 002252 001671
1192 002254 000137 006432
1193 002260 020427 000003
1194 002264 001002
1195 002266 013704 177777
1196 002272 010437 001246
1197 002276 000405
1198 002300 104401 002024
1199 002304 012737 001240 001244

```

```

*****
*****

```

MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED

- I. DESIRED DRIVES CAN NOT BE TESTED
 - A. TYPE ERROR MESSAGE
 - B. EXAMINE R4
 - 1. R4>0 GOTO BEGINX
 - 2. R4=0 EXAMINE (42)
 - A. (42)=0 GOTO BEGINX
 - B. (42)>0 GOTO SENDAD
- II. BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
 - A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
 - B. CONTINUE IN PROGRAM
- III. DESIRED DRIVE(S) CAN BE TESTED
 - A. CONTINUE IN PROGRAM

```

*****
CHKDRV: MOV    #DRVKEY,RO      ;PICKUP ADDRESS OF ASCII DRIVE KEY
        JSR    PC,2#EXAM     ;GO EXAMINE FIRST DRIVE
        BR     1$           ;OK TO TEST---GO CHECK NEXT
        MOVB  1(RO),(RO)    ;REPLACE 1ST WITH 2ND
        BEQ   2$           ;BR IF NO 2ND DRIVE SELECTED
        JSR    PC,2#EXAM     ;GO EXAMINE DRIVE
        BR     2$           ;OK TO TEST
        CLR   (RO)         ;CLEAR DRIVE CODES
        BR     2$
1$: INC   RO               ;POINT TO 2ND
        JSR    PC,2#EXAM     ;GO EXAMINE DRIVE
        BR     2$           ;OK TO TEST
        CLRB  (RO)         ;CLEAR 2ND
        BR     2$           ;RESET ADDRESS POINTERS
2$: MOV    #DRVKEY,RO
        MOV    RO,2#DRVPNT
        CMPB  (RO),1(RO)
        BNE  3$           ;1ST = 2ND?
        CLRB  1(RO)       ;BR IF NO
        BR     3$         ;YES---CLEAR 2ND
3$: TST   (RO)           ;ANY DRIVES?
        BEQ   5$         ;BR IF NO
        BR     5$
5$: ERROR 202             ;NO DRIVES AVAILABLE
        MOV    #2,R5      ;DRIVES & ADDRESS
        TST   R4         ;OPERATOR INPUTS?
        BNE  BEGINX      ;BR IF YES
        MOV    2#42,RO    ;GET MONITOR RETURN ADDRESS
        BEQ   BEGINX      ;NO MONITOR
        JMP   2#SENDAD    ;GO TO END
MANUAL: CMP R4,#3
        BNE  OK
        MOV  -1,R4
OK: MOV   R4,2#ASKKEY
        BR   START
PWRST: TYPE MSGID        ;POWER FAIL RESTART
        MOV  #DRVKEY,2#DRVPNT

```

TA11 MOTION TEST
CZTADD.P11

23-MAR-78

CZTAD-D MACY11
08:34

30R(1052) 23-MAR-78 08:36 PAGE 28
GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0028

SEQ 0028

```

1200 002312 012777 012356 176710 START: MOV @BADINT,@TAVEC ;SETUP TA11 TRAP VECTOR
1201 002320 013777 001234 176704 MOV @TAPRIO,@TAVEC+2
1202 002326 012737 000340 177776 MOV @340,@#PS ;LOCKOUT ALL I/O INT
1203 002334 013704 001220 MOV @TACSL,TACS ;SETUP TACS
1204 002340 013705 001224 MOV @TADBL,TADB ;SETUP TADB
1205 002344 005037 001102 CLR @#STSTNA ;ZERO THE TEST NUMBER
1206 002350 005037 006614 CLR @#LOOPKEY ;CLEAR THE LOOP UNDER TTY CONTROL KEY
1207 002354 012737 000001 001216 MOV #1,@#SMXCNT ;SET FOR ONE ITERATION PER TEST
1208 002362 013701 001244 MOV @DRVPNT,R1 ;GET DRIVE POINTER
1209 002366 010137 001236 MOV R1,DRIVE ;SET DRIVE TO POINTER
1210 ////////////////////////////////////////////////////
1211 //TYPE THE DRIVE(S) THAT WILL BE TESTED
1212 002372 111137 001250 MOVB (R1),@#CURDRV ;GET CURRENT DRIVE FOR TYPE-OUT
1213 002376 104401 015261 TYPE #MTSTDRV ;TYPE "TESTING DRIVE"
1214 002380 104401 001250 TYPE #CURDRV ;TYPE THE CURRENT DRIVE
1215 002384 032777 000200 176524 BIT @SW07,@SWR ;LOCK ON ONE DRIVE?
1216 002388 001021 BNE #25 ;BR IF YES
1217 002392 105737 001241 TSTB @#DRVKEY+1 ;TWO DRIVES AVAILABLE?
1218 002396 001416 BEQ #25 ;BR IF NO
1219 002400 005237 001216 INC @#SMXCNT ;SET FOR TWO ITERATIONS PER TEST (ONE PER DRIVE)
1220 002404 005201 INC R1 ;INCREMENT THE POINTER
1221 002408 111137 001250 MOVB (R1),@#CURDRV ;SET SECOND DRIVE FOR TYPE-OUT
1222 002412 001004 BNE #15 ;GO TYPE IT
1223 002416 012701 001240 MOV @DRVKEY,R1 ;RESET POINTER
1224 002420 111137 001250 MOVB (R1),@#CURDRV ;NOW GET THE SECOND DRIVE
1225 002424 104401 015302 15: TYPE #MANDRV ;TYPE "AND DRIVE"
1226 002428 104401 001250 TYPE #CURDRV ;TYPE DRIVE
1227 002432 010137 001244 25: MOV R1,@DRVPNT ;SAVE POINTER FOR NEXT TRIP THRU
1228 002436 012737 002664 001106 MOV @55,@#SLPADR ;SETUP THE SCOPE LOOP ADDRESS
1229 002440 005737 001100 TST @#PASS ;ONCE ONLY
1230 002444 001131 BNE #TST1
1231 ////////////////////////////////////////////////////
1232 //ON THE FIRST PASS OF THE PROGRAM DETERMINE THE AMOUNT OF TIME TO
1233 //WAIT ON A FLAG ("READY"/"TRANSFER REQUEST") AND THE STALL TIME TO USE
1234 //WHEN STALLING AFTER A FUNCTION IS COMPLETED
1235 002500 104422 SELDRV ;SELECT DRIVE
1236 002504 012737 002702 001172 MOV @#FATAL,@#SESCAPE ;SETUP ESCAPE ON ERROR
1237 002508 005077 176542 CLR @#RWDFLG ;SET FOR NO REWIND ERRORS
1238 002512 012737 077777 012012 MOV @#CBIT15,@#MAXCNT ;SETUP IMPOSSIBLE COUNT
1239 002516 112714 000016 MOVB @#XRWIND,@TACS ;INSURE NO ERROR DUE TO "CLEAR LEADER"
1240 002520 104421 REWIND ;GO TO BOT
1241 002524 104412 WFG ;NOW TIME A WFG
1242 002528 163737 012010 012012 SUB @#WHICNT,@#MAXCNT ;GET THE TIME IT TOOK TO WFG
1243 002532 005237 012012 INC @#MAXCNT ;MAKE IT BIGGER
1244 002536 006137 012012 ROL @#MAXCNT
1245 002540 006137 012012 ROL @#MAXCNT
1246 002544 005000 CLR R0
1247 002548 005001 CLR R1
1248 002552 012777 002616 176442 MOV @45,@TAVEC ;SETUP TA11 VECTOR
1249 002556 112714 000101 MOVB @#XWFG!INT.EN!GO,@TACS ;START A "WFG"
1250 002560 005037 177776 CLR @#PS ;ALLOW INTERRUPTS
1251 002564 162700 000001 35: SUB #1,R0 ;START TIMING HOW LONG
1252 002568 005601 SBC R1 ;IT TAKES TO WRITE A FILE GAP
1253 002572 001374 BNE #35
1254 002576 012737 000340 177776 MOV @340,@#PS ;LOCK OUT INTERRUPTS
1255 002580 000432 BR #FATAL ;IT TOOK TO LONG

```



```

1256 002616 006201          4$: ASR      R1          ;DIVIDE THE TIME BY 2
1257 002620 006000          ROR      R0
1258 002622 010037 001264    MOV      R0, @#STALL ;AND SAVE IT AS THE STALL TIME
1259 002624 010137 001266    MOV      R1, @#STALL+2
1260 002632 012777 012356 176370 MOV      @BADINT, @TAVEC ;SET TRAP CATCHER
1261 002640 105737 001241    TSTB    @#DRVKEY+1 ;TWO DRIVES?
1262 002644 001407          BEQ      SS          ;BR IF NO
1263 002646 104422          SELDRV ;SELECT THE OTHER DRIVE
1264 002650 005077 176402    CLR      @RWDFLG    ;SET REWIND FLAG
1265 002654 112714 000016    MOVB    @XRWD, @TACS ;AVOID "CLEAR LEADER" ERROR
1266 002660 104421          REWIND ;GO TO BOT
1267 002662 104412          WFG      ;GO TO OXIDE
1268 002664 005737 001246    5$: TST      @#ASKKEY ;GO START TESTING IF NO MANUAL
1269 002670 002034          BGE     TST1      ; OPERATIONS REQUESTED
1270 002672 005000          CLR      R0
1271 002674 006137 012012    ROL     @#MAXCNT   ;INCREASE THE WAIT TIME
1272 002700 000000          HALT    ;GIVE CONTROL TO THE OPERATOR
1273
1274 002702
1275 002702 104401 002710    FATAL: TYPE      65$      ;;TYPE ASCIZ STRING
1276 002706 000414          BR      64$      ;;GET OVER THE ASCIZ
1277
1278 002740          ;;65$: .ASCIZ <15><12>*FATAL ERROR ON DRIVE *
1279 002740 104401 001250    64$: TYPE      CURDRV   ;CURRENT DRIVE
1280 002744 013700 000042    MOV     @#42, R0    ;CHECK FOR A MONITOR
1281 002750 001402          BEQ     1$         ;BR IF NONE
1282 002752 000137 006432    JMP     @#SENDAD   ;LEAVE
1283 002756 000000          1$: HALT    ;ERROR OCCURRED BEFORE TESTING STARTED
1284 002760 000776          BR      1$         ;HANGUP
    
```

E03

```

1285
1286
1287
1288 002762 000004
1289 002764 012737 003016 001172
1290 002772 104422
1291 002774 104421
1292 002776 104413 017012
1293 003002 104421
1294 003004 104414 016756
1295 003010 104423
1296 003012 016756
1297 003014 017012
1298
1299
1300
1301 003016 000004
1302 003020 012737 003052 001172
1303 003026 104422
1304 003030 104421
1305 003032 104413 017012
1306 003036 104416
1307 003040 104414 016756
1308 003044 104423
1309 003046 016756
1310 003050 017012
1311
1312
1313
1314 003052 000004
1315 003054 012737 003106 001172
1316 003062 104422
1317 003064 104421
1318 003066 104413 017012
1319 003072 104415
1320 003074 104414 016756
1321 003100 104423
1322 003102 016756
1323 003104 017012
1324
1325
1326
1327 003106 000004
1328 003110 012737 003156 001172
1329 003116 104422
1330 003120 104421
1331 003122 104413 017012
1332 003126 104421
1333 003130 104414 016756
1334 003134 104423
1335 003136 016756
1336 003140 017012
1337 003142 104416
1338 003144 104414 016756
1339 003150 104423
1340 003152 016756

```

```

*****
*TEST 1 REWIND,WRITE,REWIND,READ
*****
†ST1: SCOPE
MOV #TST2,$ESCAPE ;;ESCAPE TO TEST 2 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO "CLEAR LEADER"
WRITE ,WLNK1 ;WRITE ON TAPE
REWIND ;GO TO BEGINNING OF TAPE
READ ,RLNK1 ;READ
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER
*****
*TEST 2 TEST BSBG AFTER WRITE AND READ AFTER BSBG
*****
†ST2: SCOPE
MOV #TST3,$ESCAPE ;;ESCAPE TO TEST 3 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO "BOT"
WRITE ,WLNK1 ;WRITE ONE BLOCK
BSBG ;BACK OVER THE BLOCK
READ ,RLNK1 ;NOW READ
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER
*****
*TEST 3 TEST BSFG AFTER WRITE AND READ AFTER BSFG
*****
†ST3: SCOPE
MOV #TST4,$ESCAPE ;;ESCAPE TO TEST 4 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO "BOT"
WRITE ,WLNK1 ;WRITE ONE BLOCK
BSFG ;BACK OVER THE BLOCK
READ ,RLNK1 ;NOW READ
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER
*****
*TEST 4 TEST BSBG AFTER READ
*****
†ST4: SCOPE
MOV #TST5,$ESCAPE ;;ESCAPE TO TEST 5 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO "BOT"
WRITE ,WLNK1 ;WRITE ONE BLOCK
REWIND ;GO TO "BOT"
READ ,RLNK1 ;READ
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER
BSBG ;BACK UP
READ ,RLNK1 ;READ
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER

```



```

1341 003154 017012          WLNK1          ;LINK TO THE WRITE BUFFER
1342          ;*****
1343          ;*TEST 5          TEST BSBG AFTER READ
1344          ;*****
1345          ;TST5:
1346 003156 000004          SCOPE
1347 003160 012737 003226 001172  MOV      #TST6,$ESCAPE ;;ESCAPE TO TEST 6 ON ERROR
1348 003166 104422          SELDRV          ;SELECT DRIVE FOR TESTING
1349 003172 104413 017012  REWIND          ;GO TO "BOT"
1350 003176 104421          WRITE      ,WLNK1 ;WRITE A BLOCK
1351 003200 104414 016756  REWIND          ;GO TO "BOT"
1352 003204 104423          READ       ,RLNK1 ;READ A BLOCK
1353 003206 016756          BLKCMP          ;COMPARE THE READ AND WRITE BUFFERS
1354 003210 017012          RLNK1          ;LINK TO THE READ BUFFER
1355 003212 104415          WLNK1          ;LINK TO THE WRITE BUFFER
1356 003214 104414 016756  BSBG           ;BACK OVER THE DATA
1357 003220 104423          READ       ,RLNK1 ;NOW READ THE DATA
1358 003222 016756          BLKCMP          ;COMPARE THE READ AND WRITE BUFFERS
1359 003224 017012          RLNK1          ;LINK TO THE READ BUFFER
1360          WLNK1          ;LINK TO THE WRITE BUFFER
1361          ;*****
1362          ;*TEST 6          TEST SFBG AFTER REWIND AND BSBG AFTER SFBG
1363          ;*****
1364 003226 000004          ;TST6: SCOPE
1365 003230 012737 003266 001172  MOV      #TST7,$ESCAPE ;;ESCAPE TO TEST 7 ON ERROR
1366 003236 104422          SELDRV          ;SELECT DRIVE FOR TESTING
1367 003240 104421          REWIND          ;GO TO "BOT"
1368 003242 104413 017012  WRITE      ,WLNK1 ;WRITE ONE BLOCK OF DATA
1369 003246 104421          REWIND          ;GO TO "BOT"
1370 003250 104420          SFBG           ;SPACE OVER THE DATA
1371 003252 104415          BSBG           ;BACK OVER THE DATA
1372 003254 104414 016756  READ       ,RLNK1 ;READ THE BLOCK
1373 003260 104423          BLKCMP          ;COMPARE THE READ AND WRITE BUFFERS
1374 003262 016756          RLNK1          ;LINK TO THE READ BUFFER
1375 003264 017012          WLNK1          ;LINK TO THE WRITE BUFFER
1376          ;*****
1377          ;*TEST 7          TEST WRITE AFTER READ
1378          ;*****
1379 003266 000004          ;TST7: SCOPE
1380 003270 012737 003342 001172  MOV      #TST10,$ESCAPE ;;ESCAPE TO TEST 10 ON ERROR
1381 003276 104422          SELDRV          ;SELECT DRIVE FOR TESTING
1382 003300 104421          REWIND          ;GO TO "CLEAR LEADER"
1383 003302 104413 017012  WRITE      ,WLNK1 ;WRITE ONE BLOCK
1384 003306 104416 016756  BSBG           ;BACK OVER THE DATA BLOCK
1385 003310 104414          READ       ,RLNK1 ;AND READ IT
1386 003314 104423          BLKCMP          ;COMPARE THE READ AND WRITE BUFFERS
1387 003316 016756          RLNK1          ;LINK TO THE READ BUFFER
1388 003320 017012          WLNK1          ;LINK TO THE WRITE BUFFER
1389 003322 104413 017016  WRITE      ,WLNK2 ;WRITE A SECOND BLOCK
1390 003326 104416          BSBG           ;BACK OVER IT
1391 003330 104414 016762  READ       ,RLNK2 ;AND READ IT
1392 003334 104423          BLKCMP          ;COMPARE THE READ AND WRITE BUFFERS
1393 003336 016762          RLNK2          ;LINK TO THE READ BUFFER
1394 003340 017016          WLNK2          ;LINK TO THE WRITE BUFFER
1395          ;*****
1396          ;*TEST 10         TEST WRITE AFTER WRITE AND READ AFTER READ
1397          ;*****

```

```

1397 003342 000004
1398 003344 012737 003406 001172 TST10: SCOPE
1399 003352 104422          MOV      #TST11,$ESCAPE ;;ESCAPE TO TEST 11 ON ERROR
1400 003354 104421          SELDRV   ;SELECT DRIVE FOR TESTING
          REWIND   ;GO TO "CLEAR LEADER"
1401 003356 104413 017012          WRITE   ,WLNK1 ;WRITE TWO(2) BLOCKS
1402 003362 104413 017016          WRITE   ,WLNK2
1403 003366 104421          REWIND   ;GO TO BOT
1404 003370 104414 016756          READ    ,RLNK1 ;READ BOTH BLOCKS
1405 003374 104414 016762          READ    ,RLNK2
1406 003400 104423          BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
1407 003402 016762          RLNK2    ;LINK TO THE READ BUFFER
1408 003404 017016          WLNK2    ;LINK TO THE WRITE BUFFER
1409
1410 ;*****
1411 ;*TEST 11 TEST BSBG AFTER BSBG
1412 ;*****
1412 003406 000004
1413 003410 012737 003450 001172 TST11: SCOPE
1414 003416 104422          MOV      #TST12,$ESCAPE ;;ESCAPE TO TEST 12 ON ERROR
1415 003420 104421          SELDRV   ;SELECT DRIVE FOR TESTING
          REWIND   ;GO TO "CLEAR LEADER"
1416 003422 104413 017012          WRITE   ,WLNK1 ;WRITE TWO(2) BLOCKS
1417 003426 104413 017016          WRITE   ,WLNK2
1418 003432 104416          BSBG     ;BACK OVER THE 2ND BLOCK
1419 003434 104416          BSBG     ;BACK OVER THE 1ST BLOCK
1420 003436 104414 016756          READ    ,RLNK1 ;READ THE 1ST BLOCK
1421 003442 104423          BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
1422 003444 016756          RLNK1    ;LINK TO THE READ BUFFER
1423 003446 017012          WLNK1    ;LINK TO THE WRITE BUFFER
1424
1425 ;*****
1426 ;*TEST 12 TEST READ AFTER SFBG
1427 ;*****
1427 003450 000004
1428 003452 012737 003512 001172 TST12: SCOPE
1429 003460 104422          MOV      #TST13,$ESCAPE ;;ESCAPE TO TEST 13 ON ERROR
1430 003462 104421          SELDRV   ;SELECT DRIVE FOR TESTING
          REWIND   ;GO TO "CLEAR LEADER"
1431 003464 104413 017012          WRITE   ,WLNK1 ;WRITE TWO(2) BLOCKS
1432 003470 104413 017016          WRITE   ,WLNK2
1433 003474 104421          REWIND   ;BOT
1434 003476 104420          SFBG     ;SPACE OVER THE 1ST BLOCK
1435 003500 104414 016762          READ    ,RLNK2 ;READ THE 2ND BLOCK
1436 003504 104423          BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
1437 003506 016762          RLNK2    ;LINK TO THE READ BUFFER
1438 003510 017016          WLNK2    ;LINK TO THE WRITE BUFFER
1439
1440 ;*****
1441 ;*TEST 13 TEST SFBG AFTER BSBG
1442 ;*****
1442 003512 000004
1443 003514 012737 003556 001172 TST13: SCOPE
1444 003522 104422          MOV      #TST14,$ESCAPE ;;ESCAPE TO TEST 14 ON ERROR
1445 003524 104421          SELDRV   ;SELECT DRIVE FOR TESTING
          REWIND   ;GO TO "CLEAR LEADER"
1446 003526 104413 017012          WRITE   ,WLNK1 ;WRITE TWO(2) BLOCKS
1447 003532 104413 017016          WRITE   ,WLNK2
1448 003536 104416          BSBG     ;BACK SPACE OVER BOTH BLOCKS
1449 003540 104416          BSBG
1450 003542 104420          SFBG     ;SPACE FWD OVER 1ST BLOCK
1451 003544 104414 016762          READ    ,RLNK2 ;READ 2ND BLOCK
1452 003550 104423          BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS

```



```

1453 003552 016762          RLNK2          ;LINK TO THE READ BUFFER
1454 003554 017016          WLNK2          ;LINK TO THE WRITE BUFFER
1455          ;*****
1456          ;*TEST 14          TEST SFBG AFTER BSBG
1457          ;*****
1458 003556 000004          †ST14: SCOPE
1459 003560 012737 003620 001172 MOV          #TST15,$ESCAPE ;;ESCAPE TO TEST 15 ON ERROR
1460 003566 104422          SELDRV        ;SELECT DRIVE FOR TESTING
1461 003570 104421          REWIND        ;GO TO "CLEAR LEADER"
1462 003572 104413 017012 WRITE        ,WLNK1 ;WRITE TWO BLOCKS
1463 003576 104413 017016 WRITE        ,WLNK2
1464 003602 104415          BSBG          ;SPACE REV FILE
1465 003604 104420          SFBG          ;SPACE FWD BLOCK
1466 003606 104414 016762 READ          ,RLNK2 ;READ THE 2ND BLOCK
1467 003612 104423          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1468 003614 016762          RLNK2          ;LINK TO THE READ BUFFER
1469 003616 017016          WLNK2          ;LINK TO THE WRITE BUFFER
1470          ;*****
1471          ;*TEST 15          REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER WRITE
1472          ;*****
1473 003620 000004          †ST15: SCOPE
1474 003622 012737 004012 001172 MOV          #TST16,$ESCAPE ;;ESCAPE TO TEST 16 ON ERROR
1475 003630 104422          SELDRV        ;SELECT DRIVE FOR TESTING
1476 003632 104421          REWIND        ;GO TO "CLEAR LEADER"
1477 003634 104413 017012 WRITE        ,WLNK1 ;WRITE 2 BLOCKS
1478 003640 104413 017016 WRITE        ,WLNK2
1479 003644 104416          BSBG          ;BACK OVER BOTH OF THEM
1480 003646 104416          BSBG
1481 003650 104414 016756 READ          ,RLNK1 ;READ THE FIRST ONE
1482 003654 104423          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1483 003656 016756          RLNK1          ;LINK TO THE READ BUFFER
1484 003660 017012          WLNK1          ;LINK TO THE WRITE BUFFER
1485 003662 104414 016762 READ          ,RLNK2 ;NOW READ THE 2ND ONE
1486 003666 104423          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1487 003670 016762          RLNK2          ;LINK TO THE READ BUFFER
1488 003672 017016          WLNK2          ;LINK TO THE WRITE BUFFER
1489 003674 104416          BSBG          ;BACK OVER THE 2ND ONE
1490 003676 104413 017022 WRITE        ,WLNK3 ;WRITE OVER THE 2ND BLOCK
1491 003702 104416          BSBG          ;NOW BACK UP AND
1492 003704 104414 016766 READ          ,RLNK3 ;READ IT
1493 003710 104423          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1494 003712 016766          RLNK3          ;LINK TO THE READ BUFFER
1495 003714 017022          WLNK3          ;LINK TO THE WRITE BUFFER
1496 003716 104416          BSBG          ;BACK OVER IT AGAIN
1497 003720 104413 017026 WRITE        ,WLNK4 ;AND WRITE OVER IT AGAIN
1498 003724 104416          BSBG          ;BACK UP AND
1499 003726 104414 016772 READ          ,RLNK4 ;READ THE BLOCK
1500 003732 104423          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1501 003734 016772          RLNK4          ;LINK TO THE READ BUFFER
1502 003736 017026          WLNK4          ;LINK TO THE WRITE BUFFER
1503 003740 104416          BSBG          ;BACK OVER THE BLOCK AGAIN
1504 003742 104413 017016 WRITE        ,WLNK2 ;AND WRITE OVER IT AGAIN
1505 003746 104416          BSBG          ;NOW BACK UP
1506 003750 104414 016762 READ          ,RLNK2 ;AND READ IT
1507 003754 104423          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1508 003756 016762          RLNK2          ;LINK TO THE READ BUFFER

```

```

1509 003760 017016          WLNK2          ;LINK TO THE WRITE BUFFER
1510 003762 104416          BSBG          ;BACK OVER BOTH BLOCKS
1511 003764 104416          BSBG
1512 003766 104414 016756  READ          ,RLNK1      ;AND READ THE 1ST ONE
1513 003772 104423          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1514 003774 016756          RLNK1         ;LINK TO THE READ BUFFER
1515 003776 017012          WLNK1         ;LINK TO THE WRITE BUFFER
1516 004000 104414 016762  READ          ,RLNK2      ;AND THE 2ND ONE
1517 004004 104423          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1518 004006 016762          RLNK2         ;LINK TO THE READ BUFFER
1519 004010 017016          WLNK2         ;LINK TO THE WRITE BUFFER
1520
1521          ;*****
1522          ;*TEST 16 REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER READ
1523          ;*****
1524          ;*ST16: SCOPE
1525          MOV          #TST17,$ESCAPE ;;ESCAPE TO TEST 17 ON ERROR
1526          SELDRV        ;SELECT DRIVE FOR TESTING
1527          REWIND        ;GO TO "CLEAR LEADER"
1528          WRITE        ,WLNK1      ;WRITE A BLOCK OF DATA
1529          BSBG          ;BACK OVER IT
1530          READ          ,RLNK1      ;AND READ IT
1531          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1532          RLNK1         ;LINK TO THE READ BUFFER
1533          WLNK1         ;LINK TO THE WRITE BUFFER
1534          WRITE        ,WLNK2      ;WRITE 2ND BLOCK
1535          BSBG          ;BACK OVER IT
1536          READ          ,RLNK2      ;AND READ IT
1537          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1538          RLNK2         ;LINK TO THE READ BUFFER
1539          WLNK2         ;LINK TO THE WRITE BUFFER
1540          BSBG          ;BACK OVER IT AND
1541          WRITE        ,WLNK4      ;WRITE OVER 2ND BLOCK
1542          BSBG          ;BACK UP AND
1543          READ          ,RLNK4      ;READ IT
1544          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1545          RLNK4         ;LINK TO THE READ BUFFER
1546          WLNK4         ;LINK TO THE WRITE BUFFER
1547          BSBG          ;BACK OVER IT AGAIN
1548          WRITE        ,WLNK5      ;REWRITE 2ND BLOCK
1549          BSBG
1550          READ          ,RLNK5
1551          BLKCMP        ;READ 2ND BLOCK
1552          RLNK5         ;COMPARE THE READ AND WRITE BUFFERS
1553          WLNK5         ;LINK TO THE READ BUFFER
1554          BSBG          ;LINK TO THE WRITE BUFFER
1555          WRITE        ,WLNK3      ;BACK OVER IT AGAIN
1556          BSBG          ;WRITE ANOTHER BLOCK OVER IT
1557          BSBG          ;BACK OVER 2 BLOCKS
1558          READ          ,RLNK1      ;READ THE 1ST BLOCK
1559          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1560          RLNK1         ;LINK TO THE READ BUFFER
1561          WLNK1         ;LINK TO THE WRITE BUFFER
1562          READ          ,RLNK3      ;READ 2ND BLOCK
1563          BLKCMP        ;COMPARE THE READ AND WRITE BUFFERS
1564          RLNK3         ;LINK TO THE READ BUFFER
1564          WLNK3         ;LINK TO THE WRITE BUFFER

```



```

1565
1566
1567
1568 004170 000004
1569 004172 012737 004226 001172
1570 004200 104422
1571 004202 104421
1572 004204 104413 017012
1573 004210 104412
1574 004212 104415
1575 004214 104414 016756
1576 004220 104423
1577 004222 016756
1578 004224 017012
1579
1580
1581
1582 004226 000004
1583 004230 012737 004264 001172
1584 004236 104422
1585 004240 104421
1586 004242 104413 017012
1587 004246 104412
1588 004250 104416
1589 004252 104414 016756
1590 004256 104423
1591 004260 016756
1592 004262 017012
1593
1594
1595
1596 004264 000004
1597 004266 012737 004342 001172
1598 004274 104422
1599 004276 104421
1600 004300 104413 017012
1601 004304 104421
1602 004306 104414 016756
1603 004312 104423
1604 004314 016756
1605 004316 017012
1606 004320 104412
1607 004322 104413 017016
1608 004326 104416
1609 004330 104414 016762
1610 004334 104423
1611 004336 016762
1612 004340 017016
1613
1614
1615
1616 004342 000004
1617 004344 012737 004420 001172
1618 004352 104422
1619 004354 104421
1620 004356 104413 017012

```

```

*****
*TEST 17 TEST WFG AFTER WRITE AND BSFG AFTER WFG
*****
↑ST17: SCOPE
MOV #TST20,$ESCAPE ;;ESCAPE TO TEST 20 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO "BOT"
WRITE ,WLNK1 ;WRITE ONE BLOCK
WFG ;FOLLOWED BY A FILE GAP
BSFG ;BACK OVER THE DATA BLOCK
READ ,RLNK1 ;READ THE DATA
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER
*****
*TEST 20 TEST BSBG AFTER WFG
*****
↑ST20: SCOPE
MOV #TST21,$ESCAPE ;;ESCAPE TO TEST 21 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO BOT
WRITE ,WLNK1 ;WRITE ONE BLOCK OF DATA
WFG ;WRITE A FILE GAP
BSBG ;BACK OVER THE DATA
READ ,RLNK1 ;READ ONE BLOCK
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER
*****
*TEST 21 TEST WRITE AFTER WFG
*****
↑ST21: SCOPE
MOV #TST22,$ESCAPE ;;ESCAPE TO TEST 22 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO BOT
WRITE ,WLNK1 ;WRITE ONE BOCK
REWIND ;GO TO BOT
READ ,RLNK1 ;READ ONE BLOCK
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER
WFG ;WRITE A FILE GAP
WRITE ,WLNK2 ;WRITE ANOTHER BLOCK
BSBG ;BACK OVER THE LAST BLOCK
READ ,RLNK2 ;READ THE SECOND BLOCK
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK2 ;LINK TO THE READ BUFFER
WLNK2 ;LINK TO THE WRITE BUFFER
*****
*TEST 22 TEST WFG AFTER READ AND BSFG AFTER WRITE
*****
↑ST22: SCOPE
MOV #TST23,$ESCAPE ;;ESCAPE TO TEST 23 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO "BOT"
WRITE ,WLNK1 ;WRITE ONE BLOCK

```

```

1621 004362 104421 REWIND ;GO TO "BOT"
1622 004364 104414 016756 READ ,RLNK1 ;READ ONE BLOCK
1623 004370 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1624 004372 016756 RLNK1 ;LINK TO THE READ BUFFER
1625 004374 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1626 004376 104412 WFG ;WRITE A FILE GAP
1627 004400 104413 017016 WRITE ,WLNK2 ;WRITE A BLOCK
1628 004404 104415 BSFG ;BACK OVER THE DATA
1629 004406 104414 016762 READ ,RLNK2 ;READ SECOND BLOCK
1630 004412 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1631 004414 016762 RLNK2 ;LINK TO THE READ BUFFER
1632 004416 017016 WLNK2 ;LINK TO THE WRITE BUFFER
1633 *****
1634 ;*TEST 23 TEST SFFG AFTER REWIND AND READ AFTER SFFG
1635 *****
1636 004420 000004 †ST23: SCOPE
1637 004422 012737 004464 001172 MOV #TST24,$ESCAPE ;;ESCAPE TO TEST 24 ON ERROR
1638 004430 104422 SELDRV ;SELECT DRIVE FOR TESTING
1639 004432 104421 REWIND ;GO TO "BOT"
1640 004434 104413 017012 WRITE ,WLNK1 ;WRITE A BLOCK OF DATA
1641 004440 104412 WFG ;WRITE A FILE GAP
1642 004442 104413 017016 WRITE ,WLNK2 ;WRITE A BLOCK OF DATA
1643 004446 104421 REWIND ;GO TO "BOT"
1644 004450 104417 SFFG ;SPACE OVER FIRST BLOCK
1645 004452 104414 016762 READ ,RLNK2 ;READ THE 2ND BLOCK
1646 004456 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1647 004460 016762 RLNK2 ;LINK TO THE READ BUFFER
1648 004462 017016 WLNK2 ;LINK TO THE WRITE BUFFER
1649 *****
1650 ;*TEST 24 TEST BSBG AFTER SFFG
1651 *****
1652 004464 000004 †ST24: SCOPE
1653 004466 012737 004532 001172 MOV #TST25,$ESCAPE ;;ESCAPE TO TEST 25 ON ERROR
1654 004474 104422 SELDRV ;SELECT DRIVE FOR TESTING
1655 004476 104421 REWIND ;GO TO "BOT"
1656 004500 104413 017012 WRITE ,WLNK1 ;WRITE DATA
1657 004504 104412 WFG ;WRITE FILE GAP
1658 004506 104413 017016 WRITE ,WLNK2 ;WRITE DATA
1659 004512 104421 REWIND ;GO TO "BOT"
1660 004514 104417 SFFG ;SPACE OVER FIRST FILE
1661 004516 104416 BSBG ;BACKUP TO FIRST BLOCK
1662 004520 104414 016756 READ ,RLNK1 ;READ THE BLOCK
1663 004524 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1664 004526 016756 RLNK1 ;LINK TO THE READ BUFFER
1665 004530 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1666 *****
1667 ;*TEST 25 TEST SFBG AFTER SFFG
1668 *****
1669 004532 000004 †ST25: SCOPE
1670 004534 012737 004606 001172 MOV #TST26,$ESCAPE ;;ESCAPE TO TEST 26 ON ERROR
1671 004542 104422 SELDRV ;SELECT DRIVE FOR TESTING
1672 004544 104421 REWIND ;GO TO "BOT"
1673 004546 104413 017012 WRITE ,WLNK1 ;WRITE A BLOCK OF DATA
1674 004552 104412 WFG ;FOLLOWED BY A FILE GAP
1675 004554 104413 017016 WRITE ,WLNK2 ;FOLLOWED BY 2 BLOCKS OF DATA
1676 004560 104413 017022 WRITE ,WLNK3

```



```

1677 004564 104421 REWIND ;GO TO "BOT"
1678 004566 104417 SFBG ;SPACE OVER 1ST FILE
1679 004570 104420 SFBG ;SPACE OVER BLOCK
1680 004572 104416 BSBG ;BACK OVER THE BLOCK
1681 004574 104414 016762 READ ,RLNK2 ;READ THE LAST BLOCK
1682 004600 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1683 004602 016762 RLNK2 ;LINK TO THE READ BUFFER
1684 004604 017016 WLNK2 ;LINK TO THE WRITE BUFFER
1685
1686 *****
1687 *TEST 26 TEST SFBG AFTER SFBG *****
1688 004606 000004 †ST26: SCOPE
1689 004610 012737 004666 001172 MOV #TST27,$ESCAPE ; ; ESCAPE TO TEST 27 ON ERROR
1690 004616 104422 SELDRV ;SELECT DRIVE FOR TESTING
1691 004620 104421 REWIND ;GO TO "BOT"
1692 004622 104413 017012 WRITE ,WLNK1 ;WRITE A BLOCK OF DATA
1693 004626 104412 WFG ;WRITE A FILE GAP
1694 004630 104413 017016 WRITE ,WLNK2 ;WRITE 3 BLOCKS OF DATA
1695 004634 104413 017022 WRITE ,WLNK3
1696 004640 104413 017026 WRITE ,WLNK4
1697 004644 104421 REWIND ;GO TO "BOT"
1698 004646 104417 SFBG ;SPACE OVER THE 1ST FILE
1699 004650 104420 SFBG ;SPACE OVER TWO BLOCK
1700 004652 104420 SFBG
1701 004654 104414 016772 READ ,RLNK4 ;READ LAST BLOCK OF DATA
1702 004660 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1703 004662 016772 RLNK4 ;LINK TO THE READ BUFFER
1704 004664 017026 WLNK4 ;LINK TO THE WRITE BUFFER
1705
1706 *****
1707 *TEST 27 TEST BSFG AFTER SFBG AND BSFG AFTER BSFG *****
1708 004666 000004 †ST27: SCOPE
1709 004670 012737 004740 001172 MOV #TST30,$ESCAPE ; ; ESCAPE TO TEST 30 ON ERROR
1710 004676 104422 SELDRV ;SELECT DRIVE FOR TESTING
1711 004700 104421 REWIND ;GO TO "BOT"
1712 004702 104413 017012 WRITE ,WLNK1 ;WRITE ONE BLOCK OF DATA
1713 004706 104412 WFG ;WRITE A FILE GAP
1714 004710 104413 017016 WRITE ,WLNK2 ;WRITE A BLOCK OF DATA
1715 004714 104421 REWIND ;GO TO "BOT"
1716 004716 104417 SFBG ;SPACE OVER 1ST FILE
1717 004720 104420 SFBG ;SPACE OVER 1ST BLOCK OF 2ND FILE
1718 004722 104415 BSFG ;BACK TO BEGINNING OF 2ND FILE
1719 004724 104415 BSFG ;BACK TO BEGINNING OF 1ST FILE
1720 004726 104414 016756 READ ,RLNK1 ;READ 1ST BLOCK OF 1ST FILE
1721 004732 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1722 004734 016756 RLNK1 ;LINK TO THE READ BUFFER
1723 004736 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1724
1725 *****
1726 *TEST 30 TEST BSBG THRU A FILE GAP *****
1727 004740 000004 †ST30: SCOPE
1728 004742 012737 005016 001172 MOV #TST31,$ESCAPE ; ; ESCAPE TO TEST 31 ON ERROR
1729 004750 104422 SELDRV ;SELECT DRIVE FOR TESTING
1730 004752 104421 REWIND ;GO TO "BOT"
1731 004754 104413 017012 WRITE ,WLNK1 ;WRITE ONE BLOCK OF DATA
1732 004760 104412 WFG ;WRITE A FILE GAP

```

```

1733 004762 104413 017016 WRITE ,WLNK2 ;WRITE TWO BLOCKS OF DATA
1734 004766 104413 017022 WRITE ,WLNK3
1735 004772 104421 REWIND ;GO TO "BOT"
1736 004774 104417 SFFG ;SPACE OVER 1ST FILE
1737 004776 104420 SFBG ;SPACE OVER 1ST BLOCK OF 2ND FILE
1738 005000 104416 BSBG ;BACK OVER 1 BLOCK
1739 005002 104416 BSBG ;BACK OVER 1 BLOCK
1740 005004 104414 016756 READ ,RLNK1 ;READ
1741 005010 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1742 005012 016756 RLNK1 ;LINK TO THE READ BUFFER
1743 005014 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1744 *****
1745 ;*TEST 31 TEST BSFG AFTER BSBG
1746 *****
1747 005016 000004 †ST31: SCOPE
1748 005020 012737 005070 001172 MOV #TST32,$ESCAPE ;;ESCAPE TO TEST 32 ON ERROR
1749 005026 104422 SELDRV ;SELECT DRIVE FOR TESTING
1750 005030 104421 REWIND ;GO TO "BOT"
1751 005032 104413 017012 WRITE ,WLNK1 ;WRITE ONE BLOCK OF DATA
1752 005036 104412 WFG ;WRITE A FILE GAP
1753 005040 104413 017016 WRITE ,WLNK2 ;WRITE A BLOCK OF DATA
1754 005044 104421 REWIND ;GO TO "BOT"
1755 005046 104417 SFFG ;SPACE OVER FILE
1756 005050 104420 SFBG ;SPACE OVER 1ST BLOCK OF 2ND FILE
1757 005052 104416 BSBG ;BACK OVER BLOCK
1758 005054 104415 BSBG ;BACK OVER FILE
1759 005056 104414 016756 READ ,RLNK1 ;READ 1ST BLOCK ON TAPE
1760 005062 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1761 005064 016756 RLNK1 ;LINK TO THE READ BUFFER
1762 005066 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1763 *****
1764 ;*TEST 32 TEST BSBG AFTER BSFG
1765 *****
1766 005070 000004 †ST32: SCOPE
1767 005072 012737 005134 001172 MOV #TST33,$ESCAPE ;;ESCAPE TO TEST 33 ON ERROR
1768 005100 104422 SELDRV ;SELECT DRIVE FOR TESTING
1769 005102 104421 REWIND ;GO TO "CLEAR LEADER"
1770 005104 104413 017012 WRITE ,WLNK1 ;WRITE BLOCK
1771 005110 104412 WFG ;WRITE A FILE GAP
1772 005112 104413 017016 WRITE ,WLNK2 ;FOLLOW WITH ANOTHER BLOCK
1773 005116 104415 BSFG ;BACK UP 1 FILE
1774 005120 104416 BSBG ;BACK UP 1 BLOCK
1775 005122 104414 016756 READ ,RLNK1 ;READ 1ST BLOCK ON TAPE
1776 005126 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1777 005130 016756 RLNK1 ;LINK TO THE READ BUFFER
1778 005 3 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1779 *****
1780 ;*TEST 33 TEST BSFG AFTER SFFG AND SFFG AFTER READ
1781 *****
1782 005134 000004 †ST33: SCOPE
1783 005136 012737 005216 001172 MOV #TST34,$ESCAPE ;;ESCAPE TO TEST 34 ON ERROR
1784 005144 104422 SELDRV ;SELECT DRIVE FOR TESTING
1785 005146 104421 REWIND ;GO TO "BOT"
1786 005150 104413 017012 WRITE ,WLNK1 ;WRITE A BLOCK OF DATA
1787 005154 104412 WFG ;WRITE A FILE GAP
1788 005156 104413 017016 WRITE ,WLNK2 ;WRITE A BLOCK OF DATA

```



```

1789 005162 104421 REWIND ;GO TO "BOT"
1790 005164 104417 SFFG ;SPACE OVER 1ST FILE
1791 005166 104415 BSFG ;BACK TO BEGINNING OF 1ST FILE
1792 005170 104414 016756 READ ,RLNK1 ;READ FIRST DATA BLOCK
1793 005174 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1794 005176 016756 RLNK1 ;LINK TO THE READ BUFFER
1795 005200 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1796 005202 104417 SFFG ;SPACE TO BEGINNING ON 2ND FILE
1797 005204 104414 016762 READ ,RLNK2 ;READ 1ST BLOCK OF 2ND FILE
1798 005210 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1799 005212 016762 RLNK2 ;LINK TO THE READ BUFFER
1800 005214 017016 WLNK2 ;LINK TO THE WRITE BUFFER
1801 *****
1802 ;*TEST 34 TEST SFFG AFTER SFFG
1803 *****
1804 005216 000004 ST34: SCOPE
1805 005220 012737 005274 001172 MOV #TST35,$ESCAPE ;;ESCAPE TO TEST 35 ON ERROR
1806 005226 104422 SELDRV ;SELECT DRIVE FOR TESTING
1807 005230 104421 REWIND ;GO TO BOT
1808 005232 104413 017012 WRITE ,WLNK1 ;WRITE A BLOCK
1809 005236 104412 WFG ;WRITE FILE GAP
1810 005240 104413 017016 WRITE ,WLNK2 ;WRITE A BLOCK
1811 005244 104412 WFG ;WRITE A FILE GAP
1812 005246 104413 017022 WRITE ,WLNK3 ;WRITE A BLOCK
1813 005252 104412 WFG ;WRITE A FILE GAP
1814 005254 104421 REWIND ;GO TO "BOT"
1815 005256 104417 SFFG ;SPACE OVER 1ST FILE
1816 005260 104417 SFFG ;SPACE OVER 2ND FILE
1817 005262 104414 016766 READ ,RLNK3 ;READ BLOCK OF 3RD FILE
1818 005266 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1819 005270 016766 RLNK3 ;LINK TO THE READ BUFFER
1820 005272 017022 WLNK3 ;LINK TO THE WRITE BUFFER
1821 *****
1822 ;*TEST 35 TEST SFFG AFTER SFBG
1823 *****
1824 005274 000004 ST35: SCOPE
1825 005276 012737 005350 001172 MOV #TST36,$ESCAPE ;;ESCAPE TO TEST 36 ON ERROR
1826 005304 104422 SELDRV ;SELECT DRIVE FOR TESTING
1827 005306 104421 REWIND ;GO TO "BOT"
1828 005310 104413 017012 WRITE ,WLNK1 ;WRITE A BLOCK
1829 005314 104413 017016 WRITE ,WLNK2 ;WRITE A BLOCK
1830 005320 104412 WFG ;WRITE A GAP
1831 005322 104413 017022 WRITE ,WLNK3 ;WRITE A BLOCK
1832 005326 104412 WFG ;WRITE A GAP
1833 005330 104421 REWIND ;GO TO "BOT"
1834 005332 104420 SFBG ;SPACE OVER THE 1ST BLOCK(AND 1ST FILE)
1835 005334 104417 SFFG ;SPACE TO BEGINNING OF 2ND FILE
1836 005336 104414 016766 READ ,RLNK3 ;READ 1ST BLOCK OF 2ND FILE
1837 005342 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1838 005344 016766 RLNK3 ;LINK TO THE READ BUFFER
1839 005346 017022 WLNK3 ;LINK TO THE WRITE BUFFER
1840 *****
1841 ;*TEST 36 TEST WFG AFTER SFBG
1842 *****
1843 005350 000004 ST36: SCOPE
1844 005352 012737 005422 001172 MOV #TST37,$ESCAPE ;;ESCAPE TO TEST 37 ON ERROR

```

```

1845 005360 104422 SELDRV ;SELECT DRIVE FOR TESTING
1846 005362 104421 REWIND
1847 005364 104413 017012 WRITE ,WLNK1 ;WRITE 2 BLOCKS
1848 005370 104413 017016 WRITE ,WLNK2
1849 005374 104421 REWIND ;GO TO "BOT"
1850 005376 104420 SFBG ;GET OVER THE 1ST BLOCK
1851 005400 104412 WFG ;WRITE A GAP OVER 2ND BLOCK
1852 005402 104413 017022 WRITE ,WLNK3 ;WRITE A BLOCK
1853 005406 104415 BSFG ;BACK OVER THE BLOCK
1854 005410 104414 016766 READ ,RLNK3 ;AND READ IT
1855 005414 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1856 005416 016766 RLNK3 ;LINK TO THE READ BUFFER
1857 005420 017022 WLNK3 ;LINK TO THE WRITE BUFFER
1858 *****
1859 ;*TEST 37 TEST WRITE AFTER SFBG
1860 *****
1861 005422 000004 †ST37: SCOPE
1862 005424 012737 005474 001172 MOV @TST40,$ESCAPE ;;ESCAPE TO TEST 40 ON ERROR
1863 005432 104422 SELDRV ;SELECT DRIVE FOR TESTING
1864 005434 104421 REWIND ;GO TO BOT
1865 005436 104413 017012 WRITE ,WLNK1 ;WRITE 2 BLOCK
1866 005442 104413 017016 WRITE ,WLNK2
1867 005446 104421 REWIND ;GO TO BOT
1868 005450 104420 SFBG ;SPACE OVER 1ST BLOCK
1869 005452 104413 017022 WRITE ,WLNK3 ;WRITE OVER 2ND BLOCK
1870 005456 104415 BSFG ;GO TO BEGINNING OF FILE
1871 005460 104420 SFBG ;SPACE OVER 1ST BLOCK
1872 005462 104414 016766 READ ,RLNK3 ;READ THE NEW 2ND BLOCK
1873 005466 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1874 005470 016766 RLNK3 ;LINK TO THE READ BUFFER
1875 005472 017022 WLNK3 ;LINK TO THE WRITE BUFFER
1876 *****
1877 ;*TEST 40 3 ONE BLOCK FILES---MULTI SPACING & TEST SFFG AFTER BSFG
1878 *****
1879 005474 000004 †ST40: SCOPE
1880 005476 012737 005564 001172 MOV @TST41,$ESCAPE ;;ESCAPE TO TEST 41 ON ERROR
1881 005504 104422 SELDRV ;SELECT DRIVE FOR TESTING
1882 005506 104421 REWIND ;GO TO "BOT"
1883 005510 104413 017012 WRITE ,WLNK1 ;WRITE A BLOCK
1884 005514 104412 WFG ;FILE GAP
1885 005516 104413 017016 WRITE ,WLNK2 ;BLOCK
1886 005522 104412 WFG ;FILE GAP
1887 005524 104413 017022 WRITE ,WLNK3 ;BLOCK
1888 005530 104412 WFG ;FILE GAP
1889 005532 104421 REWIND ;GO TO "BOT"
1890 005534 104417 SFFG ;SPACE OVER 1ST FILE
1891 005536 104417 SFFG ;SPACE OVER 2ND FILE
1892 005540 104417 SFFG ;SPACE OVER 3RD FILE
1893 005542 104415 BSFG ;BACK OVER 3RD FILE
1894 005544 104415 BSFG ;BACK OVER 2ND FILE
1895 005546 104415 BSFG ;BACK OVER 1ST FILE
1896 005550 104417 SFFG ;GO TO BEGINNING OF 2ND FILE
1897 005552 104414 016762 READ ,RLNK2 ;READ THE BLOCK
1898 005556 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1899 005560 016762 RLNK2 ;LINK TO THE READ BUFFER
1900 005562 017016 WLNK2 ;LINK TO THE WRITE BUFFER

```



```

1901          ;*****
1902          ;*TEST 41      TWO BLOCK FILES---MULTI SPACING FUNCTIONS
1903          ;*****
1904 005564 000004          †ST41: SCOPE
1905 005566 012737 005672 001172  MOV      #TST42,$ESCAPE ;;ESCAPE TO TEST 42 ON ERROR
1906 005574 104422          SELDRV   ;SELECT DRIVE FOR TESTING
1907 005576 104421          REWIND   ;GO TO "BOT"
1908 005600 104413 017012  WRITE    ,WLNK1 ;WRITE A 2 BLOCK FILE
1909 005604 104413 017016  WRITE    ,WLNK2
1910 005610 104412          WFG
1911 005612 104413 017022  WRITE    ,WLNK3 ;WRITE A 2 BLOCK FILE
1912 005616 104413 017026  WRITE    ,WLNK4
1913 005622 104412          WFG
1914 005624 104413 017032  WRITE    ,WLNK5 ;WRITE A 2 BLOCK FILE
1915 005630 104413 017036  WRITE    ,WLNK6
1916 005634 104412          WFG
1917 005636 104421          REWIND   ;GO TO "BOT"
1918 005640 104417          SFFG    ;SPACE OVER 1ST FILE
1919 005642 104417          SFFG    ;SPACE OVER 2ND FILE
1920 005644 104414 016776  READ     ,RLNK5 ;READ 1ST BLOCK OF 3RD FILE
1921 005650 104423          BLKCMP  ;COMPARE THE READ AND WRITE BUFFERS
1922 005652 016776          RLNK5   ;LINK TO THE READ BUFFER
1923 005654 017032          WLNK5   ;LINK TO THE WRITE BUFFER
1924 005656 104415          BSFG    ;BACK TO BEGINNING OF 3RD FILE
1925 005660 104414 016776  READ     ,RLNK5 ;READ 1ST BLOCK OF 3RD FILE
1926 005664 104423          BLKCMP  ;COMPARE THE READ AND WRITE BUFFERS
1927 005666 016776          RLNK5   ;LINK TO THE READ BUFFER
1928 005670 017032          WLNK5   ;LINK TO THE WRITE BUFFER
1929          ;*****
1930          ;*TEST 42      MULTI SPACING & TEST SFBG AFTER READ
1931          ;*****
1932 005672 000004          †ST42: SCOPE
1933 005674 012737 006042 001172  MOV      #TST43,$ESCAPE ;;ESCAPE TO TEST 43 ON ERROR
1934 005702 104422          SELDRV   ;SELECT DRIVE FOR TESTING
1935 005704 104421          REWIND   ;GO TO "BOT"
1936 005706 104413 017012  WRITE    ,WLNK1 ;WRITE A 3 BLOCK FILE
1937 005712 104413 017016  WRITE    ,WLNK2
1938 005716 104413 017022  WRITE    ,WLNK3
1939 005722 104412          WFG
1940 005724 104413 017026  WRITE    ,WLNK4 ;WRITE A 2 BLOCK FILE
1941 005730 104413 017032  WRITE    ,WLNK5
1942 005734 104412          WFG
1943 005736 104421          REWIND   ;GO TO "BOT"
1944 005740 104417          SFFG    ;SPACE OVER THE 1ST FILE
1945 005742 104417          SFFG    ;SPACE OVER THE 2ND FILE
1946 005744 104415          BSFG    ;BACK TO BEGINNING OF 2 FILE
1947 005746 104414 016772  READ     ,RLNK4 ;READ 1ST BLOCK OF 2ND FILE
1948 005752 104423          BLKCMP  ;COMPARE THE READ AND WRITE BUFFERS
1949 005754 016772          RLNK4   ;LINK TO THE READ BUFFER
1950 005756 017026          WLNK4   ;LINK TO THE WRITE BUFFER
1951 005760 104415          BSFG    ;BACK TO BEGINNING OF 2ND FILE
1952 005762 104415          BSFG    ;BACK OVER 1ST FILE
1953 005764 104414 016756  READ     ,RLNK1 ;READ 1ND BLOCK OF 1ST FILE
1954 005770 104423          BLKCMP  ;COMPARE THE READ AND WRITE BUFFERS
1955 005772 016756          RLNK1   ;LINK TO THE READ BUFFER
1956 005774 017012          WLNK1   ;LINK TO THE WRITE BUFFER

```

| | | | | | | | |
|------|--------|--------|--------|--------|--------|--|-------------------------------------|
| 1957 | 005776 | 104415 | | BSFG | | | :GO TO BEGINNING OF 1ST FILE |
| 1958 | 006000 | 104414 | 016756 | READ | ,RLNK1 | | :READ 1ST BLOCK |
| 1959 | 006004 | 104423 | | BLKCMP | | | :COMPARE THE READ AND WRITE BUFFERS |
| 1960 | 006006 | 016756 | | RLNK1 | | | :LINK TO THE READ BUFFER |
| 1961 | 006010 | 017012 | | WLNK1 | | | :LINK TO THE WRITE BUFFER |
| 1962 | 006012 | 104420 | | SFBG | | | :GET OVER 2ND BLOCK |
| 1963 | 006014 | 104414 | 016766 | READ | ,RLNK3 | | :READ 3RD BLOCK |
| 1964 | 006020 | 104423 | | BLKCMP | | | :COMPARE THE READ AND WRITE BUFFERS |
| 1965 | 006022 | 016766 | | RLNK3 | | | :LINK TO THE READ BUFFER |
| 1966 | 006024 | 017022 | | WLNK3 | | | :LINK TO THE WRITE BUFFER |
| 1967 | 006026 | 104417 | | SFFG | | | :GO TO BEGINNING OF 2ND FILE |
| 1968 | 006030 | 104414 | 016772 | READ | ,RLNK4 | | :READ 1ST BLOCK OF 2ND FILE |
| 1969 | 006034 | 104423 | | BLKCMP | | | :COMPARE THE READ AND WRITE BUFFERS |
| 1970 | 006036 | 016772 | | RLNK4 | | | :LINK TO THE READ BUFFER |
| 1971 | 006040 | 017026 | | WLNK4 | | | :LINK TO THE WRITE BUFFER |

 :TEST 43 TWO BLOCK FILES--MULTI SPACING & TEST SFBG AFTER BSBG

| | | | | | | | |
|------|--------|--------|---------------|--------------|-----------------|--------------------------------|-------------------------------------|
| 1972 | | | | | | | |
| 1973 | | | | | | | |
| 1974 | | | | | | | |
| 1975 | 006042 | 000004 | | ↑ST43: SCOPE | | | |
| 1976 | 006044 | 012737 | 006162 001172 | MOV | #TST44,\$ESCAPE | ::ESCAPE TO TEST 44 ON ERROR | |
| 1977 | 006052 | 104422 | | SELDRV | | :SELECT DRIVE FOR TESTING | |
| 1978 | 006054 | 104421 | | REWIND | | :GO TO BOT | |
| 1979 | 006056 | 104413 | 017012 | WRITE | ,WLNK1 | | :WRITE A 2 BLOCK FILE |
| 1980 | 006062 | 104413 | 017016 | WRITE | ,WLNK2 | | |
| 1981 | 006066 | 104412 | | WFG | | | |
| 1982 | 006070 | 104413 | 017022 | WRITE | ,WLNK3 | | :WRITE A 2 BLOCK FILE |
| 1983 | 006074 | 104413 | 017026 | WRITE | ,WLNK4 | | |
| 1984 | 006100 | 104412 | | WFG | | | |
| 1985 | 006102 | 104413 | 017032 | WRITE | ,WLNK5 | | :WRITE A 2 BLOCK FILE |
| 1986 | 006106 | 104413 | 017036 | WRITE | ,WLNK6 | | |
| 1987 | 006112 | 104412 | | WFG | | | |
| 1988 | 006114 | 104421 | | REWIND | | :BOT | |
| 1989 | 006116 | 104417 | | SFFG | | :SPACE OVER 3 FILES | |
| 1990 | 006120 | 104417 | | SFFG | | | |
| 1991 | 006122 | 104417 | | SFFG | | | |
| 1992 | 006124 | 104415 | | BSFG | | :BACK OVER 2 FILES | |
| 1993 | 006126 | 104415 | | BSFG | | | |
| 1994 | 006130 | 104416 | | BSBG | | :BACK TO 2ND BLOCK OF 1ST FILE | |
| 1995 | 006132 | 104417 | | SFFG | | :GO TO BEGINNING OF 2ND FILE | |
| 1996 | 006134 | 104414 | 016766 | READ | ,RLNK3 | | :READ 1ST RECORD OF 2ND FILE |
| 1997 | 006140 | 104423 | | BLKCMP | | | :COMPARE THE READ AND WRITE BUFFERS |
| 1998 | 006142 | 016766 | | RLNK3 | | | :LINK TO THE READ BUFFER |
| 1999 | 006144 | 017022 | | WLNK3 | | | :LINK TO THE WRITE BUFFER |
| 2000 | 006146 | 104417 | | SFFG | | :GO TO 3RD FILE | |
| 2001 | 006150 | 104414 | 016776 | READ | ,RLNK5 | | :READ 1ST RECORD OF 3RD FILE |
| 2002 | 006154 | 104423 | | BLKCMP | | | :COMPARE THE READ AND WRITE BUFFERS |
| 2003 | 006156 | 016776 | | RLNK5 | | | :LINK TO THE READ BUFFER |
| 2004 | 006160 | 017032 | | WLNK5 | | | :LINK TO THE WRITE BUFFER |

 :TEST 44 6 ONE BLOCK FILES---MULTI SPACING

| | | | | | | | |
|------|--------|--------|---------------|--------------|----------------|---------------------------|------------------------|
| 2005 | | | | | | | |
| 2006 | | | | | | | |
| 2007 | | | | | | | |
| 2008 | 006162 | 000004 | | ↑ST44: SCOPE | | | |
| 2009 | 006164 | 012737 | 006352 001172 | MOV | #SEOP,\$ESCAPE | ::ESCAPE TO SEOP ON ERROR | |
| 2010 | 006172 | 104422 | | SELDRV | | :SELECT DRIVE FOR TESTING | |
| 2011 | 006174 | 104421 | | REWIND | | :GO TO BOT | |
| 2012 | 006176 | 104413 | 017012 | WRITE | ,WLNK1 | | :WRITE A 1 RECORD FILE |

| | | | | | | |
|------|--------|--------|--------|--------|--------|-------------------------------------|
| 2013 | 006202 | 104412 | | WFG | | |
| 2014 | 006204 | 104413 | 017016 | WRITE | ,WLNK2 | ;WRITE A 1 BLOCK FILE |
| 2015 | 006210 | 104412 | | WFG | | |
| 2016 | 006212 | 104413 | 017022 | WRITE | ,WLNK3 | ;WRITE A 1 BLOCK FILE |
| 2017 | 006216 | 104412 | | WFG | | |
| 2018 | 006220 | 104413 | 017026 | WRITE | ,WLNK4 | ;WRITE A 1 BLOCK FILE |
| 2019 | 006224 | 104412 | | WFG | | |
| 2020 | 006226 | 104413 | 017032 | WRITE | ,WLNK5 | ;WRITE A 1 BLOCK FILE |
| 2021 | 006232 | 104412 | | WFG | | |
| 2022 | 006234 | 104413 | 017036 | WRITE | ,WLNK6 | ;WRITE A 1 BLOCK FILE |
| 2023 | 006240 | 104412 | | WFG | | |
| 2024 | 006242 | 104421 | | REWIND | | ;BOT |
| 2025 | 006244 | 104417 | | SFFG | | ;SPACE OVER 1ST FILE |
| 2026 | 006246 | 104417 | | SFFG | | ;SPACE OVER 2ND FILE |
| 2027 | 006250 | 104417 | | SFFG | | ;SPACE OVER 3RD FILE |
| 2028 | 006252 | 104417 | | SFFG | | ;SPACE OVER 4TH FILE |
| 2029 | 006254 | 104417 | | SFFG | | ;SPACE OVER 5TH FILE |
| 2030 | 006256 | 104417 | | SFFG | | ;SPACE OVER 6TH FILE |
| 2031 | 006260 | 104415 | | BSFG | | ;NOW BACK UP AND READ IT |
| 2032 | 006262 | 104414 | 017002 | READ | ,RLNK6 | |
| 2033 | 006266 | 104423 | | BLKCMP | | ;COMPARE THE READ AND WRITE BUFFERS |
| 2034 | 006270 | 017002 | | RLNK6 | | ;LINK TO THE READ BUFFER |
| 2035 | 006272 | 017036 | | WLNK6 | | ;LINK TO THE WRITE BUFFER |
| 2036 | 006274 | 104415 | | BSFG | | ;BACK UP TO 5TH FILE |
| 2037 | 006276 | 104415 | | BSFG | | |
| 2038 | 006300 | 104414 | 016776 | READ | ,RLNK5 | ;AND READ IT |
| 2039 | 006304 | 104423 | | BLKCMP | | ;COMPARE THE READ AND WRITE BUFFERS |
| 2040 | 006306 | 016776 | | RLNK5 | | ;LINK TO THE READ BUFFER |
| 2041 | 006310 | 017032 | | WLNK5 | | ;LINK TO THE WRITE BUFFER |
| 2042 | 006312 | 104415 | | BSFG | | ;BACK UP TO 3RD FILE |
| 2043 | 006314 | 104415 | | BSFG | | |
| 2044 | 006316 | 104415 | | BSFG | | |
| 2045 | 006320 | 104414 | 016766 | READ | ,RLNK3 | ;AND READ IT |
| 2046 | 006324 | 104423 | | BLKCMP | | ;COMPARE THE READ AND WRITE BUFFERS |
| 2047 | 006326 | 016766 | | RLNK3 | | ;LINK TO THE READ BUFFER |
| 2048 | 006330 | 017022 | | WLNK3 | | ;LINK TO THE WRITE BUFFER |
| 2049 | 006332 | 104415 | | BSFG | | ;BACK UP TO 1ST FILE |
| 2050 | 006334 | 104415 | | BSFG | | |
| 2051 | 006336 | 104415 | | BSFG | | |
| 2052 | 006340 | 104414 | 016756 | READ | ,RLNK1 | ;AND READ IT |
| 2053 | 006344 | 104423 | | BLKCMP | | ;COMPARE THE READ AND WRITE BUFFERS |
| 2054 | 006346 | 016756 | | RLNK1 | | ;LINK TO THE READ BUFFER |
| 2055 | 006350 | 017012 | | WLNK1 | | ;LINK TO THE WRITE BUFFER |

```

2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066 006352
2067 006352 000004
2068 006354 005037 001102
2069 006360 005037 001170
2070 006364 005237 001100
2071 006370 042737 100000 001100
2072 006376 005327
2073 006400 000001
2074 006402 003017
2075 006404 012737
2076 006406 000001
2077 006410 006400
2078 006412 104401 006451
2079 006416 104401 006446
2080 006422 013700 000042
2081 006426 001405
2082 006430 000005
2083 006432 004710
2084 006434 000240
2085 006436 000240
2086 006440 000240
2087 006442
2088 006442 000137
2089 006444 002312
2090 006446 377 377 000
2091 006451 015 042412 042116
2092 006456 050040 051501 000123

```

```

.SBTTL END OF PASS ROUTINE
:*****
: *INCREMENT THE PASS NUMBER ($PASS)
: *TYPE "END PASS"
: *IF THERES A MONITOR GO TO IT
: *IF THERE ISN'T JUMP TO START
: *IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
: *SENDMG CAN BE CHANGED TO 7.
SEOP:
SCOPE
CLR $STNM ;; ZERO THE TEST NUMBER
CLR $TIMES ;; ZERO THE NUMBER OF ITERATIONS
INC $PASS ;; INCREMENT THE PASS NUMBER
BIC #10000,$PASS ;; DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;; LOOP?
SEOPCT: .WORD 1
BGT $DOAGN ;; YES
MOV (PC)+,2(PC)+ ;; RESTORE COUNTER
SENDCT: .WORD 1
SEOPCT
TYPE $SENDMG ;; TYPE "END PASS"
TYPE $ENULL ;; TYPE A NULL CHARACTER
SGET42: MOV #42,R0 ;; GET MONITOR ADDRESS
BEQ $DOAGN ;; BRANCH IF NO MONITOR
RESET ;; CLEAR THE WORLD
SENDAD: JSR PC,(R0) ;; GO TO MONITOR
NOP ;; SAVE ROOM
NOP ;; FOR
NOP ;; ACT11
$DOAGN: JMP 2(PC)+ ;; RETURN
SRTNPD: .WORD START
$ENULL: .BYTE -1,-1,0 ;; NULL CHARACTER STRING
$SENDMG: .ASCIZ <15><12>/END PASS/

```



```

2093 ;*****
2094 ;
2095 .SBTTL ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2096
2097 ;CALL:
2098 MOV #DRVKEY,RO
2099 JSR PC,2#EXAM ;R1 IS DESTROYED
2100 NORMAL RETURN
2101 ERROR RETURN
2102
2103 EXAM: MOV 2#TACSL,R1 ;PICKUP THE "CONTROL & STATUS" REG. ADR.
2104 CLR (R1) ;DRIVE="A" FUNCTION="WFG"
2105 CMPB #'A,(R0) ;EXAMINE DRIVE "A"?
2106 BEQ 1$ ;BR IF YES
2107 BIS #UNIT,(R1) ;SELECT DRIVE "B"
2108 1$: BIT #READY,(R1) ;WAIT ON READY
2109 BEQ 1$
2110 TST (R1) ;ANY ERROR?
2111 BPL 4$ ;BR IF NO
2112 BIT #OFFLINE,(R1) ;ERROR DUE TO "OFF LINE"?
2113 BNE 3$ ;BR IF YES
2114 BIT #WRTLOCK,(R1) ;ERROR DUE TO "WRITE LOCK"?
2115 BEQ 2$ ;BR IF NO
2116 CMPB #BIT07!BIT00,2SWR ;"READONLY" SELECTED? (RDIPAS)
2117 BEQ 4$ ;BR IF YES
2118 CMPB #BIT07!BIT01!BIT00,2SWR ;(RD2PAS)?
2119 BEQ 4$ ;BR IF YES
2120 BR 3$ ;TAKE THE ERROR EXIT
2121 2$: BIT #LEADER,(R1) ;ERROR DUE TO "CLEAR LEADER"?
2122 BNE 4$ ;BR IF YES
2123 3$: ADD #2,(SP) ;TAKE ERROR RETURN
2124 4$: RTS PC ;RETURN

```

```

2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138 006570
2139 006570 104406
2140 006572 105777 172346
2141 006576 100005
2142 006600 104407
2143 006602 122627 000003
2144 006606 001001
2145 006610 104424
2146 006612 005727
2147 006614 000000
2148 006616 001105
2149 006620 032777 040000 172312
2150 006626 001101
2151
2152 006630 000416
2153
2154 006632 013746 000004
2155 006636 012737 006656 000004
2156 006644 005737 177060
2157 006650 012637 000004
2158 006654 000453
2159 006656 022626
2160 006660 012637 000004
2161 006664 000413
2162 006666
2163 006666 105737 001103
2164 006672 001421
2165 006674 123737 001115 001103
2166 006702 101015
2167 006704 032777 001000 172226
2168 006712 001404
2169 006714 013737 001110 001106
2170 006722 000443
2171 006724 105037 001103
2172 006730 005037 001170
2173 006734 000415
2174 006736 032777 004000 172174
2175 006744 001011
2176 006746 005737 001100
2177 006752 001406
2178 006754 005237 001104
2179 006760 023737 001170 001104
2180 006766 002021

```

.SBTTL SCOPE HANDLER ROUTINE

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1      LOOP ON TEST
*SW11=1      INHIBIT ITERATIONS
*SW09=1      LOOP ON ERROR
*CALL
*          SCOPE          ;;SCOPE=IOT

$SCOPE:
          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
          TSTB          $STKS          ;;TTY FLAG UP?
          BPL          100$          ;;BR IF NO
          RDCHR          ;;GO GET ONE CHARACTER FROM TTY
          CMPB          (SP)+, #3     ;;IS IT A CONTROL/C ?
          BNE          100$          ;;BR IF NO
          SETLOOP          ;;GO GET "TEST" AND "SCOPE LOOP" ADDRESSES
          TST          (PC)+          ;;LOOP ON TEST?
          LOOPKEY:      0
          BNE          $OVER          ;;BR IF YES
          BIT          #BIT14, $SWR    ;;LOOP ON PRESENT TEST?
          BNE          $OVER          ;;YES IF SW14=1
          *****START OF CODE FOR THE XOR TESTER*****
          $XTSTR:      BR          6$
          MOV          $ERRVEC, -(SP)  ;;IF RUNNING ON THE "XOR" TESTER CHANGE
          MOV          $$, $ERRVEC    ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
          TST          $#177060      ;;SAVE THE CONTENTS OF THE ERROR VECTOR
          MOV          (SP)+, $ERRVEC  ;;SET FOR TIMEOUT
          BR          $$VLAD          ;;TIME OUT ON XOR?
          CMP          (SP)+, (SP)+   ;;RESTORE THE ERROR VECTOR
          MOV          (SP)+, $ERRVEC  ;;GO TO THE NEXT TEST
          BR          7$              ;;CLEAR THE STACK AFTER A TIME OUT
          *****END OF CODE FOR THE XOR TESTER*****
          TSTB          $ERFLG        ;;HAS AN ERROR OCCURRED?
          BEQ          3$              ;;BR IF NO
          CMPB          $ERMAX, $ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
          BHI          3$              ;;BR IF NO
          BIT          #BIT09, $SWR    ;;LOOP ON ERROR?
          BEQ          4$              ;;BR IF NO
          MOV          $LPERR, $LPADR  ;;SET LOOP ADDRESS TO LAST SCOPE
          BR          $OVER
          CLRB          $ERFLG        ;;ZERO THE ERROR FLAG
          CLR          $TIMES         ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
          BR          1$              ;;ESCAPE TO THE NEXT TEST
          BIT          #BIT11, $SWR    ;;INHIBIT ITERATIONS?
          BNE          1$              ;;BR IF YES
          TST          $PASS          ;;IF FIRST PASS OF PROGRAM
          BEQ          1$              ;;INHIBIT ITERATIONS
          INC          $ICNT          ;;INCREMENT ITERATION COUNT
          CMP          $TIMES, $ICNT   ;;CHECK THE NUMBER OF ITERATIONS MADE
          BGE          $OVER          ;;BR IF MORE ITERATION REQUIRED

```


| | | | | | | | | |
|------|--------|--------|--------|--------|---------|------|------------------|---|
| 2181 | 006770 | 012737 | 000001 | 001104 | IS: | MOV | #1, SICNT | :: REINITIALIZE THE ITERATION COUNTER |
| 2182 | 006776 | 013737 | 001216 | 001170 | | MOV | \$MXCNT, STIMES | :: SET NUMBER OF ITERATIONS TO DO |
| 2183 | 007004 | 105237 | 001102 | | SSVLAD: | INCB | STSTNM | :: COUNT TEST NUMBERS |
| 2184 | 007010 | 011637 | 001106 | | | MOV | (SP), \$LPADR | :: SAVE SCOPE LOOP ADDRESS |
| 2185 | 007014 | 011637 | 001110 | | | MOV | (SP), \$LPERR | :: SAVE ERROR LOOP ADDRESS |
| 2186 | 007020 | 005037 | 001172 | | | CLR | \$ESCAPE | :: CLEAR THE ESCAPE FROM ERROR ADDRESS |
| 2187 | 007024 | 112737 | 000001 | 001115 | | MOVB | #1, \$ERMAX | :: ONLY ALLOW ONE(1) ERROR ON NEXT TEST |
| 2188 | 007032 | 013777 | 001102 | 172102 | \$OVER: | MOV | STSTNM, @DISPLAY | :: DISPLAY TEST NUMBER |
| 2189 | 007040 | 013716 | 001106 | | | MOV | \$LPADR, (SP) | :: FUDGE RETURN ADDRESS |
| 2190 | 007044 | 000002 | | | | RTI | | :: FIXES PS |

2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246

007046
007046 104406
007050 011437 001162
007054 011537 001164
007060 105714
007062 100003
007064 112714 000020
007070 000406
007072 032714 000040
007076 001003
007100 000005
007102 013714 001162
007106 022626
007110 005737 001246
007114 002002
007116 011637 001172
007122 016616 177774
007126 105237 001103
007132 001775
007134 013777 001102 172000
007142 032777 002000 171770
007150 001402
007152 104401 001174
007156 005237 001112
007162 011637 001116
007166 162737 000002 001116
007174 117737 171716 001114
007202 032777 020000 171730
007210 001004
007212 004737 007276
007216 104401 001201
007222
007222 005777 171712
007226 100002
007230 000000
007232 104406
007234 032777 001000 171676
007242 001402
007244 013716 001110
007250 005737 001172
007254 001402
007256 013716 001172

.SBTTL ERROR HANDLER ROUTINE

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

SERROR:
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
MOV @TACS,@#SREG0 ;;SAVE THE TA11 STATUS
MOV @TADB,@#SREG1 ;;SAVE THE DATA BUFFER
TSTB @TACS ;;IF "XFER REQ" IS UP KNOCK IT DOWN
BPL 100\$
MOVB #ILBS,@TACS
BR 101\$
100\$: BIT #READY,@TACS ;;IS READY SET?
BNE 101\$;;BR IF YES
RESET ;;REGAIN CONTROL
MOV @#SREG0,@TACS ;;RESTORE THE TACS
101\$: CMP (SP)+,(SP)+ ;;POP THE STACK
TST ASKKEY ;;MANUAL LOOPING?
BGE 102\$;;BR IF NO
MOV (SP),\$ESCAPE ;;SETUP FOR MANUAL ESCAPE
102\$: MOV -4(SP),(SP) ;;GET PC+2 OF ERROR
7\$: INCB \$ERFLG ;;SET THE ERROR FLAG
BEQ 7\$;;DON'T LET THE FLAG GO TO ZERO
MOV \$TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,@SWR ;;BELL ON ERROR?
BEQ 1\$;;NO - SKIP
TYPE \$BELL ;;RING BELL
1\$: INC \$ERTTL ;;COUNT THE NUMBER OF ERRORS
MOV (SP),\$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,\$ERRPC
MOVB @SWRPC,\$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
BNE 20\$;;SKIP TYPEOUTS
JSR PC,TYPERR ;;GO TO USER ERROR ROUTINE
TYPE \$CRLF
20\$: TST @SWR
2\$: BPL 3\$;;HALT ON ERROR
HALT ;;SKIP IF CONTINUE
CKSWR ;;HALT ON ERROR!
3\$: BIT #BIT09,@SWR ;;TEST FOR CHANGE IN SOFT-SWR
BEQ 4\$;;LOOP ON ERROR SWITCH SET?
MOV \$LPERR,(SP) ;;BR IF NO
4\$: TST \$ESCAPE ;;FUDGE RETURN FOR LOOPING
BEQ 5\$;;CHECK FOR AN ESCAPE ADDRESS
MOV \$ESCAPE,(SP) ;;BR IF NONE
;;FUDGE RETURN ADDRESS FOR ESCAPE

K04

TA11 MOTION TEST C2TAD-D MACY11 30A(1052) 23-MAR-78 09:36 PAGE 49
CZTADD.P11 23-MAR-78 08:34 ERROR HANDLER ROUTINE

SEQ 0049

SEQ 0049

| | | | | | | | | |
|------|--------|--------|--------|-----|------|---------------|-----------------------|--|
| 2247 | 007262 | | | 5S: | | | | |
| 2248 | 007262 | 022737 | 006432 | | CMP | #SENDAD, 2#42 | ::ACT-11 AUTO-ACCEPT? | |
| 2249 | 007270 | 001001 | | | BNE | 6S | ::BRANCH IF NO | |
| 2250 | 007272 | 000000 | | | HALT | | ::YES | |
| 2251 | 007274 | | | 6S: | | | | |
| 2252 | 007274 | 000002 | | | RTI | | ::RETURN | |

```

2253      ;:*****
2254      ; THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
2255      TYPERR: TYPE      ,SCLF      ; TYPE A CARRIAGE RETURN & LINE FEED
2256      TYPE      ; TYPE THE FUNCTION BEING PERFORMED
2257      MFUNC: 0      ; MESSAGE POINTER GOES HERE
2258      TYPE      $SCLF
2259      SUB      @2,@$SAVPC      ; ADJUST THE MAIN FLOW PC
2260      MOV      RO,-(SP)      ; SAVE RO
2261      MOVB     @,$ITEMB,RO      ; PICKUP THE ITEM INDEX
2262      DEC      RO      ; ADJUST THE INDEX
2263      ASL      RO      ; SO IT WILL WORK FOR
2264      ASL      RO      ; THE ERROR TABLE
2265      ASL      RO
2266      ADD      @$ERRTB,RO      ; FORM THE TABLE POINTER
2267      MOV      (RO)+,1$      ; PICKUP "ERROR MESSAGE" POINTER
2268      TYPE      ; TYPE "ERROR MESSAGE"
2269      1$: 0      ; "ERROR MESSAGE" POINTER GOES HERE
2270      MOV      (RO)+,2$      ; PICKUP "DATA HEADER" POINTER
2271      BEQ      3$      ; IF "0" DON'T TYPE
2272      TYPE      ; TYPE "DATA HEADER"
2273      2$: 0      ; "DATA HEADER" POINTER GOES HERE
2274      3$: MOV      (RO)+,RO      ; PICKUP "DATA POINTER"
2275      BNE      5$      ; IF THERE IS DATA TO TYPE GO DO IT
2276      4$: MOV      (SP)+,RO      ; RESTORE RO
2277      TYPE      ,SCLF      ; TYPE A CARRIAGE RETURN&LINE FEED
2278      RTS      PC      ; RETURN
2279      5$:
2280      MOV      @2(RO)+,-(SP)      ;: SAVE @2(RO)+ FOR TYPEOUT
2281      ;: TYPE DATA
2282      TYPCC      ;: GO TYPE--OCTAL ASCII(ALL DIGITS)
2283      TST      (RO)      ; TERMINATOR?
2284      BEQ      4$      ; BR IF YES
2285      TYPE      ,6$      ; TYPE 2 SPACES
2286      BR      5$      ; LOOP
2287      6$: .ASCIZ  / /      ; ASCII STRING OF 2 SPACES
2288      .EVEN

```


ERROR TYPEOUT ROUTINE

```

2289 ;:*****
2290 .SBTTL ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2291
2292 ;CALL
2293 ;:
2294 ;: JSR PC,@#ASKDRV
2295 ;: RETURN ;NOTE: R0 AND R1 ARE DESTROYED
2296
2297 ASKDRV: TYPE MSGDRV ;<CRLF>"DRIVE(S)? "
2298 CLR DRVKEY ;GO GET A DRIVE
2299 RDLIN ;SETUP TO CHECK FOR VALID DRIVE(S)
2300 MOV (SP)+,R0 ;WAS A DRIVE SELECTED?
2301 TSTB @R0 ;BR IF NO
2302 BEQ NOTLGL ;WAS DRIVE "A" SELECTED?
2303 MOV @DRVKEY,R1 ;BR IF NO
2304 CMPB #'A,@R0 ;SET KEY FOR DRIVE "A"
2305 BNE NOTA ;WAS DRIVE "B" SELECTED?
2306 MOVB (R0)+,(R1)+ ;BR IF NO
2307 BR NEXT ;SET KEY FOR DRIVE "B"
2308 NOTA: CMPB #'B,@R0 ;WAS A COMMA TYPED?
2309 BNE NOTB ;BR IF NO
2310 MOVB (R0)+,(R1)+ ;DUMP THE COMMA
2311 BR NEXT ;TERMINATOR?
2312 NOTB: CMPB #54,@R0 ;BR IF YES
2313 BNE NOTLGL ;TWO DRIVES SELECTED?
2314 TSTB (R0)+ ;BR IF NO
2315 TSTB @R0 ;ILLEGAL INPUT DETECTED
2316 BEQ EXIT ;GO TRY AGAIN
2317 CMP #DRVKEY+2,R1 ;ANY DRIVE SELECTED?
2318 BHI LOOP ;BR IF NO
2319 NOTLGL: TYPE $QUES ;
2320 BR ASKDRV ;
2321 EXIT: TST DRVKEY ;
2322 BEQ NOTLGL ;
2323 RTS PC ;
2324 ;:*****
2325 ;CALL
2326 ;: JSR PC,@#ASKADR
2327
2328 ASKADR: MOV R0,-(SP) ;SAVE R0
2329 1$: TYPE ,MSGASK ;"TACS?"
2330 RDOCT ;GET VALUE
2331 MOV (SP)+,R0 ;PICK UP THE OCTAL NUMBER
2332 BEQ 3$ ;IF "0" USE OLD VALUES
2333 CMP R0,#160000 ;MAKE SURE IT IS A BUS ADDRESS
2334 BLO 1$
2335 MOV R0,@#TACSL ;SAVE TOE TACS
2336 ADD #2,R0 ;STEP TO TADB ADDRESS
2337 MOV R0,@#TADBL ;AND SAVE IT
2338 MOV @#TACSL,@#TACSH ;SET UP TACS UPPER
2339 INC @#TACSH ;BYTE POINTER
2340 2$: MOV @#TADBL,@#TADBH ;SET UP TADB UPPER
2341 INC @#TADBH ;BYTE POINTER
2342 3$: TYPE ,MSGVEC ;"VECTOR?"
2343
2344

```

```

2345 007620 104411 RDOCT
2346 007622 012600 MOV (SP)+,RO
2347 007624 001411 BEQ 5$
2348 007626 020027 001000 CMP RO,#1000 ;MAKE SURE ADDRESS IS IN VECTOR AREA
2349 007632 103370 BHIS 3$
2350 007634 010037 001230 MOV RO,#TAVEC ;SAVE AS VECTOR ADDRESS
2351 007640 062700 000002 ADD #2,RO
2352 007644 010037 001232 MOV RO,#TAVEC+2
2353 007650 104401 015172 5$: TYPE ,MSGPRI ;ASK FOR PRIORITY
2354 007654 104411 RDOCT
2355 007656 012600 MOV (SP)+,RO
2356 007660 001411 BEQ 6$ ;IF "0" USE OLD VALUE
2357 007662 020027 000007 CMP RO,#7 ;MAKE SURE ITS VALID
2358 007666 101370 BHI 5$
2359 007670 000300 SWAB RO ;PUT INTO HIGH BYTE
2360 007672 006200 ASR RO ;AND SHIFT
2361 007674 006200 ASR RO ;INTO PROPER
2362 007676 006200 ASR RO ;POSITION
2363 007700 042700 177437 BIC #1C<340>,RO ;SAVE ONLY PRIORITY BITS
2364 007704 010037 001234 MOV RO,#TAPRIO ;STORE IT AWAY
2365 007710 104401 015204 6$: TYPE MTACS ;TACS=""
2366 007714 013746 001220 MOV TACSL,-(SP) ;;SAVE TACSL FOR TYPEOUT
2367 007720 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2368 007722 104401 015213 TYPE MTADB ;"TADB=""
2369 007726 013746 001224 MOV TADBL,-(SP) ;;SAVE TADBL FOR TYPEOUT
2370 007732 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2371 007734 104401 015222 TYPE MTAVEC ;"VECTOR=""
2372 007740 013746 001230 MOV TAVEC,-(SP) ;;SAVE TAVEC FOR TYPEOUT
2373 007744 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2374 007746 104401 015233 TYPE MTAPRI ;"PRIORITY=""
2375 007752 013746 001234 MOV TAPRIO,-(SP) ;;SAVE TAPRIO FOR TYPEOUT
2376 007756 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2377 007760 104401 015246 TYPE ,MSGOK ;"OK?"
2378 007764 104407 RDCHR ;GO READ ONE CHARACTER
2379 007766 012600 MOV (SP)+,RO ;GET IT
2380 007770 022700 000015 CMP #15,RO ;IS IT "CR"?
2381 007774 001406 BEQ 7$ ;BRANCH IF YES
2382 007776 022700 000131 CMP #'Y,RO ;IS IT "Y"?
2383 010002 001403 BEQ 7$ ;IT WAS
2384 010004 104401 001200 TYPE "?" ;TYPE "?"
2385 010010 000651 BR 1$ ;AND LET HIM CORRECT THEM
2386 010012 104401 015254 7$: TYPE MYES ;TYPE OUT "YES"
2387 010016 012600 MOV (SP)+,RO ;RESTORE RO
2388 010020 000207 RTS PC ;AND RETURN

```

: THIS ROUTINE WILL CHANGE AN .ASCII STRING TO AN OCTAL NUMBER

: REGISTER R1 MUST BE INITIALIZED BEFORE ENTERING

: CALL:

JSR R3,#A20CT
POINTER

ERROR RETURN

: CALL THE ROUTINE
: ADDRESS OF THE FIRST CHARACTER IN THE STRING
: STRING MUST BE TERMINATED BY A BYTE OF ALL 0'S
: RETURN HERE IF ILLEGAL CHARACTER

2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400

2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460

010106 104401 015316
010106 104410
010112 104410
010114 012637 010142
010120 105777 000016
010124 001442
010126 022626
010130 010137 010244
010134 005001
010136 004337 010022
010142 000000
010144 000760
010146 005721
010150 010116
010152 104401 015331
010156 104410
010160 012637 010200
010164 105777 000010
010170 001410
010172 005001
010174 004337 010022
010200 000000
010202 000763
010204 010137 001106
010210 000402
010212 011637 001106
010216 011637 001110
010222 012737 177777 006614
010230 000402
010232 005037 006614
010236 013701 010244
010242 000002
010244 000000

; THIS ROUTINE WILL INPUT THE ADDRESSES OF "TEST" AND "SCOPE LOOP" FROM THE TTY

T.SETLOOP:

TYPE ,MTEST
RDLIN
MOV (SP)+,11\$
TST 211\$
BEQ 3\$
CMP (SP)+,(SP)+
MOV R1,SAVR1
CLR R1
JSR R3,2#A2OCT
0
11\$:
BR T.SETLOOP
TST (R1)+
MOV R1,(SP)
5\$:
TYPE ,MLOOP
RDLIN
MOV (SP)+,12\$
TST 212\$
BEQ 1\$
CLR R1
JSR R3,2#A2OCT
0
12\$:
BR 5\$
MOV R1,2#SLPADR
BR 2\$
1\$:
MOV (SP),2#SLPADR
2\$:
MOV (SP),2#SLPERR
MOV #-1,2#LOOPKEY
BR 4\$
3\$:
CLR 2#LOOPKEY
4\$:
MOV SAVR1,R1
RTI
SAVR1: 0

: TYPE "TEST PC?"
: GET AN ASCII STRING
: GET FIRST ADDRESS OF INPUT BUFFER
: FIRST CHAR. A "CR" ?
: BR IF YES---GO CLEAR THE LOOP KEY
: POP POP
: SAVE R1
: SET PARTIAL TO ZERO
: GO CHANGE ASCII TO OCTAL
: POINTER TO STRING
: ERROR RETURN--GO TRY AGAIN
: JUST INCREMENTING
: PUT "TEST PC"; ON THE STACK
: TYPE "LOOP PC?"
: GET A STRING OF ASCII
: GET FIRST ADDRESS OF INPUT BUFFER
: JUST A "CR" ?
: BR IF YES
: CLEAR PARTIAL
: CHANGE TO OCTAL
: SAVE SCOPE LOOP ADDRESS
: GO SET LOOP KEY
: SAVE SCOPE LOOP ADDRESS
: ERROR LOOP SAME AS THE TEST ADDRESS
: SET THE LOOP KEY
: CLEAR THE LOOP KEY
: RESTORE R1
: BYE BYE
: HOLD R1 HERE

2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487

010246
010246 032777 000200 170664
010254 001041
010256 010146
010260 013701 001236
010264 111137 001250
010270 121127 000101
010274 001406
010276 052714 000400
010302 012737 001262 001256
010310 000405
010312 042714 000400 1\$:
010316 012737 001260 001256
010324 005201 2\$:
010326 105711
010330 001002
010332 012701 001240
010336 010137 001236 3\$:
010342 032777 001000 170570
010350 001002
010352 105037 001103
010356 012601 4\$:
010360 000002 5\$:

; THIS ROUTINE WILL SELECT THE NEXT DRIVE TO TEST

T.SELDRV:

BIT #SW07,@SWR
BNE 5\$
MOV R1, -(SP)
MOV DRIVE, R1
MOVD (R1), @CURDRV
CMPB (R1), #'A
BEQ 1\$
BIS #UNIT,@TACS
MOV #RWDB,@RWDFLG
BR 2\$
BIC #UNIT,@TACS 1\$:
MOV #RWDA,@RWDFLG
INC R1 2\$:
TSTB (R1)
BNE 3\$
MOV #DRVKEY, R1
MOV R1, DRIVE 3\$:
BIT #SW09,@SWR
BNE 4\$
CLRB @SERFLG
MOV (SP)+, R1 4\$:
RTI 5\$:

; LOCK ON CURRENT DRIVE?
; BR IF YES
; SAVE R1 ON THE STACK
; GET THE DRIVE
; SAVE CURRENT DRIVE
; IS IT DRIVE "A"?
; BR IF YES
; SELECT DRIVE "B"
; SET REWIND POINTER TO "B"
; SELECT DRIVE "A"
; SET REWIND POINTER TO "A"
; INCREMENT TO THE NEXT DRIVE
; OUT OF DRIVES TO TEST?
; BR IF NO
; RESET TO THE FIRST DRIVE
; SAVE FOR NEXT TIME
; IS SWITCH 9=1?
; BR IF YES
; CLEAR THE ERROR INDICATOR
; RESTORE R1

E05

TA11 MOTION TEST
CZTADD.P11 23-MAR-78

CZTAD-D MACY11 30A(1052)
08:34

23-MAR-78 08:36 PAGE 56
ROUTINE TO COMPARE THE READ AND WRITE BUFFERS

SEQ 0056

SEQ 0056

```

2488
2489
2490 010362 011600
2491 010362 010037 001204
2492 010364 012737 015521 007304
2493 010370 012737 015521 007304
2494 010376 012003
2495 010400 012301
2496 010402 011303
2497 010404 013002
2498 010406 005037 010446
2499 010412 010016
2500 010414 005037 001206
2501 010420 032777 000010 170512
2502 010426 001401
2503 010430 000002
2504 010432 005303 1S:
2505 010434 002001
2506 010436 000002
2507 010440 005237 001206 4S:
2508 010444 122162 000000 2S:
2509 010450 001006
2510 010452 005237 010446
2511 010456 042737 177774 010446
2512 010464 000762
2513 010456 114137 001126 3S:
2514 010472 063702 010446
2515 010476 111237 001124
2516 010502 104010
2517
2518
2519 010504 011637 001204
2520 010510 012737 015502 007304
2521 010516 004037 012224
2522 010522 000101
2523 010524 005037 177776
2524 010530 004037 011716
2525 010534 104003
2526 010536 104004
2527 010540 004037 012014
2528 010544 104002
2529 010546 104002
2530 010550 000002
2531 010552 104002

```

```

*****
THIS ROUTINE WILL COMPARE THE READ AND WRITE BUFFERS
↑.BLKCMP:
MOV (SP),R0 ;GET THE PC OF THE CALL
MOV R0,2(SAVPC) ;SAVE THE PC OF THE CALL
MOV #MBUFCK,2(MFUNC) ;POINTER FOR ASCII MESSAGE
MOV (R0)+,R3 ;PICKUP THE READ LINK ADDRESS
MOV (R3)+,R1 ;PICKUP THE READ BUFFER ADDRESS
MOV (R3),R3 ;PICKUP THE READ BUFFER SIZE
MOV 2(R0)+,R2 ;PICKUP THE WRITE BUFFER ADDRESS
CLR 2S+2 ;ZERO THE OFFSET
MOV R0,(SP) ;SET THE RETURN
CLR 2(BYTNUM) ;ZERO THE BYTE NUMBER
BIT #SW03,2SWR ;COMPARE DATA BUFFERS?
BEQ 1S ;BR IF YES
RTI ;NO---GO BACK
1S: DEC R3 ;CHECK FOR THE TERMINATION
BGE 4S ;BR IF MORE BYTES TO COMPARE
RTI ;RETURN
4S: INC 2(BYTNUM) ;COUNT THIS BYTE
2S: CMPB (R1)+,0(R2) ;COMPARE READ AND WRITE BUFFER
BNE 3S ;BR IF NOT EQUAL
INC 2S+2 ;INCREMENT WRITE BUFFER INDEX
BIC #1C3,2S+2 ;DON'T LET IT GET BIGGER THAN 3
BR 1S
3S: MOVB -(R1),2(SBDDAT) ;GET THE RECEIVED BYTE
ADD 2S+2,R2 ;FORM ADDRESS
MOVB (R2),2(SGDDAT) ;GET THE EXPECTED BYTE
ERROR 10 ;RECEIVED AND EXPECTED DATA NOT EQUAL
*****
THIS ROUTINE WILL WRITE A FILE GAP ON THE TAPE
↑.WFG:
MOV (SP),2(SAVPC) ;SAVE THE PC OF THE CALL
MOV #MXWFG,2(MFUNC) ;SET MESSAGE POINTER FOR "WFG"
JSR R0,2(DO.CMD) ;GO START A "WFG"
XWFG!INT.EN!GO
CLR 2(PS) ;ALLOW INTERRUPTS
JSR R0,2(WAITFLAG) ;GO WAIT FOR A FLAG
ERROR 3 ;NO FLAGS OCCURRED
ERROR 4 ;NO INTERRUPT
JSR R0,2(FLAGS) ;NORMAL RETURN---FIND OUT WHAT FLAG
ERROR 2 ;RETURN HERE IF "XFER REQ" AND "READY" =0
RTI ;RETURN HERE IF "XFER REQ" =1
ERROR 2 ;"RETURN HERE IF "READY"=1 AND "ERROR" =0
RTI ;"READY"=1 AND "ERROR"=1

```



```

2532      ::*****
2533      : THIS ROUTINE WILL WRITE A RECORD ON THE TAPE
2534      †.WRIT: MOV      (SP),@#SAVPC      ;SAVE PC OF CALL
2535      MOV      @#MXWRIT,@#MFUNC      ; POINTER FOR .ASCIZ OF "WRITE"
2536      MOV      @#(SP),R2      ;GET THE WRITE LINK POINTER
2537      ADD      #2,(SP)      ;ADJUST THE RETURN
2538      MOV      (R2)+,R1      ;GET FIRST ADDRESS OF WRITE BUFFER
2539      CLR      @#CRC.WD      ;CLEAR THE CRC WORD
2540      CLR      6$+2      ;ZERO THE INDEX
2541      MOV      (R2)+,R2      ;GET NUMBER OF BYTE TO WRITE
2542      BGT      2$      ;CHECK FOR LEGAL NUMBER
2543      ERROR    11      ;BAD NUMBER OF BYTES
2544      JSR      RO,@#DO.CMD      ;GO START A "WRITE"
2545      XWRITE!INT.EN!GO
2546      CLR      @#PS      ;ALLOW INTERRUPTS
2547      JSR      RO,@#WAITFLAG      ;WAIT FOR A FLAG
2548      ERROR    3      ;NO FLAGS OCCURRED
2549      ERROR    4      ;INTERRUPT FAILED
2550      CLR      @#WAITKEY      ;CLEAR THE WAIT FOR INTERRUPT KEY
2551      MOV      #340,@#PS      ;LOCK OUT THE WORLD
2552      JSR      RO,@#FLAGS      ;WAIT FLAG OCCURRED?
2553      ERROR    2      ;"READY"=0 & "XFER REQ" =0
2554      BR      4$      ;"XFER REQ"=1
2555      ERROR    5      ;"READY"=1 & "ERROR"=0
2556      ERROR    2      ;"READY"=1 & "ERROR"=1
2557      4$: DEC      R2      ;NEED TO WRITE A BYTE ON TAPE
2558      BLT      5$      ;BR IF NO
2559      CLR      RO
2560      6$: BISB   0(R1),RO      ;PICKUP BYTE FOR TRANSFER
2561      MOVB   RO,@#ADB      ;TRANSFER A BYTE TO THE TA11
2562      JSR      PC,@#DO.CRC      ;CALCULATE CRC FOR THIS BYTE
2563      INC      6$+2      ;INCREASE THE INDEX
2564      BIC      #1C3,6$+2      ;KEEP IT LESS THAN 4
2565      MOV      @#SERV,@#AVEC      ;SETUP TO SERVICE THE INTERRUPT
2566      BR      3$      ;LOOP
2567      MOV      @#CRC.WD,4(R1)      ;SAVE THE CRC WORD
2568      JSR      RO,@#DO.CMD      ;START A "ILBS"
2569      XWRITE!INT.EN!ILBS
2570      CLR      @#PS      ;ALLOW INTERRUPTS
2571      JSR      RO,@#WAITFLAG      ;WAIT FOR A FLAG
2572      ERROR    3      ;NO FLAG
2573      ERROR    4      ;NO INTERRUPT
2574      JSR      RO,@#FLAGS      ;WHAT FLAG IS SET?
2575      ERROR    2      ;NONE ("XFER REQ"=0 & "READY"=0)
2576      ERROR    2      ;"XFER REQ"
2577      RTI      ;"READY"
2578      ERROR    2      ;"READY" & "ERROR"
    
```


TA11 MOTION TEST
CZTADD.P11 23-MAR-78

CZTAD-D MACY11 30A(1052) 08:34

23-MAR-78 09:36 PAGE 59
ROUTINE TO "READ" A RECORD

SEQ 0059

SEQ 0059

| | | |
|------|--------|--------|
| 2635 | 011252 | 104002 |
| 2636 | 011254 | 104002 |
| 2637 | 011256 | 000002 |
| 2638 | 011260 | 032714 |
| 2639 | 011264 | 001401 |
| 2640 | 011266 | 104002 |
| 2641 | 011270 | 032777 |
| 2642 | 011276 | 001001 |
| 2643 | 011300 | 104012 |
| 2644 | 011302 | 000002 |
| 2645 | | |

037000

000020 167642 6\$:

7\$:

| | |
|-------|-----|
| ERROR | 2 |
| ERROR | 2 |
| RTI | |
| BIT | |
| BEQ | 6\$ |
| ERROR | 2 |
| BIT | |
| BNE | 7\$ |
| ERROR | 12 |
| RTI | |

#LEADER!WRTLOCK!FGAP!TIMERR!
#SW04,0SWR

```

:UNKNOWN FLAG
: "XFER REQ"
: "READY"
:ERR!OFFLINE,0TACS
:BR IF CRC ERROR
:ERROR OCCURRED
:IGNORE CRC ERRORS?
:BR IF YES
: "READY" AND "CRC ERROR"
:IT WAS A CRC ERROR AND WE ARE IGNORING THEM
:SO RETURN TO CALLER

```

```

2646
2647
2648 011304 011637 001204
2649 011310 012737 015433 007304
2650 011316 004037 012224
2651 011322 000107
2652 011324 005037 177776
2653 011330 004037 011716
2654 011334 104003
2655 011336 104004
2656 011340 004037 012014
2657 011344 104002
2658 011346 104002
2659 011350 000401
2660 011352 104002
2661 011354 032714 004000
2662 011360 001401
2663 011362 000002
2664 011364 104006
2665
2666
2667
2668
2669 011366 011637 001204
2670 011372 012737 015457 007304
2671 011400 004037 012224
2672 011404 000111
2673 011406 005037 177776
2674 011412 004037 011716
2675 011416 104003
2676 011420 104004
2677 011422 004037 012014
2678 011426 104002
2679 011430 104002
2680 011432 000002
2681 011434 104002

:*****
: THIS ROUTINE WILL BACK SPACE ONE FILE GAP
↑.BSFG: MOV (SP),@#SAVPC ;SAVE PC OF CALL
MOV #MXBSFG,@#MFUNC ;SAVE POINTER OF .ASCIZ NAME
1$: JSR RO,@#DO.CMD ;START A "BSFG"
XBSFG!INT.ÉN!GO
CLR @#PS ;ALLOW INTERRUPTS
JSR RO,@#WAITFLAG ;WAIT ON FLAG
ERROR 3 ;NO FLAG
ERROR 4 ;FLAG DIDN'T INTERRUPT
JSR RO,@#FLAGS ;FIND OUT WHAT FLAG OCCURRED
ERROR 2 ;UNKNOWN
BR @#S ;"READY"
ERROR 2 ;"READY" & "ERROR"
2$: BIT #FGAP,@TACS ;IN A FILE GAP?
BEQ @#S ;BR IN NO
RTI ;GO BACK
3$: ERROR 6 ;"BSFG" DIDN'T STOP IN A FILE GAP

:*****
: THIS ROUTINE WILL BACK SPACE ONE BLOCK GAP
↑.BSBG: MOV (SP),@#SAVPC ;SAVE THE PC OF THE CALL
MOV #MXBSBG,@#MFUNC ;MESSAGE POINTER
1$: JSR RO,@#DO.CMD ;START A "BSBG"
XBSBG!INT.ÉN!GO
CLR @#PS ;ALLOW INTERRUPTS
JSR RO,@#WAITFLAG ;WAIT FOR A FLAG
ERROR 3 ;NO FLAG OCCURRED
ERROR 4 ;FLAG DIDN'T INTERRUPT
JSR RO,@#FLAGS ;WHAT FLAG OCCURRED
ERROR 2 ;UNKNOWN
RTI ;"READY"-- GO BACK TO USER
ERROR 2 ;"READY" & "ERROR"

```



```

2682
2683
2684 011436 011637 001204
2685 011442 012737 015366 007304
2686 011450 004037 012224
2687 011454 000113
2688 011456 005037 177776
2689 011462 004037 011716
2690 011466 104003
2691 011470 104004
2692 011472 004037 012014
2693 011476 104002
2694 011500 104002
2695 011502 000401
2696 011504 104002
2697 011506 032714 004000
2698 011512 001401
2699 011514 000002
2700 011516 104006
2701
2702
2703
2704
2705 011520 011637 001204
2706 011524 012737 015411 007304
2707 011532 004037 012224
2708 011536 000115
2709 011540 005037 177776
2710 011544 004037 011716
2711 011550 104003
2712 011552 104004
2713 011554 004037 012014
2714 011560 104002
2715 011562 104002
2716 011564 000002
2717 011566 005737 001246
2718 011572 002001
2719 011574 000002
2720 011576 104002

```

```

*****
: THIS ROUTINE WILL SPACE FORWARD FOR ONE FILE GAP
↑.SFFG: MOV (SP), @#SAVPC ; SAVE PC OF CALL
; MOV @#XSFFG, @#MFUNC ; POINTER TO .ASCIZ MESSAGE
1$: JSR RO, @#DO.CMD ; START A "SFFG"
; XSFFG! INT. EN! GO
; CLR @#PS ; ALLOW INTERRUPTS
; JSR RO, @#WAITFLAG ; WAIT FOR A FLAG
; ERROR 3 ; FLAG FAILED TO OCCUR
; ERROR 4 ; FLAG DIDN'T INTERRUPT
; JSR RO, @#FLAGS ; WHAT FLAG OCCURRED
; ERROR 2 ; UNKNOWN
; ERROR 2 ; "XFER REQ"
; BR 2$ ; "READY"
; ERROR 2 ; "READY" AND "ERROR"
2$: BIT #FGAP, @TACS ; DID "SFFG" STOP IN A "FILE GAP"?
; BEQ 3$ ; BR IF NO
; RTI ; RETURN TO CALLER
3$: ERROR 6 ; "SFFG" DIDN'T STOP IN A FILE GAP

```

```

*****
: THIS ROUTINE WILL SPACE FORWARD ONE BLOCK GAP
↑.SFBG: MOV (SP), @#SAVPC ; SAVE PC OF CALL
; MOV @#XSFBG, @#MFUNC ; POINTER TO .ASCIZ MESSAGE
1$: JSR RO, @#DO.CMD ; START A "SFBG"
; XSFBG! INT. EN! GO
; CLR @#PS ; ALLOW INTERRUPTS
; JSR RO, @#WAITFLAG ; WAIT FOR A FLAG
; ERROR 3 ; NO FLAG
; ERROR 4 ; NO INTERRUPT
; JSR RO, @#FLAGS ; FIND OUT WHAT FLAG
; ERROR 2 ; UNKNOWN
; ERROR 2 ; "XFER REQ"
; RTI ; "READY"
; TST @#ASKKEY ; RUNNING UNDER MANUAL LOOPING?
; BGE 2$ ; BR IF NO
; RTI ; RETURN
2$: ERROR 2 ; "READY" AND "ERROR"

```

```

2721
2722
2723 011600 011637 001204
2724 011604 012737 015344 007304
2725 011612 004037 012014
2726 011616 104001
2727 011620 104001
2728 011622 000411
2729 011624 005777 167426
2730 011630 001406
2731 011632 032714 020000
2732 011636 001403
2733 011640 005077 167412
2734 011644 104001
2735 011646 004037 012224 1$:
2736 011652 000117
2737 011654 005037 177776
2738 011660 004037 011716
2739 011664 104003
2740 011666 104004
2741 011670 004037 012014
2742 011674 104002
2743 011676 104002
2744 011700 000401
2745 011702 104002
2746 011704 032714 020000 2$:
2747 011710 001401
2748 011712 000002
2749 011714 104007 3$:

```

```

*****
↑ THIS ROUTINE WILL REWIND THE TAPE
↑.RWND: MOV (SP),J#SAVPC ;SAVE PC OF CALL
MOV #MXRWND,J#MFUNC ; POINTER TO .ASCIZ MESSAGE
JSR RO,J#FLAGS ;CHECK THE FLAGS
ERROR 1 ;NO FLAGS
ERROR 1 ;"XFER REQ"
BR 1$ ;NORMAL RETURN
TST J#RWDFLG ;WAS LAST FUNCTION A REWIND?
BEQ 1$ ;BR IF YES
BIT #LEADER,J#TACS ;ON CLEAR LEADER?
BEQ 1$ ;BR IF NO
CLR J#RWDFLG ;ALLOW THIS ERROR ONLY ONE TIME
ERROR 1 ;ON CLEAR LEADER BUT SHOULDN'T BE
JSR RO,J#DO.CMD ;GO START A "REWIND"
XRWND:INT.EN!GO
CLR J#PS ;ALLOW INTERRUPTS
JSR RO,J#WAITFLAG ;WAIT ON FLAG
ERROR 3 ;NO FLAG OCCURRED
ERROR 4 ;NO INTERRUPT
JSR RO,J#FLAGS ;WHAT FLAG OCCURRED
ERROR 2 ;UNKNOWN
ERROR 2 ;"XFER REQ"
BR 2$ ;"READY"
ERROR 2 ;"READY" & "ERROR"
BIT #LEADER,J#TACS ;ON "CLEAR LEADER"?
BEQ 3$ ;BR IF NO
RTI ;YES---RETURN
ERROR 7 ;"REWIND" FAILED TO GO TO "CLEAR LEADER"

```


2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779

011716
011716 005037 012006
011722 013737 012012 012010
011730 032777 000040 167202
011736 001403
011740 012737 177777 001254
011746 032714 000240
011752 001007
011754 005237 012006
011760 001372
011762 005337 012010
011766 100367
011770 000405
011772 005720
011774 005737 001254
012000 001401
012002 005720
012004 000200
012006 000000
012010 000000
012012 000000

↑ THIS ROUTINE IS USED TO WAIT ON A FLAG

CALL: JSR RO, @WAITFLAG
RETURN FOR NO FLAGS
RETURN FOR NO INTERRUPT
NORMAL RETURN

WAITFLAG:
CLR LOCNT
MOV MAXCNT, HICNT
BIT #SW05, @SWR
BEQ 1\$
MOV #-1, WAITKEY
BIT @TR.REQ!READY, @TACS
BNE 2\$
INC LOCNT
BNE 1\$
DEC HICNT
BPL 1\$
BR 3\$
2\$: TST (RO)+
TST WAITKEY
BEQ 3\$
TST (RO)+
3\$: RTS
RO
LOCNT: 0
HICNT: 0
MAXCNT: 0

: ZERO THE LOW COUNTER
: SET THE HIGH COUNTER TO MAX.COUNT
: RUNNING WITH INTERRUPTS?
: BR IF YES
: SET THE INTERRUPT OCCURRED FLAG
: TEST FOR FLAGS
: BR IF A FLAG IS UP
: COUNT THE LOW COUNTER
: LOOP IF NOT ZERO
: COUNT THE HIGH COUNTER
: LOOP IF NOT NEG.
: LEAVE
: INCREMENT THE RETURN ADDRESS
: LOOK AT THE WAIT ON INT. KEY
: BR IF NOT SET
: INCREMENT THE RETURN ADDRESS
: RETURN TO USER

2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812

;;*****

: THIS ROUTINE IS USED TO DETERMINE WHAT FLAG IS SET

: CALL:

JSR RO,@#FLAGS
RETURN FOR UNKNOWN FLAG
RETURN FOR "TRANSFER REQUEST"
RETURN FOR "READY"
RETURN FOR "READY" & "ERROR"

: FLAGS:

BIT #TR.REQ!READY,@TACS ;CHECK FOR "XFER REQ" OR "READY"
BEQ LVFLAG ;BR IF NO FLAGS
TST (RO)+ ;INCREMENT RETURN ADDRESS
TSTB @TACS ;CHECK FOR "XFER REQ"
BMI LVFLAG ;BR IF SET
TST (RO)+ ;INCREMENT RETURN ADDRESS
BIT #SW06,@SWR ;STALL?
BEQ 3\$;BR IF NO
MOV RO,-(SP) ;;PUSH RO ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV @#STALL,RO ;GET STALL TIME
MOV @#STALL+2,R1
1\$: ADD #1,RO ;STALL
ADC R1
BNE 1\$;LOOP
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,RO ;;POP STACK INTO RO
3\$: TST @TACS ;IS "ERROR" SET
BPL LVFLAG ;BR IN NO "ERROR"
TST (RO)+ ;INCREMENT RETURN ADDRESS
LVFLAG: RTS RO

000240

001427

005720

105714

100424

005720

032777

001414

010046

010146

013700

013701

062700

005501

001374

012601

012600

167100

001264

001266

000001

1\$:

3\$:

LVFLAG:


```

2813          ;*****
2814          ;THIS ROUTINE WILL CALCULATE THE CRC
2815          ;CALL:
2816          ;      JSR      PC, @DO.CRC          ;RO=1 BYTE OF DATA
2817          ;
2818          DO.CRC:
2819          012102 0107746      MOV      R0, -(SP)          ;: PUSH R0 ON STACK
2820          012104 010146      MOV      R1, -(SP)          ;: PUSH R1 ON STACK
2821          012106 010246      MOV      R2, -(SP)          ;: PUSH R2 ON STACK
2822          012110 010346      MOV      R3, -(SP)          ;: PUSH R3 ON STACK
2823          012112 010446      MOV      R4, -(SP)          ;: PUSH R4 ON STACK
2824          012114 012737 000010 012176      MOV      #8, 3$          ;MAKE EIGHT ITERATIONS
2825          012122 013703 012222      MOV      CRC.WD, R3      ;PICKUP THE CRC WORD
2826          012126 005001      1$:      CLR      R1
2827          012130 010302      MOV      R3, R2          ;GET THE PARTIAL CRC WORD
2828          012132 042702 177776      BIC      #1<BIT00, R2    ;STRIP AWAY EVERYTHING BUT BIT00
2829          012136 006000      ROR      R0          ;PULL OFF THIS BIT
2830          012140 006101      ROL      R1          ;AND SETUP TO XOR IT
2831          012142 010104      MOV      R1, R4          ;FORM THE XOR OF "R1" AND "R2"
2832          012144 040204      BIC      R2, R4
2833          012146 040102      BIC      R1, R2
2834          012150 050402      BIS      R4, R2          ;RESULTS TO "R2"
2835          012152 006002      ROR      R2
2836          012154 103006      BCC      2$
2837          012156 012701 040002      MOV      #BIT14!BIT01, R1
2838          012162 010104      MOV      R1, R4          ;FORM THE XOR OF "R1" AND "R3"
2839          012164 040304      BIC      R3, R4
2840          012166 040103      BIC      R1, R3
2841          012170 050403      BIS      R4, R3          ;RESULTS TO "R3"
2842          012172 006003      2$:      ROR      R3
2843          012174 005327      DEC      (PC)+
2844          012176 000000      3$:      0
2845          012200 003352      BGT      1$          ;BR IF MORE BITS TO DO
2846          012202 010337 012222      MOV      R3, CRC.WD
2847          012206 012604      MOV      (SP)+, R4
2848          012210 012603      MOV      (SP)+, R3      ;: POP STACK INTO R3
2849          012212 012602      MOV      (SP)+, R2      ;: POP STACK INTO R2
2850          012214 012601      MOV      (SP)+, R1      ;: POP STACK INTO R1
2851          012216 012600      MOV      (SP)+, R0      ;: POP STACK INTO R0
2852          012220 000207      RTS
2853          012222 000000      CRC.WD: 0          ;CRC WORD

```

2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897

012224 010146
012226 012737 000340 177776
012234 012001
012236 042701 177640
012242 032777 000040 166670
012250 001405
012252 042701 000100
012256 012777 012356 166744
012264 005037 001254
012270 032701 000100
012274 001403
012276 012777 012340 166724
012304 110114
012306 032701 000001
012312 001403
012314 032714 000040
012320 001375
012322 005101
012324 042701 177761
012330 010177 166722
012334 012601
012336 000200

012340 012737 177777 001254
012346 012777 012356 166654
012354 000002

012356 011637 001204
012362 104014

```
*****  
; THIS ROUTINE WILL LOAD THE LOW BYTE OF TACS WITH THE DESIRED FUNCTION  
; CALL:  
; JSR    RO, @DO, CMD  
; DESIRED FUNCTIN  
; RETURN  
DO. CMD: MOV    R1, -(SP)           ; SAVE R1  
          MOV    #340, @MPS       ; LOCK OUT ALL INERRUPTS  
          MOV    (RO)+, R1        ; GET THE COMMAND  
          BIC    @C<INT.EN!ILBS!FUNCTION!GO>, R1 ; STRIP AWAY JUNK  
          BIT    @SWOS, @SWR      ; WITH OR WITHOUT INTERRUPTS  
          BEQ    #5                ; BR IF WITH  
          BIC    @INT.EN, R1      ; CLEAR INTERRUPT ENABLE IF SET  
          MOV    @BADINT, @TAVEC  ; SETUP TO CATCH BAD INTERRUPT  
          CLR    WAITKEY          ; ZERO THE WAIT ON INTERRUPT KEY  
          BIT    @INT.EN, R1      ; RUNNING WITH INTERRUPTS?  
          BEQ    #3                ; BR IF NO  
          MOV    @SERV, @TAVEC    ; SETUP TO SERVICE THE INTERRUPT  
          MOVB   R1, @TACS        ; LOAD THE COMMAND  
          BIT    @GO, R1          ; IS GO BIT ON A "1"?  
          BEQ    #2                ; BR IF GO =0  
          BIT    @READY, @TACS    ; WAIT FOR "READY" TO CLEAR  
          BNE    #1                ;  
          COM    R1                ; SET OR CLEAR REWIND FLAG  
          BIC    @C<FUNCTION>, R1 ; EXTRACT FUNCTION BITS  
          MOV    R1, @RWDFLG      ; AND SAVE AS THE REWIND FLAG  
          MOV    (SP)+, R1        ; GET R1 BACK  
          RTS    RO                ; RETURN TO USER  
*****  
; THIS ROUTINE WILL SERVICE ALL LEGAL INTERRUPTS  
SERV:    MOV    #-1, @WAITKEY    ; SET THE WAIT KEY  
          MOV    @BADINT, @TAVEC ; SET TO CATCH ILLEGAL INTERRUPTS  
          RTI  
*****  
; THIS ROUTINE WILL CATCH ILLEGAL INTERRUPTS FROM THE TAIL  
BADINT: MOV    (SP), @SAVPC      ; SAVE WHERE WE WERE AT  
          ERROR 14                ; ILLEGAL INTERRUPT OCCURRED
```


2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953

012364 012706 001100
012370 013704 001220
012374 013705 001224
012400 000005
012402 012737 012364 001110
012410 104422
012412 112714 000017
012416 032714 000040
012422 001775
012424 104412
012426 032714 020000
012432 001774
012434 000000
012436 000752

012440 004737 013106
012444 012706 001100
012450 013704 001220
012454 013705 001224
012460 000005
012462 012737 012444 001110
012470 104422
012472 112714 000017
012476 032714 000040

```
*****  
*****  
THE FOLLOWING ROUTINES CAN BE USED TO MAKE ADJUSTMENTS TO THE TU60  
NOTE: *** BEFORE USING ANY OF THE ROUTINES LOAD AND START AT 214 ***  
: ** NOTE: IF USING SOFTWARE SWITCH REGISTER THE  
: PROGRAM MUST HAVE BEEN STARTED AT A  
: NORMAL STARTING ADDRESS FIRST SO THAT  
: THE AUTO-SIZING ROUTINE CAN SET UP ADDRESS  
: 176 AS THE SOFTWARE SWITCH REGISTER  
: PLACE VALUE INTO 176  
*****  
*****
```

```
*****  
WRITE FILE GAPS FROM "BOT" TO "EOT"  
: START AT 220  
: THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND  
: THE "WRITE DELAY MONO".  
*****
```

```
WFGSUB: MOV #STACK, SP ;KEEP THE STACK OUT OF THE WAY  
MOV @TACSL, TACS ;SETUP THE TAI1 STATUS AND  
MOV @TAOBL, TAOB ;DATA BUFFER REGISTERS  
RESET ;RESET THE WORLD  
MOV #WFGSUB, @SLPERR ;SETUP THE LOOP ON ERROR ADDRESS  
SELDRV ;SELECT THE DRIVE  
MOVB #XRWND!GO, @TACS ;SEND TAPE TO "BOT"  
100$: BIT #READY, @TACS ;WAIT ON READY  
BEQ 100$  
1$: WFG ;WRITE A FILE GAP  
BIT #LEADER, @TACS ;AT "CLEAR LEADER"?  
BEQ 1$ ;BR IF NO  
HALT ;STOP IF YES  
BR WFGSUB ;LOOP ON CONT.
```

```
*****  
WRITE CONTINUOUS BLOCKS OF DATA  
: START AT 224  
: THE PROGRAM WILL HALT THREE(3) TIMES  
: AFTER EACH HALT SET THE SWR AND PRESS CONTINUE  
: HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK  
: HALT 2 --- SWR<7:0> = PATTERN DESIRED  
: HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS  
: THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"  
: THIS ROUTINE SUPPORTS THE SW SWITCH REG FUNCTIONS  
*****
```

```
WRTSUB: JSR PC, @SETBUF ;GET BLOCK SIZE AND PATTERN  
WLOOP: MOV #STACK, SP ;KEEP THE STACK OUT OF THE WAY  
MOV @TACSL, TACS ;SETUP THE TAI1 STATUS AND  
MOV @TAOBL, TAOB ;DATA BUFFER REGISTERS  
RESET ;RESET THE WORLD  
MOV #WLOOP, @SLPERR ;SETUP THE LOOP ON ERROR ADDRESS  
SELDRV ;SELECT THE DRIVE  
MOVB #XRWND!GO, @TACS ;SEND TAPE TO "BOT"  
100$: BIT #READY, @TACS ;WAIT ON READY
```

TAII MOTION TEST
CZTADD.P11 23-MAR-78

CZTAD-D MACY11 30A(1052) 23-MAR-78 08:36 PAGE 68
WRITE CONTINUOUS BLOCKS OF DATA

SEQ 0068
SEQ 0068

| | | | |
|------|--------|--------|--------|
| 2954 | 012502 | 001775 | |
| 2955 | 012504 | 104413 | 017042 |
| 2956 | 012510 | 032714 | 020000 |
| 2957 | 012514 | 001773 | |
| 2958 | 012516 | 000000 | |
| 2959 | 012520 | 000751 | |

```

1$: BEQ 1005
WRITE WLNKX ;WRITE A BLOCK
BIT @LEADER,@TACS ;AT "CLEAR LEADER"?
BEQ 1$ ;BR IF NO
HALT ;STOP IF "EOT"
BR WLOOP ;LOOP IF CONT.

```

```

;*****
; READ AND COMPARE CONTINUOUS BLOCKS OF DATA
; START AT 230
; THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
; AND THE "THRESHOLD POT".
;*****

```

| | | | |
|------|--------|--------|---------------|
| 2968 | 012522 | 012706 | 001100 |
| 2969 | 012526 | 013704 | 001220 |
| 2970 | 012532 | 013705 | 001224 |
| 2971 | 012536 | 000005 | |
| 2972 | 012540 | 012737 | 012522 001110 |
| 2973 | 012546 | 104422 | |
| 2974 | 012550 | 112714 | 000017 |
| 2975 | 012554 | 032714 | 000040 |
| 2976 | 012560 | 001775 | |
| 2977 | 012562 | 104414 | 017006 |
| 2978 | 012566 | 012706 | 001100 |
| 2979 | 012572 | 032714 | 020000 |
| 2980 | 012576 | 001351 | |
| 2981 | 012600 | 104423 | |
| 2982 | 012602 | 017006 | |
| 2983 | 012604 | 017042 | |
| 2984 | 012606 | 000765 | |

```

RDSUB: MOV @STACK, SP ;KEEP THE STACK OUT OF THE WAY
MOV @TACSL, TACS ;SETUP THE TAI1 STATUS AND
MOV @TADBL, TADB ;DATA BUFFER REGISTERS
RESET ;RESET THE WORLD
MOV @RDSUB, @SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
SELDIV ;SELECT THE DRIVE
MOV @XRWD!GO, @TACS ;SEND TAPE TO "BOT"
100$: BIT @READY, @TACS ;WAIT ON READY
BEQ 1005
1$: READ RLNKX ;READ A BLOCK
MOV @STACK, SP ;INIT. IN CASE OF ERROR
BIT @LEADER, @TACS ;AT "CLEAR LEADER"?
BNE RDSUB ;BR IF YES---LOOP
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNKX ;LINK TO THE READ BUFFER
WLNKX ;LINK TO THE WRITE BUFFER
BR 1$

```

```

;*****
; WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
; START AT 234
; THE PROGRAM WILL HALT THREE(3) TIMES
; AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
; HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
; HALT 2 --- SWR<7:0> = PATTERN DESIRED
; HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
; THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
; AND THE "GAP TIME MONO".
;*****

```

2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996

2997
2998
2999 012610 004737 013106
3000 012614 012706 001100
3001 012620 013704 001220
3002 012624 013705 001224
3003 012630 000005
3004 012632 012737 012614 001110
3005 012640 104422
3006 012642 112714 000017
3007 012646 032714 000040
3008 012652 001775
3009 012654 104412
3010 012656 032714 020000
3011 012662 001005
3012 012664 104413 017042
3013 012670 032714 020000
3014 012674 001767
3015 012676 000000
3016 012700 000745
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026 012702 012706 001100
3027 012706 013704 001220
3028 012712 013705 001224
3029 012716 000005
3030 012720 012737 012702 001110
3031 012726 104422
3032 012730 112714 000017
3033 012734 032714 000040
3034 012740 001775
3035 012742 104414 017006
3036 012746 012706 001100
3037 012752 032714 020000
3038 012756 001351
3039 012760 104423
3040 012762 017006
3041 012764 017042
3042 012766 104420
3043 012770 032714 020000
3044 012774 001342
3045 012776 000761
3046
3047
3048
3049
3050
3051
3052

```

; THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
; *****
WGPBLK: JSR PC,@SETBUF ; GET BLOCK SIZE AND PATTERN
WGBLOP: MOV @STACK,SP ; KEEP THE STACK OUT OF THE WAY
MOV @TACSL,TACS ; SETUP THE TAIL STATUS AND
MOV @TADBL,TADB ; DATA BUFFER REGISTERS
RESET ; RESET THE WORLD
MOV #WGBLOP,@SLPERR ; SETUP THE LOOP ON ERROR ADDRESS
SELDRV ; SELECT THE DRIVE
MOV# #XRWD!GO,@TACS ; SEND TAPE TO "BOT"
100$: BIT #READY,@TACS ; WAIT ON READY
BEQ 100$
1$: WFG ; WRITE A FILE GAP
BIT #LEADER,@TACS ; AT "CLEAR LEADER"?
BNE 2$ ; BR IF YES
WRITE ,WLNKX ; WRITE A BLOCK
BIT #LEADER,@TACS ; AT "CLEAR LEADER"?
BEQ 1$ ; BR IF NO
2$: HALT ; STOP AT "EOT"
BR WGBLOP ; START OVER ON CONT.

; *****
; READ AND COMPARE A BLOCK OF DATA AND A FILE GAP
; START AT 240
; THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
; IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
; AND THE "TAPE BLANK MONO".
; *****
RGPBLK: MOV @STACK,SP ; KEEP THE STACK OUT OF THE WAY
MOV @TACSL,TACS ; SETUP THE TAIL STATUS AND
MOV @TADBL,TADB ; DATA BUFFER REGISTERS
RESET ; RESET THE WORLD
MOV #RGPBLK,@SLPERR ; SETUP THE LOOP ON ERROR ADDRESS
SELDRV ; SELECT THE DRIVE
MOV# #XRWD!GO,@TACS ; SEND TAPE TO "BOT"
100$: BIT #READY,@TACS ; WAIT ON READY
BEQ 100$
1$: READ ,RLNKX ; READ A BLOCK OF DATA
MOV @STACK,SP ; KEEP OUT OF THE WAY
BIT #LEADER,@TACS ; AT "CLEAR LEADER"?
BNE RGPBLK ; BR IF YES
BLKCMP ; COMPARE THE READ AND WRITE BUFFERS
RLNKX ; LINK TO THE READ BUFFER
WLNKX ; LINK TO THE WRITE BUFFER
SFBG ; GET INTO FILE GAP
BIT #LEADER,@TACS ; AT "CLEAR LEADER"?
BNE RGPBLK ; BR IF YES
BR 1$ ; LOOP

; *****
; SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
; START AT 244
; THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
; SPACE FOWARD (TAPE BLANK MONO CAN BE ADJUSTED), OR AFTER READ OR

```

F06

TAII MOTION TEST
CZTADD.P11 23-MAR-78

CZTAD-D MACY11 30A(1052)
08:34

23-MAR-78 08:36 PAGE 70
SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"

SEQ 0070

SEQ 0070

```

3053                                     ;WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
3054                                     ;(SIGNAL MONO CAN BE CHECKED).
3055                                     ;*****
3056 013000 012706 001100 SFFGSB: MOV #STACK, SP ;KEEP THE STACK OUT OF THE WAY
3057 013004 013704 001220 MOV @TACSL, TACS ;SETUP THE TAI1 STATUS AND
3058 013010 013705 001224 MOV @TADBL, TADB ;DATA BUFFER REGISTERS
3059 013014 000005 RESET ;RESET THE WORLD
3060 013016 012737 013000 001110 MOV #SFFGSB, @SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3061 013024 104422 SELDRV ;SELECT THE DRIVE
3062 013026 112714 000017 MOVB #XRWD!GO, @TACS ;SEND TAPE TO "BOT"
3063 013032 032714 000040 100$: BIT #READY, @TACS ;WAIT ON READY
3064 013036 001775 BEQ 100$
3065 013040 104417 1$: SFFG ;SPACE INTO A FILE GAP
3066 013042 032714 020000 BIT #LEADER, @TACS ;AT "CLEAR LEADER"?
3067 013046 001774 BEQ 1$ ;BR IF NO
3068 013050 000000 HALT ;STOP AT "EOT"
3069 013052 000752 BR SFFGSB ;LOOP ON CONT.
3070
3071
3072                                     ;*****
3073                                     ; BACK SPACE FILE GAP
3074                                     ; START AT 250
3075                                     ; THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
3076                                     ;*****
3077 013054 000005 BSFGSB: RESET ;RESET THE WORLD
3078 013056 012737 013054 001110 MOV #BSFGSB, @SLPERR ;LOOP ON ERROR ADDRESS
3079 013064 005037 001172 CLR #ESCAPE ;DON'T ESCAPE ON ERROR
3080 013070 104422 SELDRV ;SELECT THE DRIVE
3081 013072 104415 1$: BSFG ;BACK SPACE A FILE GAP
3082 013074 032714 020000 BIT #LEADER, @TACS ;AT "CLEAR LEADER"?
3083 013100 001774 BEQ 1$ ;BR IF NO
3084 013102 000000 HALT ;STOP AT BOT
3085 013104 000763 BR BSFGSB ;START OVER ON CONT.
3086
3087
3088                                     ;*****
3089                                     ; SETUP READ AND WRITE LINKS FOR SUBROUTINES
3090 SETBUF: CLR RO
3091 013110 000000 HALT ;OPERATOR PUTS BYTE COUNT IN THE SWR
3092 013112 022737 000176 001140 CMP #SWREG, SWR ;USING S/W SWITCH REG?
3093 013120 001001 BNE 20$ ;NO- GET OUT
3094 013122 104405 GTSWR ;GET VALUE
3095 013124 20$: ;CONTINUE
3096 013124 157700 166010 BISB @SWR, RO ;PICKUP THE BYTE COUNT
3097 013130 001006 BNE 2$ ;BR IF NON-ZERO
3098 013132 105777 166003 TSTB @SWR+1 ;CHECK IF GREATER THAN 377
3099 013136 001402 BEQ 1$ ;BR IF NO
3100 013140 012700 000376 MOV #376, RO ;SET FOR MAX ALLOWED
3101 013144 005200 1$: INC RO ;MAKE IT 377 OR 1
3102 013146 010037 017010 2$: MOV RO, @RLNKX+2 ;SETUP READ LINK
3103 013152 010037 017044 MOV RO, @WLNKX+2 ;SETUP WRITE LINK
3104 013156 000000 HALT ;SET PATTERN INTO THE SWR
3105 013160 022737 000176 001140 CMP #SWREG, SWR ;USING S/W SWITCH REG?
3106 013166 001001 BNE 21$ ;NO- GET OUT
3107 013170 104405 GTSWR ;GET VALUE
3108 013172 21$: ;CONTINUE

```


| | | | |
|------|--------|--------|---------------|
| 3109 | 013172 | 017700 | 165742 |
| 3110 | 013176 | 012701 | 017112 |
| 3111 | 013202 | 110021 | |
| 3112 | 013204 | 110021 | |
| 3113 | 013206 | 110021 | |
| 3114 | 013210 | 110021 | |
| 3115 | 013212 | 000000 | |
| 3116 | 013214 | 022737 | 000176 001140 |
| 3117 | 013222 | 001001 | |
| 3118 | 013224 | 104405 | |
| 3119 | 013226 | | 22\$: |
| 3120 | 013226 | 000207 | |

```

MOV    @SWR, R0
MOV    #WBUF, R1
MOVB  R0, (R1)+
MOVB  R0, (R1)+
MOVB  R0, (R1)+
MOVB  R0, (R1)+
HALT
CMP    #SWREG, SWR
BNE   22$
GTSWR
RTS    PC

```

```

; PICKUP THE PATTERN
; PICKUP FIRST ADDRESS OF BUFFER
; FILL THE BUFFER

; SET OPERATIONAL SWITCHES
; USING S/W SWITCH REG?
; NO- GET OUT
; GET VALUE
; CONTINUE
; RETURN

```

3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176

.SBTTL TYPE ROUTINE

:ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
:THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
:NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
:NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
:NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.

*CALL:
:1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ;:MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR

\$TYPE: TSTB \$TFPLG ;: IS THERE A TERMINAL?
BPL 1\$;: BR IF YES
HALT ;: HALT HERE IF NO TERMINAL
BR ;: LEAVE
1\$: MOV RO, -(SP) ;: SAVE RO
MOV 22(SP), RO ;: GET ADDRESS OF ASCIZ STRING
2\$: MOVVB (RO)+, -(SP) ;: PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4\$;: BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;: IF TERMINATOR POP IT OFF THE STACK
60\$: MOV (SP)+, RO ;: RESTORE RO
3\$: ADD #2, (SP) ;: ADJUST RETURN PC
RTI ;: RETURN
4\$: CMPB #HT, (SP) ;: BRANCH IF <HT>
BEQ 8\$;: BRANCH IF NOT <CRLF>
CMPB #CRLF, (SP) ;: BRANCH IF NOT <CRLF>
BNE 5\$;: POP <CR><LF> EQUIV
TST (SP)+ ;: TYPE A CR AND LF
\$CHARCNT ;: CLEAR CHARACTER COUNT
BR 2\$;: GET NEXT CHARACTER
5\$: JSR PC, \$TYPEC ;: GO TYPE THIS CHARACTER
6\$: CMPB \$FILLC, (SP)+ ;: IS IT TIME FOR FILLER CHARS.?
BNE 2\$;: IF NO GO GET NEXT CHAR.
MOV \$NULL, -(SP) ;: GET # OF FILLER CHARS. NEEDED
AND THE NULL CHAR.
7\$: DECB 1(SP) ;: DOES A NULL NEED TO BE TYPED?
BLT 6\$;: BR IF NO--GO POP THE NULL OFF OF STACK
JSR PC, \$TYPEC ;: GO TYPE A NULL
DECB \$CHARCNT ;: DO NOT COUNT AS A COUNT
BR 7\$;: LOOP

;HORIZONTAL TAB PROCESSOR

8\$: MOVB #' (SP) ;: REPLACE TAB WITH SPACE
9\$: JSR PC, \$TYPEC ;: TYPE A SPACE
BITB #7, \$CHARCNT ;: BRANCH IF NOT AT
BNE 9\$;: TAB STOP
TST (SP)+ ;: POP SPACE OFF STACK

013444


```

3177 013376 000724
3178 013400 105777 165544
3179 013404 100375
3180 013406 116677 000002 165536
3181 013414 122766 000015 000002
3182 013422 001003
3183 013424 105037 013444
3184 013430 000406
3185 013432 122766 000012 000002 1$:
3186 013440 001402
3187 013442 105227
3188 013444 000000 $CHARCNT: .WORD 0
3189 013446 000207 $TYPEX: RTS PC
3190
3191 .SBTTL TTY INPUT ROUTINE
3192
3193 ::*****
3194 .ENABL LSB
3195
3196 ::*****
3197 ::SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3198 ::ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
3199 ::SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
3200 ::WHEN OPERATING IN TTY FLAG MODE.
3201 013450 022737 000176 001140 $CKSWR: CMP #SWREG,SWR :: IS THE SOFT-SWR SELECTED?
3202 013456 001074 BNE 15$ :: BRANCH IF NO
3203 013460 105777 165460 TSTB @STKS :: CHAR THERE?
3204 013464 100071 BPL 15$ :: IF NO, DON'T WAIT AROUND
3205 013466 117746 165454 MOVB @STKB, -(SP) :: SAVE THE CHAR
3206 013472 042716 177600 BIC #1C177, (SP) :: STRIP-OFF THE ASCII
3207 013476 022726 000007 CMP #7, (SP)+ :: IS IT A CONTROL G?
3208 013502 001062 BNE 15$ :: NO, RETURN TO USER
3209 013504 123727 001134 000001 CMPB $AUTOB, #1 :: ARE WE RUNNING IN AUTO-MODE?
3210 013512 001456 BEQ 15$ :: BRANCH IF YES
3211
3212 013514 104401 014175 $GTSWR: TYPE , $CNTLG :: ECHO THE CONTROL-G (↑G)
3213 013520 104401 014202 TYPE , $MSWR :: TYPE CURRENT CONTENTS
3214 013524 013746 000176 MOV $WREG, -(SP) :: SAVE SWREG FOR TYPEOUT
3215 013530 104402 TYPOC :: GO TYPE--OCTAL ASCII(ALL DIGITS)
3216 013532 104401 014213 TYPE , $MNEW :: PROMPT FOR NEW SWR
3217 013536 005046 19$: CLR -(SP) :: CLEAR COUNTER
3218 013540 005046 CLR -(SP) :: THE NEW SWR
3219 013542 105777 165376 7$: TSTB @STKS :: CHAR THERE?
3220 013546 100375 BPL 7$ :: IF NOT TRY AGAIN
3221
3222 013550 117746 165372 MOVB @STKB, -(SP) :: PICK UP CHAR
3223 013554 042716 177600 BIC #1C177, (SP) :: MAKE IT 7-BIT ASCII
3224
3225
3226
3227 013560 021627 000025 9$: CMP (SP), #25 :: IS IT A CONTROL-U?
3228 013564 001005 BNE 10$ :: BRANCH IF NOT
3229 013566 104401 014170 TYPE , $CNTLU :: YES, ECHO CONTROL-U (↑U)
3230 013572 062706 000006 20$: ADD #6, SP :: IGNORE PREVIOUS INPUT
3231 013576 000757 BR 19$ :: LET'S TRY IT AGAIN
3232
    
```

```

3233
3234 013600 021627 000015 10$: CMP (SP),#15      ;; IS IT A <CR>?
3235 013604 001022          BNE 16$          ;; BRANCH IF NO
3236 013606 005766 000004    TST 4(SP)        ;; YES, IS IT THE FIRST CHAR?
3237 013612 001403          BEQ 11$          ;; BRANCH IF YES
3238 013614 016677 000002 165316  MOV 2(SP),@SWR   ;; SAVE NEW SWR
3239 013622 062706 000006    11$: ADD #6,SP     ;; CLEAR UP STACK
3240 013626 104401 001201    14$: TYPE $CRLF  ;; ECHO <CR> AND <LF>
3241 013632 123727 001135 000001  CMPB $INTAG,#1  ;; RE-ENABLE TTY KBD INTERRUPTS?
3242 013640 001003          BNE 15$          ;; BRANCH IF NOT
3243 013642 012777 000100 165274  MOV #100,@STKS  ;; RE-ENABLE TTY KBD INTERRUPTS
3244 013650 000002          RTI             ;; RETURN
3245 013652 004737 013400    15$: JSR PC,$TYPEC  ;; ECHO CHAR
3246 013656 021627 000060    16$: CMP (SP),#60   ;; CHAR < 0?
3247 013662 002420          BLT 18$          ;; BRANCH IF YES
3248 013664 021627 000067    CMP (SP),#67    ;; CHAR > 7?
3249 013670 003015          BGT 18$          ;; BRANCH IF YES
3250 013672 042726 000060    BIC #60,(SP)+  ;; STRIP-OFF ASCII
3251 013676 005766 000002    TST 2(SP)       ;; IS THIS THE FIRST CHAR
3252 013702 001403          BEQ 17$          ;; BRANCH IF YES
3253 013704 006316          ASL (SP)        ;; NO, SHIFT PRESENT
3254 013706 006316          ASL (SP)        ;; CHAR OVER TO MAKE
3255 013710 006316          ASL (SP)        ;; ROOM FOR NEW ONE.
3256 013712 005266 000002    17$: INC 2(SP)     ;; KEEP COUNT OF CHAR
3257 013716 056616 177776    BIS -2(SP),(SP) ;; SET IN NEW CHAR
3258 013722 000707          BR 7$           ;; GET THE NEXT ONE
3259 013724 104401 001200    18$: TYPE $QUES  ;; TYPE ?<CR><LF>
3260 013730 000720          BR 20$          ;; SIMULATE CONTROL-U
3261
3262 .DSABL LSB
3263
3264
3265
3266
3267
3268
3269
3270
3271

```

```

*****
*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

```

```

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

```

```

3272 013732 011646          $RDCHR: MOV (SP),-(SP)  ;; PUSH DOWN THE PC
3273 013734 016666 000004 000002  MOV 4(SP),2(SP) ;; SAVE THE PS
3274 013742 105777 165176    1$: TSTB @STKS    ;; WAIT FOR
3275 013746 100375          BPL 1$          ;; A CHARACTER
3276 013750 117766 165172 000004  MOVB @STKB,4(SP) ;; READ THE TTY
3277 013756 042766 177600 000004  BIC #177,4(SP)  ;; GET RID OF JUNK IF ANY
3278 013764 026627 000004 000023  CMP 4(SP),#23  ;; IS IT A CONTROL-S?
3279 013772 001013          BNE 3$          ;; BRANCH IF NO
3280 013774 105777 165144    2$: TSTB @STKS    ;; WAIT FOR A CHARACTER
3281 014000 100375          BPL 2$          ;; LOOP UNTIL ITS THERE
3282 014002 117746 165140    MOVB @STKB, -(SP) ;; GET CHARACTER
3283 014006 042716 177600    BIC #177,(SP)  ;; MAKE IT 7-BIT ASCII
3284 014012 022627 000021    CMP (SP)+,#21  ;; IS IT A CONTROL-Q?
3285 014016 001366          BNE 2$          ;; IF NOT DISCARD IT
3286 014020 000750          BR 1$          ;; YES, RESUME
3287 014022 026627 000004 000140  3$: CMP 4(SP),#140 ;; IS IT UPPER CASE?
3288 014030 002407          BLT 4$          ;; BRANCH IF YES

```



```

3289 014032 026627 000004 000175      CMP      4(SP),#175      ;; IS IT A SPECIAL CHAR?
3290 014040 003003                BGT      4$              ;; BRANCH IF YES
3291 014042 042766 000040 000004      BIC      #40,4(SP)      ;; MAKE IT UPPER CASE
3292 014050 000002                RTI                    ;; GO BACK TO USER
3293                ;; *****
3294                ;; *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3295                ;; *CALL:
3296                ;; *      RDLIN                ;; INPUT A STRING FROM THE TTY
3297                ;; *      RETURN HERE          ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3298                ;; *                      ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
3299
3300 014052 010346                $RDLIN: MOV      R3,-(SP)      ;; SAVE R3
3301 014054 012703 014160          1$:  MOV      #TTYIN,R3      ;; GET ADDRESS
3302 014060 022703 014170          2$:  CMP      #TTYIN+8.,R3     ;; BUFFER FULL?
3303 014064 101405                BLOS     4$              ;; BR IF YES
3304 014066 104407                RDCHR                    ;; GO READ ONE CHARACTER FROM THE TTY
3305 014070 112613                MOVB    (SP)+(R3)        ;; GET CHARACTER
3306 014072 122713 000177          10$: CMPB    #177,(R3)       ;; IS IT A RUBOUT
3307 014076 001003                BNE     3$              ;; SKIP IF NOT
3308 014100 104401 001200          4$:  TYPE    $QUES        ;; TYPE A '?'
3309 014104 000763                BR      1$              ;; CLEAR THE BUFFER AND LOOP
3310 014106 111337 014156          3$:  MOVB    (R3),9$        ;; ECHO THE CHARACTER
3311 014112 104401 014156          TYPE    9$
3312 014116 122723 000015          CMPB    #15,(R3)+      ;; CHECK FOR RETURN
3313 014122 001356                BNE     2$              ;; LOOP IF NOT RETURN
3314 014124 105063 177777          CLRB   -1(R3)          ;; CLEAR RETURN (THE 15)
3315 014130 104401 001202          TYPE    $LF           ;; TYPE A LINE FEED
3316 014134 012603                MOV     (SP)+,R3        ;; RESTORE R3
3317 014136 011646                MOV     (SP),-(SP)      ;; ADJUST THE STACK AND PUT ADDRESS OF THE
3318 014140 016666 000004 000002          MOV     4(SP),2(SP)    ;; FIRST ASCII CHARACTER ON IT
3319 014146 012766 014160 000004          MOV     #TTYIN,4(SP)
3320 014154 000002                RTI                    ;; RETURN
3321 014156 000                9$:  .BYTE    0              ;; STORAGE FOR ASCII CHAR. TO TYPE
3322 014157 000                .BYTE    0              ;; TERMINATOR
3323 014160 000010                $TTYIN: .BLKB 8          ;; RESERVE 8 BYTES FOR TTY INPUT
3324 014170 052536 005015 000          $CNTLU: .ASCIZ /?U/<15><12> ;; CONTROL "U"
3325 014175 136 006507 000012          $CNTLG: .ASCIZ /?G/<15><12> ;; CONTROL "G"
3326 014202 005015 053523 020122          $MSWR:  .ASCIZ <15><12>/SWR = /
3327 014210 020075 000
3328 014213 040 047040 053505          $MNEW:  .ASCIZ / NEW = /
3329 014220 036440 000040
3330                .SBTTL READ AN OCTAL NUMBER FROM THE TTY
3331
3332                ;; *****
3333                ;; *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
3334                ;; *CHANGE IT TO BINARY.
3335                ;; *CALL:
3336                ;; *      RDOCT                ;; READ AN OCTAL NUMBER
3337                ;; *      RETURN HERE          ;; LOW ORDER BITS ARE ON TOP OF THE STACK
3338                ;; *                      ;; HIGH ORDER BITS ARE IN $HIOCT
3339
3340 014224 011646                $RDOCT: MOV     (SP),-(SP) ;; PROVIDE SPACE FOR THE
3341 014226 016666 000004 000002          MOV     4(SP),2(SP)    ;; INPUT NUMBER
3342 014234 010046                MOV     R0,-(SP)       ;; PUSH R0 ON STACK
3343 014236 010146                MOV     R1,-(SP)       ;; PUSH R1 ON STACK
3344 014240 010246                MOV     R2,-(SP)       ;; PUSH R2 ON STACK

```

| | | | | | | | |
|------|--------|--------|--------|------|------------------|--|---------------------------------|
| 3345 | 014242 | 104410 | | 1\$: | RDLIN | | :::READ AN ASCIZ LINE |
| 3346 | 014244 | 012600 | | | MOV (SP)+,R0 | | :::GET ADDRESS OF 1ST CHARACTER |
| 3347 | 014246 | 005001 | | | CLR R1 | | :::CLEAR DATA WORD |
| 3348 | 014250 | 005002 | | | CLR R2 | | |
| 3349 | 014252 | 112046 | | 2\$: | MOVB (R0)+,-(SP) | | :::PICKUP THIS CHARACTER |
| 3350 | 014254 | 001412 | | | BEQ 3\$ | | :::IF ZERO GET OUT |
| 3351 | 014256 | 006301 | | | ASL R1 | | :::*2 |
| 3352 | 014260 | 006102 | | | ROL R2 | | |
| 3353 | 014262 | 006301 | | | ASL R1 | | :::*4 |
| 3354 | 014264 | 006102 | | | ROL R2 | | |
| 3355 | 014266 | 006301 | | | ASL R1 | | :::*8 |
| 3356 | 014270 | 006102 | | | ROL R2 | | |
| 3357 | 014272 | 042716 | 177770 | | BIC #C7,(SP) | | :::STRIP THE ASCII JUNK |
| 3358 | 014276 | 062601 | | | ADD (SP)+,R1 | | :::ADD IN THIS DIGIT |
| 3359 | 014300 | 000764 | | | BR 2\$ | | :::LOOP |
| 3360 | 014302 | 005726 | | 3\$: | TST (SP)+ | | :::CLEAN TERMINATOR FROM STACK |
| 3361 | 014304 | 010166 | 000012 | | MOV R1,12(SP) | | :::SAVE THE RESULT |
| 3362 | 014310 | 010237 | 014324 | | MOV R2,\$HIOCT | | |
| 3363 | 014314 | 012602 | | | MOV (SP)+,R2 | | :::POP STACK INTO R2 |
| 3364 | 014316 | 012601 | | | MOV (SP)+,R1 | | :::POP STACK INTO R1 |
| 3365 | 014320 | 012600 | | | MOV (SP)+,R0 | | :::POP STACK INTO R0 |
| 3366 | 014322 | 000002 | | | RTI | | :::RETURN |
| 3367 | 014324 | 000000 | | | \$HIOCT: .WORD 0 | | :::HIGH ORDER BITS GO HERE |

3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423

014326 017646 000000
014332 116637 000001
014340 112637 014553
014344 062716 000002
014350 000406
014352 112737 000001
014360 112737 000006
014366 112737 000005
014374 010346
014376 010446
014400 010546
014402 113704 014553
014406 005404
014410 062704 000006
014414 110437 014552
014420 113704 014551
014424 016605 000012
014430 005003
014432 006105
014434 000404
014436 006105
014440 006105
014442 006105
014444 010503
014446 006103
014450 105337 014552
014454 100016
014456 042703 177770
014462 001002
014464 005704
014466 001403

014551
014551
014553
014550
014553
014552
177770

```
.SBTTL BINARY TO OCTAL (ASCII) AND TYPE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOS   N              ;;CALL FOR TYPEOUT
*   .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*STYPOS OR STYPOC
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON   N              ;;CALL FOR TYPEOUT
*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOC   N              ;;CALL FOR TYPEOUT
*STYPOS:  MOV     2(SP),-(SP)  ;;PICKUP THE MODE
*         MOV     1(SP),SOFILL ;;LOAD ZERO FILL SWITCH
*         MOV     (SP)+,SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
*         ADD     #2,(SP)      ;;ADJUST RETURN ADDRESS
*         BR     STYPON
*STYPOC:  MOV     #1,SOFILL    ;;SET THE ZERO FILL SWITCH
*         MOV     #6,SOMODE+1  ;;SET FOR SIX(6) DIGITS
*STYPON:  MOV     #5,SOCNT     ;;SET THE ITERATION COUNT
*         MOV     R3,-(SP)     ;;SAVE R3
*         MOV     R4,-(SP)     ;;SAVE R4
*         MOV     R5,-(SP)     ;;SAVE R5
*         MOV     SOMODE+1,R4  ;;GET THE NUMBER OF DIGITS TO TYPE
*         NEG     R4
*         ADD     #6,R4        ;;SUBTRACT IT FOR MAX. ALLOWED
*         MOV     R4,SOMODE    ;;SAVE IT FOR USE
*         MOV     SOFILL,R4    ;;GET THE ZERO FILL SWITCH
*         MOV     12(SP),R5    ;;PICKUP THE INPUT NUMBER
*         CLR     R3          ;;CLEAR THE OUTPUT WORD
*         ROL     R5          ;;ROTATE MSB INTO "C"
*         BR     3$          ;;GO TO MSB
*         ROL     R5          ;;FORM THIS DIGIT
*         ROL     R5
*         ROL     R5
*         MOV     R5,R3
*         ROL     R3          ;;GET LSB OF THIS DIGIT
*         DECB   SOMODE      ;;TYPE THIS DIGIT?
*         BPL   7$          ;;BR IF NO
*         BIC   #17777C,R3  ;;GET RID OF JUNK
*         BNE   4$          ;;TEST FOR 0
*         TST   R4          ;;SUPPRESS THIS 0?
*         BEQ   5$          ;;BR IF YES
```

| | | | | | | | |
|------|--------|--------|---------------|----------|-------|-------------|------------------------------------|
| 3424 | 014470 | 005204 | | 4\$: | INC | R4 | :: DON'T SUPPRESS ANYMORE D'S |
| 3425 | 014472 | 052703 | 000060 | | BIS | #'0,R3 | :: MAKE THIS DIGIT ASCII |
| 3426 | 014476 | 052703 | 000040 | 5\$: | BIS | #',R3 | :: MAKE ASCII IF NOT ALREADY |
| 3427 | 014502 | 110337 | 014546 | | MOVB | R3,8\$ | :: SAVE FOR TYPING |
| 3428 | 014504 | 104401 | 014546 | | TYPE | 8\$ | :: GO TYPE THIS DIGIT |
| 3429 | 014512 | 105337 | 014550 | 7\$: | DECB | \$OCNT | :: COUNT BY 1 |
| 3430 | 014516 | 003347 | | | BGT | 2\$ | :: BR IF MORE TO DO |
| 3431 | 014520 | 002402 | | | BLT | 6\$ | :: BR IF DONE |
| 3432 | 014522 | 005204 | | | INC | R4 | :: INSURE LAST DIGIT ISN'T A BLANK |
| 3433 | 014524 | 000744 | | | BR | 2\$ | :: GO DO THE LAST DIGIT |
| 3434 | 014526 | 012605 | | 6\$: | MOV | (SP)+,R5 | :: RESTORE R5 |
| 3435 | 014530 | 012604 | | | MOV | (SP)+,R4 | :: RESTORE R4 |
| 3436 | 014532 | 012603 | | | MOV | (SP)+,R3 | :: RESTORE R3 |
| 3437 | 014534 | 016666 | 000002 000004 | | MOV | 2(SP),4(SP) | :: SET THE STACK FOR RETURNING |
| 3438 | 014542 | 012616 | | | MOV | (SP)+,(SP) | |
| 3439 | 014544 | 000002 | | | RTI | | :: RETURN |
| 3440 | 014546 | 000 | | 8\$: | .BYTE | 0 | :: STORAGE FOR ASCII DIGIT |
| 3441 | 014547 | 000 | | | .BYTE | 0 | :: TERMINATOR FOR TYPE ROUTINE |
| 3442 | 014550 | 000 | | \$OCNT: | .BYTE | 0 | :: OCTAL DIGIT COUNTER |
| 3443 | 014551 | 000 | | \$OFILL: | .BYTE | 0 | :: ZERO FILL SWITCH |
| 3444 | 014552 | 000000 | | \$OMODE: | .WORD | 0 | :: NUMBER OF DIGITS TO TYPE |

.SBTTL TRAP DECODER

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

3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460

```
STRAP:  MOV    RO,-(SP)          ;;SAVE RO
        MOV    2(SP),RO         ;;GET TRAP ADDRESS
        TST    -(RO)           ;;BACKUP BY 2
        MOVB   (RO),RO         ;;GET RIGHT BYTE OF TRAP
        ASL    RO              ;;POSITION FOR INDEXING
        MOV    STRPAD(RO),RO    ;;INDEX TO TABLE
        RTS    RO              ;;GO TO ROUTINE
```

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

3461
3462
3463
3464
3465
3466
3467

```
STRAP2: MOV    (SP),-(SP)      ;;MOVE THE PC DOWN
        MOV    4(SP),2(SP)    ;;MOVE THE PSW DOWN
        RTI                    ;;RESTORE THE PSW
```

.SBTTL TRAP TABLE

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

3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497

| ROUTINE | STRAP2 | TRAP | DESCRIPTION |
|---------------|----------------|-----------------|--|
| STRPAD: .WORD | STRAP2 | TRAP+1(104401) | TTY TYPEOUT ROUTINE |
| STTYPE | ;;CALL=TYPE | TRAP+2(104402) | TYPE OCTAL NUMBER (WITH LEADING ZEROS) |
| STYPOC | ;;CALL=TYPOC | TRAP+3(104403) | TYPE OCTAL NUMBER (NO LEADING ZEROS) |
| STYPOS | ;;CALL=TYPOS | TRAP+4(104404) | TYPE OCTAL NUMBER (AS PER LAST CALL) |
| STYPON | ;;CALL=TYPON | | |
| SGTSWR | ;;CALL=GTSWR | TRAP+5(104405) | GET SOFT-SWR SETTING |
| SCKSWR | ;;CALL=CKSWR | TRAP+6(104406) | TEST FOR CHANGE IN SOFT-SWR |
| SRDCHR | ;;CALL=RDCHR | TRAP+7(104407) | TTY TYPEIN CHARACTER ROUTINE |
| SRDLIN | ;;CALL=RDLIN | TRAP+10(104410) | TTY TYPEIN STRING ROUTINE |
| SRDOCT | ;;CALL=RDOCT | TRAP+11(104411) | READ AN OCTAL NUMBER FROM TTY |
| T.WFG | ;;CALL=WFG | TRAP+12(104412) | ROUTINE TO WRITE A FILE GAP |
| T.WRIT | ;;CALL=WRITE | TRAP+13(104413) | ROUTINE TO WRITE A BLOCK OF DATA |
| T.READ | ;;CALL=READ | TRAP+14(104414) | ROUTINE TO READ A BLOCK OF DATA |
| T.BSFG | ;;CALL=BSFG | TRAP+15(104415) | ROUTINE TO BACK SPACE A FILE GAP |
| T.BSBG | ;;CALL=BSBG | TRAP+16(104416) | ROUTINE TO BACK SPACE A BLOCK GAP |
| T.SFFG | ;;CALL=SFFG | TRAP+17(104417) | ROUTINE TO SPACE FWD A FILE GAP |
| T.SFBG | ;;CALL=SFBG | TRAP+20(104420) | ROUTINE TO SPACE FWD A BLOCK GAP |
| T.RWIND | ;;CALL=REWIND | TRAP+21(104421) | ROUTINE TO REWIND THE TAPE |
| T.SELDRV | ;;CALL=SELDV | TRAP+22(104422) | ROUTINE TO SELECT THE NEXT DRIVE |
| T.BLKCMP | ;;CALL=BLKCMP | TRAP+23(104423) | COMPARE READ AND WRITE BUFFERS |
| T.SETLOOP | ;;CALL=SETLOOP | TRAP+24(104424) | SETUP TO LOOP AS PER THE TTY |

.SBTTL POWER DOWN AND UP ROUTINES

```

3498
3499
3500
3501
3502 014662 012737 015026 000024 $PWRDN: MOV $SILLUP, @PWRVEC ;; SET FOR FAST UP
3503 014670 012737 000340 000026 MOV @340, @PWRVEC+2 ;; PRIO:7
3504 014676 010046 MOV RO, -(SP) ;; PUSH RO ON STACK
3505 014700 010146 MOV R1, -(SP) ;; PUSH R1 ON STACK
3506 014702 010246 MOV R2, -(SP) ;; PUSH R2 ON STACK
3507 014704 010346 MOV R3, -(SP) ;; PUSH R3 ON STACK
3508 014706 010446 MOV R4, -(SP) ;; PUSH R4 ON STACK
3509 014710 010546 MOV R5, -(SP) ;; PUSH R5 ON STACK
3510 014712 017746 164222 MOV @SWR, -(SP) ;; PUSH @SWR ON STACK
3511 014716 010637 015032 MOV SP, $SAVR6 ;; SAVE SP
3512 014722 012737 014734 000024 MOV $PWRUP, @PWRVEC ;; SET UP VECTOR
3513 014730 000000 HALT
3514 014732 000776 BR .-2 ;; HANG UP
3515
3516
3517
3518 014734 012737 015026 000024 $PWRUP: MOV $SILLUP, @PWRVEC ;; SET FOR FAST DOWN
3519 014742 013706 015032 MOV $SAVR6, SP ;; GET SP
3520 014746 005037 015032 CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
3521 014752 005237 015032 IS: INC $SAVR6 ;; WAIT FOR THE INC
3522 014756 001375 BNE IS OF WORD
3523 014760 012677 164154 MOV (SP)+, @SWR ;; POP STACK INTO @SWR
3524 014764 012605 MOV (SP)+, R5 ;; POP STACK INTO R5
3525 014766 012604 MOV (SP)+, R4 ;; POP STACK INTO R4
3526 014770 012603 MOV (SP)+, R3 ;; POP STACK INTO R3
3527 014772 012602 MOV (SP)+, R2 ;; POP STACK INTO R2
3528 014774 012601 MOV (SP)+, R1 ;; POP STACK INTO R1
3529 014776 012600 MOV (SP)+, R0 ;; POP STACK INTO R0
3530 015000 012737 014662 000024 MOV $PWRDN, @PWRVEC ;; SET UP THE POWER DOWN VECTOR
3531 015006 012737 000340 000026 MOV @340, @PWRVEC+2 ;; PRIO:7
3532 015014 104401 TYPE REPORT THE POWER FAILURE
3533 015016 015034 SPWRMG: .WORD $POWER ;; POWER FAIL MESSAGE POINTER
3534 015020 012716 MOV (PC)+, (SP) ;; RESTART AT PWRST
3535 015022 002300 SPWRAD: .WORD PWRST ;; RESTART ADDRESS
3536 015024 000002 RTI
3537 015026 000000 $SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
3538 015030 000776 BR .-2 ;; BEFORE THE POWER DOWN WAS COMPLETE
3539 015032 000000 $SAVR6: 0 ;; PUT THE SP HERE
3540 015034 005015 047520 042527 $POWER: .ASCIIZ <15><12>"POWER"
3541 015042 000122 .EVEN
3542

```


3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563

;;*****

;DATA POINTERS

| | | | | | | |
|--------|--------|--------|--------|------|-------|---|
| 015044 | 001204 | 001116 | 001162 | DT1: | .WORD | SAVPC, SERRPC, SREGO, SREG1, 0 |
| 015052 | 001164 | 000000 | | | | |
| 015056 | 001204 | 001116 | 001162 | DT2: | .WORD | SAVPC, SERRPC, SREGO, SGDDAT, SBDDAT, BYTNUM, 0 |
| 015064 | 001124 | 001126 | 001206 | | | |
| 015072 | 000000 | | | | | |
| 015074 | 001204 | 001116 | 001162 | DT3: | .WORD | SAVPC, SERRPC, SREGO, RCRCO, RCRC1, RCRC2, 0 |
| 015102 | 001210 | 001212 | 001214 | | | |
| 015117 | 000000 | | | | | |
| 015112 | 001204 | 001116 | 001162 | DT4: | .WORD | SAVPC, SERRPC, SREGO, SREG1, STMPD, 0 |
| 015120 | 001164 | 001166 | 000000 | | | |
| 015126 | 001204 | 001116 | 001162 | DT5: | .WORD | SAVPC, SERRPC, SREGO, 0 |
| 015134 | 000000 | | | | | |

```

3564 ;*****
3565 ;
3566 ;MESSAGES
3567
3568 015136 042200 044522 042526 MSGDRV: .ASCIZ <CRLF>"DRIVE(S)? "
3569 015144 051450 037451 000040
3570 015152 005015 040524 051503 MSGASK: .ASCIZ <15><12>/TACS?/
3571 015160 000077
3572 015162 042526 052103 051117 MSGVEC: .ASCIZ /VECTOR?/
3573 015170 000077
3574 015172 051120 047511 044522 MSGPRI: .ASCIZ /PRIORITY?/
3575 015200 054524 000077
3576 015204 052040 041501 036523 MTACS: .ASCIZ / TACS=/
3577 015212 000
3578 015213 040 040524 041104 MTADB: .ASCIZ / TADB=/
3579 015220 000075
3580 015222 053040 041505 047524 MTAVEC: .ASCIZ / VECTOR=/
3581 015230 036522 000
3582 015233 040 051120 047511 MTAPRI: .ASCIZ / PRIORITY=/
3583 015240 044522 054524 000075
3584 015246 005015 045517 000077 MSGOK: .ASCIZ <15><12>/OK?/
3585 015254 042531 100123 000 MYES: .ASCIZ /YES/<CRLF>
3586 015261 015 052012 051505 MTSTDRV: .ASCIZ <15><12>/TESTING DRIVE /
3587 015266 044524 043516 042040
3588 015274 044522 042526 000040
3589 015302 040440 042116 042040 MANDRV: .ASCIZ / AND DRIVE /
3590 015310 044522 042526 000040
3591 015316 005015 042524 052123 MTEST: .ASCIZ <15><12>/TEST PC?/
3592 015324 050040 037503 000
3593 015331 015 046012 047517 NLOOP: .ASCIZ <15><12>/LOOP PC?/
3594 015336 020120 041520 000077
3595 015344 042522 044527 042116 MXRWND: .ASCIZ /REWIND/
3596 015352 000
3597 015353 127 044522 042524 MXWRIT: .ASCIZ /WRITE/
3598 015360 000
3599 015361 122 047505 000104 MXREAD: .ASCIZ /READ/
3600 015366 050123 041501 026505 MXSFFG: .ASCIZ /SPACE-FWD-FILE-GAP/
3601 015374 053506 026504 044506
3602 015402 042514 043455 050101
3603 015410 000
3604 015411 123 040520 042503 MXSFBG: .ASCIZ /SPACE-FWD-BLK-GAP/
3605 015416 043055 042127 041055
3606 015424 045514 043455 050101
3607 015432 000
3608 015433 102 041501 026513 MXBSFG: .ASCIZ /BACK-SPACE-FILE-GAP/
3609 015440 050123 041501 026505
3610 015446 044506 042514 043455
3611 015454 050101 000
3612 015457 102 041501 026513 MXBSBG: .ASCIZ /BACK-SPACE-BLK-GAP/
3613 015464 050123 041501 026505
3614 015472 046102 026513 040507
3615 015500 000120
3616 015502 051127 052111 026505 MXWFG: .ASCIZ /WRITE-FILE-GAP/
3617 015510 044506 042514 043455
3618 015516 050101 000
3619 015521 102 043125 042506 MBUFCK: .ASCIZ /BUFFER COMPARE/

```


| | | | | | | | | | |
|------|--------|--------|--------|--------|-------|--------|-------------------------------|--|--|
| 3620 | 015526 | 020122 | 047503 | 050115 | | | | | |
| 3621 | 015534 | 051101 | 000105 | | | | | | |
| 3622 | 015540 | 046511 | 051120 | 050117 | EM1: | .ASCIZ | /IMPROPER FLAG SETTING/ | | |
| 3623 | 015546 | 051105 | 043040 | 040514 | | | | | |
| 3624 | 015554 | 020107 | 042523 | 052124 | | | | | |
| 3625 | 015562 | 047111 | 000107 | | | | | | |
| 3626 | 015566 | 046511 | 051120 | 050117 | EM2: | .ASCIZ | /IMPROPER FLAG OCCURRED/ | | |
| 3627 | 015574 | 051105 | 043040 | 040514 | | | | | |
| 3628 | 015602 | 020107 | 041517 | 052503 | | | | | |
| 3629 | 015610 | 051122 | 042105 | 000 | | | | | |
| 3630 | 015615 | 115 | 051511 | 042523 | EM3: | .ASCIZ | /MISSED A FLAG/ | | |
| 3631 | 015622 | 020104 | 020101 | 046106 | | | | | |
| 3632 | 015630 | 043501 | 000 | | | | | | |
| 3633 | 015633 | 111 | 052116 | 051105 | EM4: | .ASCIZ | /INTERRUPT FAILED/ | | |
| 3634 | 015640 | 052522 | 052120 | 043040 | | | | | |
| 3635 | 015646 | 044501 | 042514 | 000104 | | | | | |
| 3636 | 015654 | 051120 | 046505 | 052101 | EM5: | .ASCIZ | /PREMATURE READY OCCURRED/ | | |
| 3637 | 015662 | 051125 | 020105 | 042522 | | | | | |
| 3638 | 015670 | 042101 | 020131 | 041517 | | | | | |
| 3639 | 015676 | 052503 | 051122 | 042105 | | | | | |
| 3640 | 015704 | 000 | | | | | | | |
| 3641 | 015705 | 104 | 042111 | 023516 | EM6: | .ASCIZ | /DIDN'T STOP IN A FILE GAP/ | | |
| 3642 | 015712 | 020124 | 052123 | 050117 | | | | | |
| 3643 | 015720 | 044440 | 020116 | 020101 | | | | | |
| 3644 | 015726 | 044506 | 042514 | 043440 | | | | | |
| 3645 | 015734 | 050101 | 000 | | | | | | |
| 3646 | 015737 | 104 | 042111 | 023516 | EM7: | .ASCIZ | /DIDN'T STOP ON CLEAR LEADER/ | | |
| 3647 | 015744 | 020124 | 052123 | 050117 | | | | | |
| 3648 | 015752 | 047442 | 020116 | 046103 | | | | | |
| 3649 | 015760 | 040505 | 020122 | 042514 | | | | | |
| 3650 | 015766 | 042101 | 051105 | 000 | | | | | |
| 3651 | 015773 | 102 | 042101 | 042040 | EM10: | .ASCIZ | /BAD DATA READ/ | | |
| 3652 | 016000 | 052101 | 020101 | 042522 | | | | | |
| 3653 | 016006 | 042101 | 000 | | | | | | |
| 3654 | 016011 | 111 | 046114 | 043505 | EM11: | .ASCIZ | /ILLEGAL BUFFER SIZE/ | | |
| 3655 | 016016 | 046101 | 041040 | 043125 | | | | | |
| 3656 | 016024 | 042506 | 020122 | 044523 | | | | | |
| 3657 | 016032 | 042532 | 000 | | | | | | |
| 3658 | 016035 | 103 | 041522 | 042440 | EM12: | .ASCIZ | /CRC ERROR/ | | |
| 3659 | 016042 | 051122 | 051117 | 000 | | | | | |
| 3660 | 016047 | 123 | 047510 | 052122 | EM13: | .ASCIZ | /SHORT RECORD/ | | |
| 3661 | 016054 | 051040 | 041505 | 051117 | | | | | |
| 3662 | 016062 | 000104 | | | | | | | |
| 3663 | 016064 | 040502 | 020104 | 047111 | EM14: | .ASCIZ | /BAD INTERRUPT/ | | |
| 3664 | 016072 | 042524 | 051122 | 050125 | | | | | |
| 3665 | 016100 | 000124 | | | | | | | |
| 3666 | 016102 | 005015 | | | DH1: | .ASCII | <15><12> | | |
| 3667 | 016104 | 042524 | 052123 | 020040 | | .ASCII | /TEST ERROR/<15><12> | | |
| 3668 | 016112 | 020040 | 051105 | 047522 | | | | | |
| 3669 | 016120 | 006522 | 012 | | | | | | |
| 3670 | 016123 | 120 | 020103 | 020040 | | .ASCIZ | /PC PC TACS TADB/<15><12> | | |
| 3671 | 016130 | 020040 | 050040 | 020103 | | | | | |
| 3672 | 016136 | 020040 | 020040 | 052040 | | | | | |
| 3673 | 016144 | 041501 | 020123 | 020040 | | | | | |
| 3674 | 016152 | 052040 | 042101 | 006502 | | | | | |
| 3675 | 016160 | 000012 | | | | | | | |

| | | | | |
|------|--------|--------|--------|--------|
| 3676 | 016162 | 005015 | | |
| 3677 | 016164 | 042524 | 052123 | 020040 |
| 3678 | 016172 | 020040 | 051105 | 047522 |
| 3679 | 016200 | 020122 | 020040 | 020040 |
| 3680 | 016206 | 020040 | 020040 | 020040 |
| 3681 | 016214 | 054105 | 052120 | 042047 |
| 3682 | 016222 | 020040 | 041522 | 023526 |
| 3683 | 016230 | 020104 | 020040 | 054502 |
| 3684 | 016236 | 042524 | 005015 | |
| 3685 | 016242 | 041520 | 020040 | 020040 |
| 3686 | 016250 | 020040 | 041520 | 020040 |
| 3687 | 016256 | 020040 | 020040 | 040524 |
| 3688 | 016264 | 051503 | 020040 | 020040 |
| 3689 | 016272 | 040504 | 040524 | 020040 |
| 3690 | 016300 | 020040 | 040504 | 040524 |
| 3691 | 016306 | 020040 | 020040 | 052516 |
| 3692 | 016314 | 041115 | 051105 | 005015 |
| 3693 | 016322 | 000 | | |
| 3694 | 016323 | 015 | 012 | |
| 3695 | 016325 | 124 | 051505 | 020124 |
| 3696 | 016332 | 020040 | 042440 | 051122 |
| 3697 | 016340 | 051117 | 020040 | 020040 |
| 3698 | 016346 | 020040 | 020040 | 020040 |
| 3699 | 016354 | 053440 | 044522 | 042524 |
| 3700 | 016362 | 020040 | 051040 | 040505 |
| 3701 | 016370 | 020104 | 020040 | 052040 |
| 3702 | 016376 | 033125 | 006460 | 012 |
| 3703 | 016403 | 120 | 020103 | 020040 |
| 3704 | 016410 | 020040 | 050040 | 020103 |
| 3705 | 016416 | 020040 | 020040 | 052040 |
| 3706 | 016424 | 041501 | 020123 | 020040 |
| 3707 | 016432 | 041440 | 041522 | 020040 |
| 3708 | 016440 | 020040 | 041440 | 041522 |
| 3709 | 016446 | 020040 | 020040 | 041440 |
| 3710 | 016454 | 041522 | 005015 | 000 |
| 3711 | 016461 | 015 | 012 | |
| 3712 | 016463 | 124 | 051505 | 020124 |
| 3713 | 016470 | 020040 | 042440 | 051122 |
| 3714 | 016476 | 051117 | 020040 | 020040 |
| 3715 | 016504 | 020040 | 020040 | 020040 |
| 3716 | 016512 | 020040 | 020040 | 020040 |
| 3717 | 016520 | 020040 | 041040 | 052131 |
| 3718 | 016526 | 051505 | 005015 | |
| 3719 | 016532 | 041520 | 020040 | 020040 |
| 3720 | 016540 | 020040 | 041520 | 020040 |
| 3721 | 016546 | 020040 | 020040 | 040524 |
| 3722 | 016554 | 051503 | 020040 | 020040 |
| 3723 | 016562 | 040524 | 041104 | 020040 |
| 3724 | 016570 | 020040 | 042514 | 052106 |
| 3725 | 016576 | 005015 | 000 | |
| 3726 | 016601 | 015 | 012 | |
| 3727 | 016603 | 124 | 051505 | 020124 |
| 3728 | 016610 | 020040 | 042440 | 051122 |
| 3729 | 016616 | 051117 | 005015 | |
| 3730 | 016622 | 041520 | 020040 | 020040 |
| 3731 | 016630 | 020040 | 041520 | 020040 |

| | | | | | | | | | |
|------|--------|----------|----------------|---------------|--------|---------------|-----------------|--|--|
| DH2: | .ASCII | <15><12> | | | | | | | |
| | .ASCII | /TEST | ERROR | | EXPT'D | RCV'D | BYTE/<15><12> | | |
| | .ASCIZ | /PC | PC | TACS | DATA | DATA | NUMBER/<15><12> | | |
| DH3: | .ASCII | <15><12> | | | | | | | |
| | .ASCII | /TEST | ERROR | | WRITE | READ | TU60/<15><12> | | |
| | .ASCIZ | /PC | PC | TACS | CRC | CRC | CRC/<15><12> | | |
| DH4: | .ASCII | <15><12> | | | | | | | |
| | .ASCII | /TEST | ERROR | | | | BYTES/<15><12> | | |
| | .ASCIZ | /PC | PC | TACS | TADB | LEFT/<15><12> | | | |
| DH5: | .ASCII | <15><12> | | | | | | | |
| | .ASCII | /TEST | ERROR/<15><12> | | | | | | |
| | .ASCIZ | /PC | PC | TACS/<15><12> | | | | | |

H07

TA11 MOTION TEST CZTAD-D MACY11 30A(1052) 23-MAR-78 09:36 PAGE 85
CZTADD.P11 23-MAR-78 08:34 ASCII MESSAGES

SEQ 0085

SEQ 0085

| | | | | | | | | | |
|------|--------|--------|--------|--------|--------|--------|--------------------------|--|--|
| 3732 | 016636 | 020040 | 020040 | 040524 | | | | | |
| 3733 | 016644 | 051503 | 005015 | 000 | | | | | |
| 3734 | | 016652 | | | | .EVEN | | | |
| 3735 | 016652 | 001116 | 001220 | 000000 | DT201: | .WORD | SERRPC,TACSL,0 | | |
| 3736 | | | | | | | | | |
| 3737 | 016660 | 001116 | 000000 | | DT202: | .WORD | SERRPC,0 | | |
| 3738 | | | | | | | | | |
| 3739 | 016664 | 040524 | 030461 | 043040 | EM201: | .ASCIZ | "TA11 FAILED TO RESPOND" | | |
| 3740 | 016672 | 044501 | 042514 | 020104 | | | | | |
| 3741 | 016700 | 047524 | 051040 | 051505 | | | | | |
| 3742 | 016706 | 047520 | 042116 | 000 | | | | | |
| 3743 | 016713 | 116 | 020117 | 051104 | EM202: | .ASCIZ | "NO DRIVE AVAILABLE" | | |
| 3744 | 016720 | 053111 | 020105 | 053101 | | | | | |
| 3745 | 016726 | 044501 | 040514 | 046102 | | | | | |
| 3746 | 016734 | 000105 | | | | | | | |
| 3747 | 016736 | 041520 | 020040 | 020040 | DH201: | .ASCIZ | /PC TACS/ | | |
| 3748 | 016744 | 020040 | 040524 | 051503 | | | | | |
| 3749 | 016752 | 000 | | | | | | | |
| 3750 | 016753 | 120 | 000103 | | DH202: | .ASCIZ | /PC/ | | |
| 3751 | | | | | | .EVEN | | | |

```

3752
3753
3754
3755
3756
3757 016756 017120 000200
3758 016762 017322 000200
3759 016766 017524 000010
3760 016772 017726 000040
3761 016776 020130 000020
3762 017002 020332 000100
3763 017006 017120 000000
3764 017012 017046 000200
3765 017016 017054 000200
3766 017022 017062 000010
3767 017026 017070 000040
3768 017032 017076 000020
3769 017036 017104 000100
3770 017042 017112 000000
3771
3772
3773
3774 017046 000002
3775 017046 314
3776 017047 063
3777 017050 314
3778 017051 063
3779 017052 000000
3780
3781 017054 000002
3782 017054 377
3783 017055 000
3784 017056 377
3785 017057 000
3786 017060 000000
3787
3788 017062 000002
3789 017062 231
3790 017063 146
3791 017064 231
3792 017065 146
3793 017066 000000
3794
3795 017070 000002
3796 017070 360
3797 017071 017
3798 017072 360
3799 017073 017
3800 017074 000000
3801
3802 017076 000002
3803 017076 252
3804 017077 125
3805 017100 252
3806 017101 125
3807 017102 000000
    
```

```

;BUFFER LINKS
;THESE LINKS POINT TO THE STARTING ADDRESS OF THE BUFFERS
;AND ALSO INDICATE HOW MANY BYTES TO READ OR WRITE
    
```

```

RLNK1: RBUF1,128.
RLNK2: RBUF2,128.
RLNK3: RBUF3,8.
RLNK4: RBUF4,32.
RLNK5: RBUF5,16.
RLNK6: RBUF6,64.
RLNKX: RBUF1,0
WLNK1: WBUF1,128.
WLNK2: WBUF2,128.
WLNK3: WBUF3,8.
WLNK4: WBUF4,32.
WLNK5: WBUF5,16.
WLNK6: WBUF6,64.
WLNKX: WBUF6,0
    
```

;BUFFERS

```

WBUF1:
    .BYTE  ↑811001100
    .BYTE  ↑800110011
    .BYTE  ↑811001100
    .BYTE  ↑800110011
WCRC1: .WORD  0 ;CRC STORAGE FOR WBUF1

WBUF2:
    .BYTE  ↑811111111
    .BYTE  ↑800000000
    .BYTE  ↑811111111
    .BYTE  ↑800000000
WCRC2: .WORD  0 ;CRC STORAGE FOR WBUF2

WBUF3:
    .BYTE  ↑810011001
    .BYTE  ↑801100110
    .BYTE  ↑810011001
    .BYTE  ↑801100110
WCRC3: .WORD  0 ;CRC STORAGE FOR WBUF3

WBUF4:
    .BYTE  ↑811110000
    .BYTE  ↑800001111
    .BYTE  ↑811110000
    .BYTE  ↑800001111
WCRC4: .WORD  0 ;CRC STORAGE FOR WBUF4

WBUF5:
    .BYTE  ↑810101010
    .BYTE  ↑801010101
    .BYTE  ↑810101010
    .BYTE  ↑801010101
WCRC5: .WORD  0 ;CRC STORAGE FOR WBUF5
    
```


3808
3809 017104 000001
3810 017104 377
3811 017105 377
3812 017106 000
3813 017107 000
3814 017110 000000
3815
3816 017112 000004
3817 017112 000
3818 017113 000
3819 017114 000
3820 017115 000
3821 017116 000000
3822
3823
3824 017120 000202
3825 017322 000202
3826 017524 000202
3827 017726 000202
3828 020130 000202
3829 020332 000202
3830
3831 020534
3832
3833 000001

WBUF6:
.BYTE ↑811111111
.BYTE ↑811111111
.BYTE ↑800000000
.BYTE ↑800000000

WCRC6: .WORD 0 ;CRC STORAGE FOR WBUF6

WBUFx:
.BYTE 0
.BYTE 0
.BYTE 0
.BYTE 0
WCRCx: .WORD 0

RBUF1: .BLKB 130.
RBUF2: .BLKB 130.
RBUF3: .BLKB 130.
RBUF4: .BLKB 130.
RBUF5: .BLKB 130.
RBUF6: .BLKB 130.

LASTADDRESS= .
.END

| | | | | | | | | | | | | | | | | | | | | |
|---------|---------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------|--|--|--|--|--|--|
| CRC.WD | 012222 | 2539* | 2567 | 2592* | 2625 | 2825 | 2846* | 2853# | | | | | | | | | | | | |
| CRLF = | 000200 | 667# | 1091 | 3152 | 3191 | 3568 | 3585 | | | | | | | | | | | | | |
| CURDRV | 001250 | 893# | 1212* | 1214 | 1221* | 1224* | 1226 | 1279 | 2470* | | | | | | | | | | | |
| DDISP = | 177570 | 673# | 855 | 1062 | | | | | | | | | | | | | | | | |
| DH1 | 016102 | 918 | 925 | 932 | 939 | 946 | 953 | 960 | 3666# | | | | | | | | | | | |
| DH2 | 016162 | 967 | 3676# | | | | | | | | | | | | | | | | | |
| DH201 | 016736 | 1003 | 3747# | | | | | | | | | | | | | | | | | |
| DH202 | 016753 | 1008 | 3750# | | | | | | | | | | | | | | | | | |
| DH3 | 016323 | 981 | 3694# | | | | | | | | | | | | | | | | | |
| DH4 | 016461 | 988 | 3711# | | | | | | | | | | | | | | | | | |
| DH5 | 016601 | 974 | 995 | 3726# | | | | | | | | | | | | | | | | |
| DISPLA | 001142 | 855# | 1062* | 1070* | 2188* | 2224* | | | | | | | | | | | | | | |
| DISPRE | 000174 | 811# | 1070 | | | | | | | | | | | | | | | | | |
| DO.CMD | 012224 | 2521 | 2544 | 2568 | 2593 | 2626 | 2650 | 2671 | 2686 | 2707 | 2735 | 2862# | | | | | | | | |
| DO.CRC | 012102 | 2562 | 2612 | 2818# | | | | | | | | | | | | | | | | |
| DRIVE | 001236 | 889# | 1209* | 2469 | 2482* | | | | | | | | | | | | | | | |
| DRVKEY | 001240 | 890# | 891 | 1023* | 1165 | 1178 | 1199 | 1217 | 1223 | 1251 | 2298* | 2303 | 2317 | 2321 | | | | | | |
| | | 2481 | | | | | | | | | | | | | | | | | | |
| DRVPNT | 001244 | 891# | 1179* | 1199* | 1208 | 1227* | | | | | | | | | | | | | | |
| DSWR = | 177570 | 672# | 854 | 1061 | | | | | | | | | | | | | | | | |
| DT1 | 015044 | 920 | 927 | 934 | 941 | 948 | 955 | 962 | 3547# | | | | | | | | | | | |
| DT2 | 015056 | 969 | 3550# | | | | | | | | | | | | | | | | | |
| DT201 | 016652 | 1004 | 3735# | | | | | | | | | | | | | | | | | |
| DT202 | 016660 | 1009 | 3737# | | | | | | | | | | | | | | | | | |
| DT3 | 015074 | 983 | 3554# | | | | | | | | | | | | | | | | | |
| DT4 | 015112 | 990 | 3558# | | | | | | | | | | | | | | | | | |
| DT5 | 015126 | 976 | 997 | 3561# | | | | | | | | | | | | | | | | |
| EMTVEC= | 000030 | 761# | 1045* | 1046* | | | | | | | | | | | | | | | | |
| EM1 | 015540 | 917 | 3622# | | | | | | | | | | | | | | | | | |
| EM10 | 015773 | 966 | 3651# | | | | | | | | | | | | | | | | | |
| EM11 | 016011 | 973 | 3654# | | | | | | | | | | | | | | | | | |
| EM12 | 016035 | 980 | 3658# | | | | | | | | | | | | | | | | | |
| EM13 | 016047 | 987 | 3660# | | | | | | | | | | | | | | | | | |
| EM14 | 016064 | 994 | 3663# | | | | | | | | | | | | | | | | | |
| EM2 | 015566 | 924 | 3626# | | | | | | | | | | | | | | | | | |
| EM201 | 016664 | 1002 | 3739# | | | | | | | | | | | | | | | | | |
| EM202 | 016713 | 1007 | 3743# | | | | | | | | | | | | | | | | | |
| EM3 | 015615 | 931 | 3630# | | | | | | | | | | | | | | | | | |
| EM4 | 015633 | 938 | 3633# | | | | | | | | | | | | | | | | | |
| EM5 | 015654 | 945 | 3636# | | | | | | | | | | | | | | | | | |
| EM6 | 015705 | 952 | 3641# | | | | | | | | | | | | | | | | | |
| EM7 | 015737 | 959 | 3646# | | | | | | | | | | | | | | | | | |
| ERROR = | 100000 | 779# | | | | | | | | | | | | | | | | | | |
| ERRVEC= | 000004 | 754# | 1059 | 1060* | 1071* | 1127* | 1133* | 2154 | 2155* | 2157* | 2160* | | | | | | | | | |
| EXAM | 006464 | 1166 | 1170 | 1175 | 2103# | | | | | | | | | | | | | | | |
| EXIT | 007522 | 2316 | 2321# | | | | | | | | | | | | | | | | | |
| FATAL | 002702 | 1236 | 1255 | 1274# | | | | | | | | | | | | | | | | |
| FGAP = | 004000 | 783# | 2638 | 2661 | 2697 | | | | | | | | | | | | | | | |
| FLAGS | 012014 | 2527 | 2552 | 2574 | 2602 | 2634 | 2656 | 2677 | 2692 | 2713 | 2725 | 2741 | 2792# | | | | | | | |
| FUNCTI= | 000016 | 795# | 2865 | 2880 | | | | | | | | | | | | | | | | |
| FUNCO = | 000002 | 793# | 795 | | | | | | | | | | | | | | | | | |
| FUNC1 = | 000004 | 792# | 795 | | | | | | | | | | | | | | | | | |
| FUNC2 = | 000010 | 791# | 795 | | | | | | | | | | | | | | | | | |
| GNS = | ***** U | 810 | 1090 | 1277 | 3476 | 3477 | 3478 | 3479 | 3481 | 3483 | 3484 | 3485 | 3486 | 3487 | | | | | | |
| | | 3488 | 3489 | 3490 | 3491 | 3492 | 3493 | 3494 | 3495 | 3496 | 3497 | | | | | | | | | |
| GO = | 000001 | 794# | 1249 | 2522 | 2545 | 2594 | 2651 | 2672 | 2687 | 2708 | 2736 | 2865 | 2875 | 2924 | | | | | | |

TA11 MOTION TEST
CZTADD.P11

23-MAR-78 08:34

CZTAD-D MACY11

30A(1052)

23-MAR-78

09:36

PAGE 98

SEQ 0097

SEQ 0097

CROSS REFERENCE TABLE -- USER SYMBOLS

| | | | | | | | | | | | | | | | | | | |
|---------|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--|--|--|--|
| STRAP | 014554 | 1047 | 3453# | | | | | | | | | | | | | | | |
| STRAP2 | 014576 | 3464# | 3475 | | | | | | | | | | | | | | | |
| STRP = | 000025 | 3468# | 3477# | 3478# | 3479# | 3480# | 3481 | 3482# | 3483 | 3484# | 3485# | 3486# | 3487# | 3488# | | | | |
| | | 3489# | 3490# | 3491# | 3492# | 3493# | 3494# | 3495# | 3496# | 3497# | 3498# | | | | | | | |
| STRPAD | 014610 | 3458 | 3475# | | | | | | | | | | | | | | | |
| STSTNM | 001102 | 836# | 1205* | 2068* | 2130 | 2183* | 2188 | 2191 | 2224 | 2253 | | | | | | | | |
| STTYIN | 014160 | 3301 | 3302 | 3319 | 3323# | | | | | | | | | | | | | |
| STYPBN= | ***** U | 3480 | | | | | | | | | | | | | | | | |
| STYPDS= | ***** U | 3480 | | | | | | | | | | | | | | | | |
| STYPE | 013230 | 3138# | 3468 | 3476 | | | | | | | | | | | | | | |
| STYPEC | 013400 | 3159 | 3166 | 3173 | 3178# | 3179 | 3245 | | | | | | | | | | | |
| STYPEX | 013446 | 3184 | 3186 | 3189# | | | | | | | | | | | | | | |
| STYPOC | 014352 | 3398# | 3477 | | | | | | | | | | | | | | | |
| STYPOB | 014366 | 3397 | 3400# | 3479 | | | | | | | | | | | | | | |
| STYPOS | 014326 | 3393# | 3478 | | | | | | | | | | | | | | | |
| SXTSTR | 006630 | 2152# | | | | | | | | | | | | | | | | |
| SSGET4= | 000000 | 2082# | | | | | | | | | | | | | | | | |
| SOFILL | 014551 | 3394# | 3398* | 3408 | 3443# | | | | | | | | | | | | | |
| S4OCAT= | ***** U | 2149 | 2234 | | | | | | | | | | | | | | | |
| . | = 020534 | 806# | 810# | 833# | 875 | 1040 | 1055 | 1056 | 2090 | 2093 | 2191 | 2253 | 2288# | 3191 | | | | |
| | | 3194 | 3323# | 3324 | 3330 | 3514 | 3538 | 3734# | 3824# | 3825# | 3826# | 3827# | 3828# | 3829# | | | | |
| | | 3831 | | | | | | | | | | | | | | | | |

. ABS. 020534 000

ERRORS DETECTED: 0

DSKZ:CZTADD DSKZ:CZTADD.SEQ=DSKZ:CZTADD.P11

RUN-TIME: 20 14 1 SECONDS

RUN-TIME RATIO: 213/36=5.8

CORE USED: 23K (45 PAGES)

DOCUMENT PAGES: 97

H08