

.REM @

IDENTIFICATION

PRODUCT CODE: AC-9357D-MC  
PRODUCT NAME: CZTADD0 TA11 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	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

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 TA11 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=XXXXXXXXNEW= (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.  
TRAP

5.2.4

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 ASCII 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.RWND) REWIND THE TAPE TO BOT

5.2.5.9 SELDRV (T.SELDRV) SELECT A DRIVE

5.2.5.10 BLKCOMP (T.BLKCOMP) 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 TA11 VECTOR IS SET TO "SERV". WHEN THE FUNCTION IS TO BE PERFORMED WITH "INTERRUPT ENABLE"=0 THE VECTOR IS SET TO "BADINT"

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

5.2.6.2 WAITFLAG

THIS ROUTINE IS CALLED AFTER A COMMAND HAS BEEN SENT TO THE TA11. IT WAITS A PREDETERMINED AMOUNT OF TIME A 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 RO 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 TU60. 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 MONO". 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 FORWARD (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 \$ERRTB 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.

9. PROGRAM DESCRIPTION

THIS PROGRAM IS A SEQUENCE OF INDEPENDENT TESTS THAT CHECK THE TA11 FOR PROPER OPERATION.

EACH TESTS IS A SERIES OF "TRAP" CALLS TO ROUTINES THAT PERFORM THE DESIRED FUNCTION.

THE PROGRAM STARTS WITH A SIMPLE SEQUENCE OF FUNCTIONS AND BUILTS IN COMPLEXITY INTRODUCING ONE NEW FUNCTION AT A TIME, UNTIL ALL LEGAL COMBINATIONS HAVE BEEN PERFORMED. THEN, MULTI SPACING IS PERFORMED TO TRY AND GENERATE NOISE THAT MIGHT CAUSE PROBLEMS DUE TO SPEED CHANGES AND FREQUENT START STOPPING.

```

582 .TITLE TA11 MOTION TEST CZTAD-D
583 ;*COPYRIGHT (C) 1973,1978
584 ;*DIGITAL EQUIPMENT CORP.
585 ;*MAYNARD, MASS. 01754
586 ;*
587 ;*PROGRAM BY JIM LACEY
588 ;*
589 ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
590 ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
591 ;*
592 ;*****
593 ;*****
594 ;*****
595 ;*****
596 ;*****
597 ;*****
598 ;*****
599 ;*****
600 ;*****
601 ;*****
602 ;*****
603 ;*****
604 ;*****
605 ;*****
606 ;*****
607 ;*****
608 ;*****
609 ;*****
610 ;*****
611 ;*****
612 ;*****
613 ;*****
614 ;*****
615 ;*****
616 ;*****
617 ;*****
618 ;*****
619 ;*****

```

GENERAL INFORMATION ABOUT THE TA11/TU60 CASSETTE

ADDRESS MNEMONIC DESCRIPTION

```

777500 TACS CONTROL AND STATUS REGISTER
777502 TADB DATA BUFFER REGISTER
260 TAVEC INTERRUPT VECTOR

```

TACS REGISTER DESCRIPTION

BIT	NAME	INIT STATE	READ AND/OR WRITE?
15	ERROR	?	READ ONLY
14	BLOCK CHECK ERROR	0	READ ONLY
13	CLEAR LEADER	?	READ ONLY
12	WRITE LOCK	?	READ ONLY
11	FILE GAP	0	READ ONLY
10	TIMING ERROR	0	READ ONLY

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 .SBTTL OPERATIONAL SWITCH SETTINGS
639 ;*
640 ;* SWITCH USE
641 ;* -----
642 ;* 15 HALT ON ERROR
643 ;* 14 LOOP ON TEST
644 ;* 13 INHIBIT ERROR TYPEDOUTS
645 ;* 11 INHIBIT ITERATIONS
646 ;* 10 BELL ON ERROR
647 ;* 9 LOOP ON ERROR
648 ;* 7 LOCK ON CURRENT DRIVE
649 ;* 6 DELAY AT END OF EACH FUNCTION
650 ;* 5 RUN WITHOUT INTERRUPTS
651 ;* 4 IGNORE BLOCK CHECK ERRORS
652 ;* 3 INHIBIT DATA COMPARE
653
654
655
656 .SBTTL BASIC DEFINITIONS
657 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
658 STACK= 1100
659 001100 .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
660 .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
661
662 ;*MISCELLANEOUS DEFINITIONS
663 HT= 11 ;;CODE FOR HORIZONTAL TAB
664 000011 LF= 12 ;;CODE FOR LINE FEED
665 000012 CR= 15 ;;CODE FOR CARRIAGE RETURN
666 000015 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
667 000200 PS= 177776 ;;PROCESSOR STATUS WORD
668 177776 .EQUIV PS,PSW
669 STKLMT= 177774 ;;STACK LIMIT REGISTER
670 177774 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
671 177772 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
672 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
673
674 ;*GENERAL PURPOSE REGISTER DEFINITIONS
675 R0= %0 ;;GENERAL REGISTER
676 000000 R1= %1 ;;GENERAL REGISTER
677 000001 R2= %2 ;;GENERAL REGISTER
678 000002 R3= %3 ;;GENERAL REGISTER
679 000003 R4= %4 ;;GENERAL REGISTER
680 000004 R5= %5 ;;GENERAL REGISTER
681 000005 R6= %6 ;;GENERAL REGISTER
682 000006 R7= %7 ;;GENERAL REGISTER
683 000007 SP= %6 ;;STACK POINTER
684 000006 PC= %7 ;;PROGRAM COUNTER
685 000007
686
687 ;*PRIORITY LEVEL DEFINITIONS
688 PRO= 0 ;;PRIORITY LEVEL 0
689 000040 PR1= 40 ;;PRIORITY LEVEL 1
690 000100 PR2= 100 ;;PRIORITY LEVEL 2
691 000140 PR3= 140 ;;PRIORITY LEVEL 3
692 000200 PR4= 200 ;;PRIORITY LEVEL 4
693 000240 PR5= 240 ;;PRIORITY LEVEL 5
  
```

```

694 000300 PR6= 300 ;;PRIORITY LEVEL 6
695 000340 PR7= 340 ;;PRIORITY LEVEL 7
696
697 ;*"SWITCH REGISTER" SWITCH DEFINITIONS
698 SW15= 100000
699 040000 SW14= 40000
700 020000 SW13= 20000
701 010000 SW12= 10000
702 004000 SW11= 4000
703 002000 SW10= 2000
704 001000 SW09= 1000
705 000400 SW08= 400
706 000200 SW07= 200
707 000100 SW06= 100
708 000040 SW05= 40
709 000020 SW04= 20
710 000010 SW03= 10
711 000004 SW02= 4
712 000002 SW01= 2
713 000001 SW00= 1
714 .EQUIV SW09,SW9
715 .EQUIV SW08,SW8
716 .EQUIV SW07,SW7
717 .EQUIV SW06,SW6
718 .EQUIV SW05,SW5
719 .EQUIV SW04,SW4
720 .EQUIV SW03,SW3
721 .EQUIV SW02,SW2
722 .EQUIV SW01,SW1
723 .EQUIV SW00,SW0
724
725 ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
726 100000 BIT15= 100000
727 040000 BIT14= 40000
728 020000 BIT13= 20000
729 010000 BIT12= 10000
730 004000 BIT11= 4000
731 002000 BIT10= 2000
732 001000 BIT09= 1000
733 000400 BIT08= 400
734 000200 BIT07= 200
735 000100 BIT06= 100
736 000040 BIT05= 40
737 000020 BIT04= 20
738 000010 BIT03= 10
739 000004 BIT02= 4
740 000002 BIT01= 2
741 000001 BIT00= 1
742 .EQUIV BIT09,BIT9
743 .EQUIV BIT08,BIT8
744 .EQUIV BIT07,BIT7
745 .EQUIV BIT06,BIT6
746 .EQUIV BIT05,BIT5
747 .EQUIV BIT04,BIT4
748 .EQUIV BIT03,BIT3
749 .EQUIV BIT02,BIT2
  
```



```

750 .EQUIV BIT01,BIT1
751 .EQUIV BIT00,BIT0
752
753 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
754 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
755 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
756 TBITVEC=14 ;:"T" BIT
757 TRTVEC= 14 ;:TRACE TRAP
758 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
759 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
760 PWRVEC= 24 ;:POWER FAIL
761 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
762 TRAPVEC=34 ;:"TRAP" TRAP
763 TKVEC= 60 ;:TTY KEYBOARD VECTOR
764 TPVEC= 64 ;:TTY PRINTER VECTOR
765 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
766 ;:*****
767
768 ;#####TA11 FUNCTIONS#####
769 XWFG= 0 ;WRITE FILE GAP FUNCTION
770 XWRITE= 2 ;WRITE FUNCTION
771 XREAD= 4 ;READ FUNCTION
772 XBSFG= 6 ;BACK SPACE FILE GAP FUNCTION
773 XBSBG= 10 ;BACK SPACE BLOCK GAP FUNCTION
774 XSFFG= 12 ;SPACE FWD FILE GAP FUNCTION
775 XSFBG= 14 ;SPACE FWD BLOCK GAP FUNCTION
776 XRWND= 16 ;REWIND FUNCTION
777
778 ;#####TA11 BIT ASSIGNMENT#####
779 ERROR= BIT15
780 CRCERR= BIT14
781 LEADER= BIT13
782 WRTLOCK=BIT12
783 FGAP= BIT11
784 TIMERR= BIT10
785 OFFLINE=BIT09
786 UNIT= BIT08
787 TR.REQ= BIT07
788 INT.EN= BIT06
789 READY= BIT05
790 ILBS= BIT04
791 FUNC2= BIT03
792 FUNC1= BIT02
793 FUNC0= BIT01
794 GO= BIT00
795 FUNCTION= FUNC2+FUNC1+FUNC0
796
797 ;//////////////////////////////////////////////////////////////////
798 ;SPECIAL REGISTERS
799
800 TACS= %4 ;R4 IS USED AS A POINTER TO THE TACS REGISTER
801 TADB= %5 ;R5 IS USED AS A POINTER TO THE TADB REGISTER.
802
803 ;//////////////////////////////////////////////////////////////////
    
```

```

804 .SBTTL TRAP CATCHER
805
806 .=0
807 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
808 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
809 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
810 .=174
811 DISPREG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
812 SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER
813
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 @#WFGSUB ;: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 ;:*****
    
```

```

827          .SBTTL COMMON TAGS
828
829          ;*****
830          ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
831          ;*USED IN THE PROGRAM.
832
833          .=1100
834 001100    SCMTAG:          ;;START OF COMMON TAGS
835 001100 000000 $PASS: .WORD 0          ;;CONTAINS PASS COUNT
836 001102 000    STSNM: .BYTE 0          ;;CONTAINS THE TEST NUMBER
837 001103 000    SERFLG: .BYTE 0         ;;CONTAINS ERROR FLAG
838 001104 000000 SICNT: .WORD 0         ;;CONTAINS SUBTEST ITERATION COUNT
839 001106 000000 SLPADR: .WORD 0         ;;CONTAINS SCOPE LOOP ADDRESS
840 001110 000000 SLPERR: .WORD 0         ;;CONTAINS SCOPE RETURN FOR ERRORS
841 001112 000000 SERTTL: .WORD 0         ;;CONTAINS TOTAL ERRORS DETECTED
842 001114 000    SITEMB: .BYTE 0        ;;CONTAINS ITEM CONTROL BYTE
843 001115 001    SERMAX: .BYTE 1        ;;CONTAINS MAX. ERRORS PER TEST
844 001116 000000 SERRPC: .WORD 0        ;;CONTAINS PC OF LAST ERROR INSTRUCTION
845 001120 000000 SGDADR: .WORD 0        ;;CONTAINS ADDRESS OF 'GOOD' DATA
846 001122 000000 SBDADR: .WORD 0        ;;CONTAINS ADDRESS OF 'BAD' DATA
847 001124 000000 SGGDAT: .WORD 0        ;;CONTAINS 'GOOD' DATA
848 001126 000000 SBDAT: .WORD 0        ;;CONTAINS 'BAD' DATA
849 001130 000000          .WORD 0        ;;RESERVED--NOT TO BE USED
850 001132 000000          .WORD 0
851 001134 000    SAUTOB: .BYTE 0        ;;AUTOMATIC MODE INDICATOR
852 001135 000    SINTAG: .BYTE 0        ;;INTERRUPT MODE INDICATOR
853 001136 000000          .WORD 0
854 001140 177570 SWR: .WORD DSWR          ;;ADDRESS OF SWITCH REGISTER
855 001142 177570 DISPLAY: .WORD DDISP      ;;ADDRESS OF DISPLAY REGISTER
856 001144 177560 $TKS: 177560          ;;TTY KBD STATUS
857 001146 177562 $TKB: 177562          ;;TTY KBD BUFFER
858 001150 177564 $TPS: 177564          ;;TTY PRINTER STATUS REG. ADDRESS
859 001152 177566 $TPB: 177566          ;;TTY PRINTER BUFFER REG. ADDRESS
860 001154 000    $NULL: .BYTE 0          ;;CONTAINS NULL CHARACTER FOR FILLS
861 001155 002    $FILLS: .BYTE 2        ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
862 001156 012    $FILLC: .BYTE 12       ;;INSERT FILL CHARS. AFTER A "LINE FEED"
863 001157 000    $TPFLG: .BYTE 0       ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
864 001160 000000 $REGAD: .WORD 0          ;;CONTAINS THE ADDRESS FROM
865          ;;WHICH ($REGO) WAS OBTAINED
866 001162 000000 $REGO: .WORD 0          ;;CONTAINS (($REGAD)+0)
867 001164 000000 $REG1: .WORD 0        ;;CONTAINS (($REGAD)+2)
868 001166 000000 $TMPO: .WORD 0        ;;USER DEFINED
869 001170 000000 $TIMES: 0           ;;MAX. NUMBER OF ITERATIONS
870 001172 000000 $ESCAPE:0          ;;ESCAPE ON ERROR ADDRESS
871 001174 177607 000377 $BELL: .ASCIZ <207><377><377> ;;CODE FOR BELL
872 001200 077    $QUES: .ASCIZ /?/      ;;QUESTION MARK
873 001201 015    $CRLF: .ASCIZ <15>    ;;CARRIAGE RETURN
874 001202 000012 $LF: .ASCIZ <12>          ;;LINE FEED
875          ;*****
876 001204 000000 SAVPC: .WORD 0          ;;STORAGE FOR THE PC
877 001206 000000 BYTNUM: .WORD 0        ;;THE NUMBER OF BYTES "READ" OR "WRITTEN"
878 001210 000000 RRCRC: .WORD 0        ;;CRC CALCULATED FOR "WRITE"
879 001212 000000 RRCRC1: .WORD 0       ;;CRC CALCULATED FOR "READ"
880 001214 000000 RRCRC2: .WORD 0       ;;CRC FROM THE TA11 FOR "READ"
881
882 001216 000001 $MXCNT: 1           ;;MAX. NUMBER OF ITERATIONS
    
```

```

883 001220 177500 TACSL: 177500          ;;LOW BYTE ADDRESS OF TACS
884 001222 177501 TACSH: 177501          ;;HIGH BYTE ADDRESS OF TACS
885 001224 177502 TADBL: 177502          ;;LOW BYTE ADDRESS OF TADB
886 001226 177503 TADBH: 177503          ;;HIGH BYTE ADDRESS OF TADB
887 001230 000260 000262 TAVEC: 260,262    ;;TA11 VECTOR ADDRESS
888 001234 000300 TAPRID: 300          ;;TA11 BR LEVEL 6
889 001236 000000 DRIVE: 0           ;;NEXT DRIVE TO TEST
890 001240 000000 000000 DRVKEY: 0,0      ;;DRIVE SELECT KEY:
891 001244 001240 DRVPNT: DRVKEY
892 001246 000000 ASKKEY: 0
893 001250 177777 000000 CURDRV: -1,0      ;;CURRENT DRIVE BEING TESTED
894 001254 000000 WAITKEY:          ;;WAIT ON INTERRUPT KEY
895 001256 001260 RWDFLG: RWDA 0        ;;REWIND FLAG POINTER
896 001260 000000 RWDA: 0           ;;DRIVE "A" REWIND FLAG
897 001262 000000 RWDB: 0           ;;DRIVE "B" REWIND FLAG
898 001264 000000 000000 STALL: 0,0     ;;STALL TIME
    
```

```

CZTADD.P11 23-MAR-78 08:34 ERROR POINTER TABLE
      899          .SBTTL ERROR POINTER TABLE
      900
      901          :*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
      902          :*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
      903          :*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
      904          :*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
      905          :*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
      906
      907          :* EM          ;;POINTS TO THE ERROR MESSAGE
      908          :* DH          ;;POINTS TO THE DATA HEADER
      909          :* DT          ;;POINTS TO THE DATA
      910          :* DF          ;;POINTS TO THE DATA FORMAT
      911
      912
      913 001270      $ERRTB:
      914          :NOTE: ALL NUMBERS WILL BE TYPED AS 6 DIGIT OCTAL NUMBERS
      915
      916          :ITEM 1
      917 EM1          ;IMPROPER FLAG SETTING
      918 001272 016102 DH1          ;TEST ERROR
      919          ;PC PC TACS TADB
      920 001274 015044 DT1          ;SAVPC $ERRPC $REGO $REG1
      921 001276 000000 0
      922
      923          :ITEM 2
      924 001300 015666 EM2          ;IMPROPER FLAG OCCURRED
      925 001302 016102 DH1          ;TEST ERROR
      926          ;PC PC TACS TADB
      927 001304 015044 DT1          ;SAVPC $ERRPC $REGO $REG1
      928 001306 000000 0
      929
      930          :ITEM 3
      931 001310 015615 EM3          ;MISSED A FLAG
      932 001312 016102 DH1          ;TEST ERROR
      933          ;PC PC TACS TADB
      934 001314 015044 DT1          ;SAVPC $ERRPC $REGO $REG1
      935 001316 000000 0
      936
      937          :ITEM 4
      938 001320 015633 EM4          ;INTERRUPT FAILED
      939 001322 016102 DH1          ;TEST ERROR
      940          ;PC PC TACS TADB
      941 001324 015044 DT1          ;SAVPC $ERRPC $REGO $REG1
      942 001326 000000 0
      943
      944          :ITEM 5
      945 001330 015654 EM5          ;PREMATURE READY OCCURRED
      946 001332 016102 DH1          ;TEST ERROR
      947          ;PC PC TACS TADB
      948 001334 015044 DT1          ;SAVPC $ERRPC $REGO $REG1
      949 001336 000000 0
      950
      951          :ITEM 6
      952 001340 015705 EM6          ;DIDN'T STOP IN A FILE GAP
      953 001342 016102 DH1          ;TEST ERROR
      954          ;PC PC TACS TADB
  
```

```

CZTADD.P11 23-MAR-78 08:34 ERROR POINTER TABLE
      955 001344 015044 DT1          ;SAVPC $ERRPC $REGO $REG1
      956 001346 000000 0
      957
      958          :ITEM 7
      959 001350 015737 EM7          ;DIDN'T STOP ON CLEAR LEADER
      960 001352 016102 DH1          ;TEST ERROR
      961          ;PC PC TACS TADB
      962 001354 015044 DT1          ;SAVPC $ERRPC $REGO $REG1
      963 001356 000000 0
      964
      965          :ITEM 10
      966 001360 015773 EM10         ;BAD DATA READ
      967 001362 016162 DH2          ;TEST ERROR
      968          ;PC PC TACS DATA EXPT'D RCV'D BYTE
      969 001364 015056 DT2          ;SAVPC $ERRPC $REGO $GDDAT $BDDAT BYTNUM
      970 001366 000000 0
      971
      972          :ITEM 11
      973 001370 016011 EM11         ;ILLEGAL BUFFER
      974 001372 016601 DH5          ;TEST ERROR
      975          ;PC PC TACS
      976 001374 015126 DT5          ;SAVPC $ERRPC $REGO
      977 001376 000000 0
      978
      979          :ITEM 12
      980 001400 016035 EM12         ;CRC ERROR
      981 001402 016323 DH3          ;TEST ERROR
      982          ;PC PC TACS WRITE READ TU60
      983 001404 015074 DT3          ;SAVPC $ERRPC $REGO CRC RCRC0 RCRC1 RCRC2
      984 001406 000000 0
      985
      986          :ITEM 13
      987 001410 016047 EM13         ;SHORT RECORD
      988 001412 016461 DH4          ;TEST ERROR
      989          ;PC PC TACS TADB LEFT BYTES
      990 001414 015112 DT4          ;SAVPC $ERRPC $REGO $REG1 $TMP0
      991 001416 000000 0
      992
      993          :ITEM 14
      994 001420 016064 EM14         ;BAD INTERRUPT
      995 001422 016601 DH5          ;TEST ERROR
      996          ;PC PC TACS
      997 001424 015126 DT5          ;SAVPC $ERRPC $REGO
      998 001426 000000 0
      999
      1000 001430      ITEMS2:
      1001          ;ITEMS 201-202
      1002 001430 016664 EM201         ;TA11 FAILED TO RESPOND
      1003 001432 016736 DH201         ;PC TACS
      1004 001434 016652 DT201         ;$ERRPC TACS
      1005 001436 000000 0
      1006          ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS
      1007 001440 016713 EM202         ;NO DRIVES AVAILABLE
      1008 001442 016753 DH202         ;PC
      1009 001444 016660 DT202         ;$ERRPC
      1010 001446 000000 0
  
```

```

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 ($CMTAG) AREA
1037 001522 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
1038 001526 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
1039 001530 022706 001140 CMP #SWR,R6 ;;DONE?
1040 001534 001374 BNE -.6 ;;LOOP BACK IF NO
1041 001536 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
1042 ;;INITIALIZE A FEW VECTORS
1043 001542 012737 006570 000020 MOV #SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1044 001550 012737 000340 000022 MOV #340,@#IOTVEC+2 ;;LEVEL 7
1045 001556 012737 007046 000030 MOV #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1046 001564 012737 000340 000032 MOV #340,@#EMTVEC+2 ;;LEVEL 7
1047 001572 012737 014554 000034 MOV #TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1048 001600 012737 000340 000036 MOV #340,@#TRAPVEC+2;LEVEL 7
1049 001606 012737 014662 000024 MOV #SPWRN,@#PWRVEC ;;POWER FAILURE VECTOR
1050 001614 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;LEVEL 7
1051 001622 013737 006406 006400 MOV #ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
1052 001630 005037 001170 CLR TIMES ;;INITIALIZE NUMBER OF ITERATIONS
1053 001634 005037 001172 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1054 001640 112737 000001 001115 MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
1055 001646 012737 001646 001106 MOV #.,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1056 001654 012737 001654 001110 MOV #.,$LPERR ;;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 #64$,@#ERRVEC ;;SET UP ERROR VECTOR
1061 001674 012737 177570 001140 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
1062 001702 012737 177570 001142 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1063 001710 022777 177777 177222 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
1064 001716 001012 BNE 66$ ;;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$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
1070 001736 012737 000174 001142 MOV #DISPREG,DISPLAY
1071 001744 012637 000004 66$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
1072
1073 .SBTTL TYPE PROGRAM NAME
1074 ;;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 68$
1087 002014 112737 000001 001134 67$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
1088 002022 68$: BR
1089 002022 000405 BR HERE ;;GET OVER THE ASCIZ
1090 ;;MSGID: .ASCIZ <CRLF>/C2TAD-D/<CRLF>
1091 002036 HERE:
    
```

```
1092 ;*****
1093 ;*****
1094 ;
1095 ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
1096 ;
1097 ; R5=3 MANUAL OPERATIONS
1098 ; R5=2 ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
1099 ; R5=1 ASK FOR DRIVE(S)
1100 ; R5=0 DON'T ASK FOR ANYTHING
1101 ;
1102 ;*****
1103 002036 010504 BEGINX: MOV R5,R4 ;COPY R5
1104 002040 005305 DEC R5 ;ASK FOR DRIVES?
1105 002042 002406 BLT CHKADR ;BR IF NO
1106 002044 004737 007422 JSR PC,@#ASKDRV ;GO GET DRIVES TO BE TESTED
1107 002050 005305 DEC R5 ;ASK FOR ADDRESSES?
1108 002052 002402 BLT CHKADR ;BR IF NO
1109 002054 004737 007532 JSR PC,@#ASKADR ;GO GET TA11 ADDRESSES
1110 ;*****
1111 ;*****
1112 ;
1113 ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
1114 ;
1115 ;I. TIMEOUT OCCURRED
1116 ; A. TYPE ERROR MESSAGE
1117 ; B. EXAMINE R4
1118 ; 1. R4>0 GOTO BEGINX
1119 ; 2. R4=0 EXAMINE (42)
1120 ; A. (42)=0 GOTO BEGINX
1121 ; B. (42)>0 GOTO SENDAD
1122 ;
1123 ;II. TIMEOUT DIDN'T OCCUR
1124 ; A. CONTINUE
1125 ;
1126 ;*****
1127 002060 012737 002076 000004 CHKADR: MOV #1$,@#ERRVEC ;IN CASE OF TIMEOUTS
1128 002066 005000 CLR R0 ;USE AS A SWITCH
1129 002070 005777 177124 TST @TACSL ;SEE IF TA11 RESPONDS
1130 002074 000402 BR 2$ ;BR IF NO TIMEOUT
1131 002076 005200 1$: INC R0 ;COME HERE ON TIMEOUT
1132 002100 022626 CMP (SP)+,(SP)+ ;CLEANUP THE STACK
1133 002102 012737 000006 000004 2$: MOV #ERRVEC+2,@#ERRVEC ;RESTORE TIMEOUT VECTOR
1134 002110 005700 TST R0 ;DID A TIMEOUT OCCUR?
1135 002112 001412 BEQ 3$ ;BR IF NO
1136 002114 104201 ERROR 201 ;TA11 FAILED TO RESPOND
1137 002116 012705 000002 MOV #2,R5 ;DRIVES & ADDRESSES
1138 002122 005704 TST R4 ;OPERATOR INPUTS?
1139 002124 001344 BNE BEGINX ;BR IF YES
1140 002126 013700 000042 MOV @#42,R0 ;GET MONITOR RETURN ADDRESS
1141 002132 001741 BEQ BEGINX ;BR IF NO MONITOR
1142 002134 000137 006432 JMP @#SENDAD ;GO TO END
1143 002140 3$:
```

```
1144 ;*****
1145 ;*****
1146 ;
1147 ;MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED
1148 ;
1149 ;I. DESIRED DRIVES CAN NOT BE TESTED
1150 ; A. TYPE ERROR MESSAGE
1151 ; B. EXAMINE R4
1152 ; 1. R4>0 GOTO BEGINX
1153 ; 2. R4=0 EXAMINE (42)
1154 ; A. (42)=0 GOTO BEGINX
1155 ; B. (42)>0 GOTO SENDAD
1156 ;
1157 ;II. BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
1158 ; A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
1159 ; B. CONTINUE IN PROGRAM
1160 ;
1161 ;III. DESIRED DRIVE(S) CAN BE TESTED
1162 ; A. CONTINUE IN PROGRAM
1163 ;
1164 ;*****
1165 002140 012700 001240 CHKDRV: MOV #DRVKEY,R0 ;PICKUP ADDRESS OF ASCII DRIVE KEY
1166 002144 004737 006464 JSR PC,@#EXAM ;GO EXAMINE FIRST DRIVE
1167 002150 000410 BR 1$ ;OK TO TEST---GO CHECK NEXT
1168 002152 116010 000001 MOV 1(R0),(R0) ;REPLACE 1ST WITH 2ND
1169 002156 001412 BEQ 2$ ;BR IF NO 2ND DRIVE SELECTED
1170 002160 004737 006464 JSR PC,@#EXAM ;GO EXAMINE DRIVE
1171 002164 000407 BR 2$ ;OK TO TEST
1172 002166 005010 CLR (R0) ;CLEAR DRIVE CODES
1173 002170 000405 BR 2$
1174 002172 005200 1$: INC R0 ;POINT TO 2ND
1175 002174 004737 006464 JSR PC,@#EXAM ;GO EXAMINE DRIVE
1176 002200 000401 BR 2$ ;OK TO TEST
1177 002202 105010 CLRB (R0) ;CLEAR 2ND
1178 002204 012700 001240 2$: MOV #DRVKEY,R0 ;RESET ADDRESS POINTERS
1179 002210 010037 001244 MOV R0,@#DRVPT
1180 002214 121060 000001 CMPB (R0),1(R0) ;1ST = 2ND?
1181 002220 001002 BNE 3$ ;BR IF NO
1182 002222 105060 000001 CLRB 1(R0) ;YES---CLEAR 2ND
1183 002226 005710 3$: TST (R0) ;ANY DRIVES?
1184 002230 001401 BEQ 5$ ;BR IF NO
1185 002232 000412 BR MANUAL
1186 002234 104202 5$: ERROR 202 ;NO DRIVES AVAILABLE
1187 002236 012705 000002 MOV #2,R5 ;DRIVES & ADDRESS
1188 002242 005704 TST R4 ;OPERATOR INPUTS?
1189 002244 001274 BNE BEGINX ;BR IF YES
1190 002246 013700 000042 MOV @#42,R0 ;GET MONITOR RETURN ADDRESS
1191 002252 001671 BEQ BEGINX ;NO MONITOR
1192 002254 000137 006432 JMP @#SENDAD ;GO TO END
1193 002260 020427 000003 MANUAL: CMP R4,#3
1194 002264 001002 BNE OK
1195 002266 013704 177777 MOV -1,R4
1196 002272 010437 001246 OK: MOV R4,@#ASKKEY
1197 002276 000405 BR START
1198 002300 104401 002024 PWRST: TYPE #MSGID ;POWER FAIL RESTART
1199 002304 012737 001240 001244 MOV #DRVKEY,@#DRVPT
```

```

1200 002312 012777 012356 176710 START: MOV #BADINT,@TAVEC ;SETUP TA11 TRAP VECTOR
1201 002320 013777 001234 176704 MOV @#TAPRIO,@TAVEC+2 ;LOCKOUT ALL I/O INT
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 @#STSTNM ;ZERO THE TEST NUMBER
1206 002350 005037 006614 CLR @#LOOPKEY ;CLEAR THE LOOP UNDER TTY CONTROL KEY
1207 002354 012737 000001 001216 MOV #1,@#MXCNT ;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 MOV (R1),@#CURDRV ;GET CURRENT DRIVE FOR TYPE-OUT
1213 002376 104401 015261 TYPE ,MTSTDRV ;TYPE "TESTING DRIVE"
1214 002402 104401 001250 TYPE ,CURDRV ;TYPE THE CURRENT DRIVE
1215 002406 032777 000200 176524 BIT #SW07,@SWR ;LOCK ON ONE DRIVE?
1216 002414 001021 BNE 2$ ;BR IF YES
1217 002416 105737 001241 TSTB @#DRVKEY+1 ;TWO DRIVES AVAILABLE?
1218 002422 001416 BEQ 2$ ;BR IF NO
1219 002424 005237 001216 INC @#MXCNT ;SET FOR TWO ITERATIONS PER TEST (ONE PER DRIVE)
1220 002430 005201 INC R1 ;INCREMENT THE POINTER
1221 002432 111137 001250 MOV (R1),@#CURDRV ;SET SECOND DRIVE FOR TYPE-OUT
1222 002436 001004 BNE 1$ ;GO TYPE IT
1223 002440 012701 001240 MOV #DRVKEY,R1 ;RESET POINTER
1224 002444 111137 001250 MOV (R1),@#CURDRV ;NOW GET THE SECOND DRIVE
1225 002450 104401 015302 1$: TYPE ,MANDRV ;TYPE "AND DRIVE"
1226 002454 104401 001250 TYPE ,CURDRV ;TYPE DRIVE
1227 002460 010137 001244 2$: MOV R1,@#DRVPNT ;SAVE POINTER FOR NEXT TRIP THRU
1228 002464 012737 002664 001106 MOV #5$,@#SLPADR ;SETUP THE SCOPE LOOP ADDRESS
1229 002472 005737 001100 TST @#SPASS ;ONCE ONLY
1230 002476 001131 BNE TST1
1231
1232 ;////////////////////////////////////
;ON THE FIRST PASS OF THE PROGRAM DETERMINE THE AMOUNT OF TIME TO
1233 ;WAIT ON A FLAG ("READY"/"TRANSFER REQUEST") AND THE STALL TIME TO USE
1234 ;WHEN STALLING AFTER A FUNCTION IS COMPLETED
1235 SELDRV ;SELECT DRIVE
1236 002500 104422 MOV #FATAL,@#$ESCAPE ;SETUP ESCAPE ON ERROR
1237 002510 005077 176542 CLR @#RWDFLG ;SET FOR NO REWIND ERRORS
1238 002514 012737 077777 012012 MOV #CBIT15,@#MXCNT ;SETUP IMPOSSIBLE COUNT
1239 002522 112714 000016 MOV @#XRWND,@TACS ;INSURE NO ERROR DUE TO "CLEAR LEADER"
1240 002526 104421 WFG ;GO TO BOT
1241 002530 104412 WFG ;NOW TIME A WFG
1242 002532 163737 012010 012012 SUB @#HICNT,@#MXCNT ;GET THE TIME IT TOOK TO WFG
1243 002540 005237 012012 INC @#MXCNT ;MAKE IT BIGGER
1244 002544 006137 012012 ROL @#MXCNT
1245 002550 006137 012012 ROL @#MXCNT
1246 002554 005000 CLR RO
1247 002556 005001 CLR R1
1248 002560 012777 002616 176442 MOV #4$,@TAVEC ;SETUP TA11 VECTOR
1249 002566 112714 000101 MOV @#XWFG!INT.ENIGO,@TACS ;START A "WFG"
1250 002572 005037 177776 CLR @#PS ;ALLOW INTERRUPTS
1251 002576 162700 000001 3$: SUB #1,RO ;START TIMING HOW LONG
1252 002602 005601 SBC R1 ; IT TAKES TO WRITE A FILE GAP
1253 002604 001374 BNE 3$
1254 002606 012737 000340 177776 MOV #340,@#PS ;LOCK OUT INTERRUPTS
1255 002614 000432 BR FATAL ;IT TOOK TO LONG
    
```

```

1256 002616 006201 4$: ASR R1 ;DIVIDE THE TIME BY 2
1257 002620 006000 ROR RO
1258 002622 010037 001264 MOV RO,@#STALL ;AND SAVE IT AS THE STALL TIME
1259 002626 010137 001266 MOV R1,@#STALL+2
1260 002632 012777 012356 176370 MOV #BADINT,@TAVEC ;SET TRAP CATCHER
1261 002640 105737 001241 TSTB @#DRVKEY+1 ;TWO DRIVES?
1262 002644 001407 BEQ 5$ ;BR IF NO
1263 002646 104422 SELDRV ;SELECT THE OTHER DRIVE
1264 002650 005077 176402 CLR @#RWDFLG ;SET REWIND FLAG
1265 002654 112714 000016 MOV @#XRWND,@TACS ;AVOID "CLEAR LEADER" ERROR
1266 002660 104421 WFG ;GO TO BOT
1267 002662 104412 WFG ;GO TO OXIDE
1268 002664 005737 001246 5$: TST @#ASKKEY ;GO START TESTING IF NO MANUAL
1269 002670 002034 BGE TST1 ; OPERATIONS REQUESTED
1270 002672 005000 CLR RO
1271 002674 006137 012012 ROL @#MXCNT ;INCREASE THE WAIT TIME
1272 002700 000000 HALT ;GIVE CONTROL TO THE OPERATOR
1273
1274 002702 FATAL:
1275 002702 104401 002710 TYPE ,65$ ;TYPE ASCIZ STRING
1276 002706 000414 BR 64$ ;GET OVER THE ASCIZ
1277 ;:65$: .ASCIZ <15><12>*FATAL ERROR ON DRIVE *
1278 002740 64$:
1279 002740 104401 001250 TYPE ,CURDRV ;CURRENT DRIVE
1280 002744 013700 000042 MOV @#42,RO ;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
    
```

```

1285 ;*****
1286 ;*TEST 1 REWIND,WRITE,REWIND,READ
1287 ;*****
1288 002762 000004 TST1: SCOPE
1289 002764 012737 003016 001172 MOV #TST2,$ESCAPE ;;ESCAPE TO TEST 2 ON ERROR
1290 002772 104422 SELDRV ;SELECT DRIVE FOR TESTING
1291 002774 104421 REWIND ;GO TO "CLEAR LEADER"
1292 002776 104413 017012 WRITE ,WLNK1 ;WRITE ON TAPE
1293 003002 104421 REWIND ;GO TO BEGINNING OF TAPE
1294 003004 104414 016756 READ ,RLNK1 ;READ
1295 003010 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1296 003012 016756 RLNK1 ;LINK TO THE READ BUFFER
1297 003014 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1298 ;*****
1299 ;*TEST 2 TEST BSBG AFTER WRITE AND READ AFTER BSBG
1300 ;*****
1301 003016 000004 TST2: SCOPE
1302 003020 012737 003052 001172 MOV #TST3,$ESCAPE ;;ESCAPE TO TEST 3 ON ERROR
1303 003026 104422 SELDRV ;SELECT DRIVE FOR TESTING
1304 003030 104421 REWIND ;GO TO "BOT"
1305 003032 104413 017012 WRITE ,WLNK1 ;WRITE ONE BLOCK
1306 003036 104416 BSBG ;BACK OVER THE BLOCK
1307 003040 104414 016756 READ ,RLNK1 ;NOW READ
1308 003044 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1309 003046 016756 RLNK1 ;LINK TO THE READ BUFFER
1310 003050 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1311 ;*****
1312 ;*TEST 3 TEST BSBG AFTER WRITE AND READ AFTER BSBG
1313 ;*****
1314 003052 000004 TST3: SCOPE
1315 003054 012737 003106 001172 MOV #TST4,$ESCAPE ;;ESCAPE TO TEST 4 ON ERROR
1316 003062 104422 SELDRV ;SELECT DRIVE FOR TESTING
1317 003064 104421 WRITE ,WLNK1 ;GO TO "BOT"
1318 003066 104413 017012 WRITE ,WLNK1 ;WRITE ONE BLOCK
1319 003072 104415 BSBG ;BACK OVER THE BLOCK
1320 003074 104414 016756 READ ,RLNK1 ;NOW READ
1321 003100 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1322 003102 016756 RLNK1 ;LINK TO THE READ BUFFER
1323 003104 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1324 ;*****
1325 ;*TEST 4 TEST BSBG AFTER READ
1326 ;*****
1327 003106 000004 TST4: SCOPE
1328 003110 012737 003156 001172 MOV #TST5,$ESCAPE ;;ESCAPE TO TEST 5 ON ERROR
1329 003116 104422 SELDRV ;SELECT DRIVE FOR TESTING
1330 003120 104421 REWIND ;GO TO "BOT"
1331 003122 104413 017012 WRITE ,WLNK1 ;WRITE ONE BLOCK
1332 003126 104421 REWIND ;GO TO "BOT"
1333 003130 104414 016756 READ ,RLNK1 ;GO TO "BOT"
1334 003134 104423 BLKCMP ;READ
1335 003136 016756 RLNK1 ;COMPARE THE READ AND WRITE BUFFERS
1336 003140 017012 WLNK1 ;LINK TO THE READ BUFFER
1337 003142 104416 BSBG ;LINK TO THE WRITE BUFFER
1338 003144 104414 016756 BSBG ;BACK UP
1339 003150 104423 BLKCMP ;READ
1340 003152 016756 RLNK1 ;COMPARE THE READ AND WRITE BUFFERS
;LINK TO THE READ BUFFER

```

```

1341 003154 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1342 ;*****
1343 ;*TEST 5 TEST BSBG AFTER READ
1344 ;*****
1345 003156 000004 TST5: SCOPE
1346 003160 012737 003226 001172 MOV #TST6,$ESCAPE ;;ESCAPE TO TEST 6 ON ERROR
1347 003166 104422 SELDRV ;SELECT DRIVE FOR TESTING
1348 003170 104421 REWIND ;GO TO "BOT"
1349 003172 104413 017012 WRITE ,WLNK1 ;WRITE A BLOCK
1350 003176 104421 REWIND ;GO TO "BOT"
1351 003200 104414 016756 READ ,RLNK1 ;READ A BLOCK
1352 003204 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1353 003206 016756 RLNK1 ;LINK TO THE READ BUFFER
1354 003210 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1355 003212 104415 BSBG ;BACK OVER THE DATA
1356 003214 104414 016756 READ ,RLNK1 ;NOW READ THE DATA
1357 003220 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1358 003222 016756 RLNK1 ;LINK TO THE READ BUFFER
1359 003224 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1360 ;*****
1361 ;*TEST 6 TEST SFBG AFTER REWIND AND BSBG AFTER SFBG
1362 ;*****
1363 003226 000004 TST6: SCOPE
1364 003230 012737 003266 001172 MOV #TST7,$ESCAPE ;;ESCAPE TO TEST 7 ON ERROR
1365 003236 104422 SELDRV ;SELECT DRIVE FOR TESTING
1366 003240 104421 REWIND ;GO TO "BOT"
1367 003242 104413 017012 WRITE ,WLNK1 ;WRITE ONE BLOCK OF DATA
1368 003246 104421 REWIND ;GO TO "BOT"
1369 003250 104420 SFBG ;SPACE OVER THE DATA
1370 003252 104415 BSBG ;BACK OVER THE DATA
1371 003254 104414 016756 READ ,RLNK1 ;READ THE BLOCK
1372 003260 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1373 003262 016756 RLNK1 ;LINK TO THE READ BUFFER
1374 003264 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1375 ;*****
1376 ;*TEST 7 TEST WRITE AFTER READ
1377 ;*****
1378 003266 000004 TST7: SCOPE
1379 003270 012737 003342 001172 MOV #TST10,$ESCAPE ;;ESCAPE TO TEST 10 ON ERROR
1380 003276 104422 SELDRV ;SELECT DRIVE FOR TESTING
1381 003300 104421 REWIND ;GO TO "CLEAR LEADER"
1382 003302 104413 017012 WRITE ,WLNK1 ;WRITE ONE BLOCK
1383 003306 104416 BSBG ;BACK OVER THE DATA BLOCK
1384 003310 104414 016756 READ ,RLNK1 ;AND READ IT
1385 003314 104423 BLKCMF ;COMPARE THE READ AND WRITE BUFFERS
1386 003316 016756 RLNK1 ;LINK TO THE READ BUFFER
1387 003320 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1388 003322 104413 017016 WRITE ,WLNK2 ;WRITE A SECOND BLOCK
1389 003326 104416 BSBG ;BACK OVER IT
1390 003330 104414 016762 READ ,RLNK2 ;AND READ IT
1391 003334 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1392 003336 016762 RLNK2 ;LINK TO THE READ BUFFER
1393 003340 017016 WLNK2 ;LINK TO THE WRITE BUFFER
1394 ;*****
1395 ;*TEST 10 TEST WRITE AFTER WRITE AND READ AFTER READ
1396 ;*****

```

```

1397 003342 000004 TST10: SCOPE
1398 003344 012737 003406 001172 MOV #TST11,$ESCAPE ;;ESCAPE TO TEST 11 ON ERROR
1399 003352 104422 SELDRV ;SELECT DRIVE FOR TESTING
1400 003354 104421 REWIND ;GO TO "CLEAR LEADER"
1401 003356 104413 017012 WRITE ,WLNK1 ;WRITE TWO(2) BLOCKS
1402 003362 104413 017016 WRITE ,WLNK2
1403 003366 104421 REWIND ;GO TO BOT
1404 003370 104414 016756 READ ,RLNK1 ;READ BOTH BLOCKS
1405 003374 104414 016762 READ ,RLNK2
1406 003400 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1407 003402 016762 RLNK2 ;LINK TO THE READ BUFFER
1408 003404 017016 WLNK2 ;LINK TO THE WRITE BUFFER
1409
1410 ;;*****
1411 ;*TEST 11 TEST BSBG AFTER BSBG
1412 TST11: SCOPE
1413 003410 012737 003450 001172 MOV #TST12,$ESCAPE ;;ESCAPE TO TEST 12 ON ERROR
1414 003416 104422 SELDRV ;SELECT DRIVE FOR TESTING
1415 003420 104421 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 TST12: SCOPE
1428 003452 012737 003512 001172 MOV #TST13,$ESCAPE ;;ESCAPE TO TEST 13 ON ERROR
1429 003460 104422 SELDRV ;SELECT DRIVE FOR TESTING
1430 003462 104421 REWIND ;GO TO "CLEAR LEADER"
1431 003464 104413 017012 WRITE ,WLNK1 ;WRITE TWO(2) BLOCKS
1432 003470 104413 017016 WRITE ,WLNK2
1433 003474 104421 REWIND ;BOT
1434 003476 104420 SFBG ;SPACE OVER THE 1ST BLOCK
1435 003500 104414 016762 READ ,RLNK2 ;READ THE 2ND BLOCK
1436 003504 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1437 003506 016762 RLNK2 ;LINK TO THE READ BUFFER
1438 003510 017016 WLNK2 ;LINK TO THE WRITE BUFFER
1439
1440 ;;*****
1441 ;*TEST 13 TEST SFBG AFTER BSBG
1442 TST13: SCOPE
1443 003514 012737 003556 001172 MOV #TST14,$ESCAPE ;;ESCAPE TO TEST 14 ON ERROR
1444 003522 104422 SELDRV ;SELECT DRIVE FOR TESTING
1445 003524 104421 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 ;;*****
1457 ;*TEST 14 TEST SFBG AFTER BSBG
1458 TST14: SCOPE
1459 003560 012737 003620 001172 MOV #TST15,$ESCAPE ;;ESCAPE TO TEST 15 ON ERROR
1460 003566 104422 SELDRV ;SELECT DRIVE FOR TESTING
1461 003570 104421 REWIND ;GO TO "CLEAR LEADER"
1462 003572 104413 017012 WRITE ,WLNK1 ;WRITE TWO BLOCKS
1463 003576 104413 017016 WRITE ,WLNK2
1464 003602 104415 BSBG ;SPACE REV FILE
1465 003604 104420 SFBG ;SPACE FWD BLOCK
1466 003606 104414 016762 READ ,RLNK2 ;READ THE 2ND BLOCK
1467 003612 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1468 003614 016762 RLNK2 ;LINK TO THE READ BUFFER
1469 003616 017016 WLNK2 ;LINK TO THE WRITE BUFFER
1470
1471 ;;*****
1472 ;*TEST 15 REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER WRITE
1473 TST15: SCOPE
1474 003620 000004 MOV #TST16,$ESCAPE ;;ESCAPE TO TEST 16 ON ERROR
1475 003622 012737 004012 001172 SELDRV ;SELECT DRIVE FOR TESTING
1476 003630 104422 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

```



TA11 MOTION TEST C2TAD-D MACY11 30A(1052) 23-MAR-78 08:36 PAGE 34 SEQ 0034  
C2TADD.P11 23-MAR-78 08:34 T15 REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER WRITE SEQ 0034

```

1509 003760 017016          ;LINK TO THE WRITE BUFFER
1510 003762 104416          ;BACK OVER BOTH BLOCKS
1511 003764 104416          ;AND READ THE 1ST ONE
1512 003766 104414 016756  ;COMPARE THE READ AND WRITE BUFFERS
1513 003772 104423          ;LINK TO THE READ BUFFER
1514 003774 016756          ;LINK TO THE WRITE BUFFER
1515 003776 017012          ;AND THE 2ND ONE
1516 004000 104414 016762  ;COMPARE THE READ AND WRITE BUFFERS
1517 004004 104423          ;LINK TO THE READ BUFFER
1518 004006 016762          ;LINK TO THE WRITE BUFFER
1519 004010 017016          ;LINK TO THE WRITE BUFFER
1520
1521 ;*****
1522 ;*TEST 16 REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER READ
1523 ;*****
1523 004012 000004          TST16: SCOPE
1524 004014 012737 004170 001172  MOV #TST17,$ESCAPE ;;ESCAPE TO TEST 17 ON ERROR
1525 004022 104422          SELDRV ;SELECT DRIVE FOR TESTING
1526 004024 104421          REWIND ;GO TO "CLEAR LEADER"
1527 004026 104413 017012  WRITE ,WLNK1 ;WRITE A BLOCK OF DATA
1528 004032 104416          BSBG ;BACK OVER IT
1529 004034 104414 016756  READ ,RLNK1 ;AND READ IT
1530 004040 104423          BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1531 004042 016756          RLNK1 ;LINK TO THE READ BUFFER
1532 004044 017012          RLNK1 ;LINK TO THE WRITE BUFFER
1533 004046 104413 017016  WRITE ,WLNK2 ;WRITE 2ND BLOCK
1534 004052 104416          BSBG ;BACK OVER IT
1535 004054 104414 016762  READ ,RLNK2 ;AND READ IT
1536 004060 104423          BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1537 004062 016762          RLNK2 ;LINK TO THE READ BUFFER
1538 004064 017016          WLNK2 ;LINK TO THE WRITE BUFFER
1539 004066 104416          BSBG ;BACK OVER IT AND
1540 004070 104413 017026  WRITE ,WLNK4 ;WRITE OVER 2ND BLOCK
1541 004074 104416          BSBG ;BACK UP AND
1542 004076 104414 016772  READ ,RLNK4 ;READ IT
1543 004102 104423          BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1544 004104 016772          RLNK4 ;LINK TO THE READ BUFFER
1545 004106 017026          WLNK4 ;LINK TO THE WRITE BUFFER
1546 004110 104416          BSBG ;BACK OVER IT AGAIN
1547 004112 104413 017032  WRITE ,WLNK5 ;REWRITE 2ND BLOCK
1548 004116 104416          BSBG
1549 004120 104414 016776  READ ,RLNK5 ;READ 2ND BLOCK
1550 004124 104423          BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1551 004126 016776          RLNK5 ;LINK TO THE READ BUFFER
1552 004130 017032          WLNK5 ;LINK TO THE WRITE BUFFER
1553 004132 104416          BSBG ;BACK OVER IT AGAIN
1554 004134 104413 017022  WRITE ,WLNK3 ;WRITE ANOTHER BLOCK OVER IT
1555 004140 104416          BSBG ;BACK OVER 2 BLOCKS
1556 004142 104416          BSBG
1557 004144 104414 016756  READ ,RLNK1 ;READ THE 1ST BLOCK
1558 004150 104423          BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1559 004152 016756          RLNK1 ;LINK TO THE READ BUFFER
1560 004154 017012          WLNK1 ;LINK TO THE WRITE BUFFER
1561 004156 104414 016766  READ ,RLNK3 ;READ 2ND BLOCK
1562 004162 104423          BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1563 004164 016766          RLNK3 ;LINK TO THE READ BUFFER
1564 004166 017022          WLNK3 ;LINK TO THE WRITE BUFFER

```

TA11 MOTION TEST C2TAD-D MACY11 30A(1052) 23-MAR-78 08:36 PAGE 35 SEQ 0035  
C2TADD.P11 23-MAR-78 08:34 T17 TEST WFG AFTER WRITE AND BSBG AFTER WFG SEQ 0035

```

1565 ;*****
1566 ;*TEST 17 TEST WFG AFTER WRITE AND BSBG AFTER WFG
1567 ;*****
1568 004170 000004          TST17: SCOPE
1569 004172 012737 004226 001172  MOV #TST20,$ESCAPE ;;ESCAPE TO TEST 20 ON ERROR
1570 004200 104422          SELDRV ;SELECT DRIVE FOR TESTING
1571 004202 104421          REWIND ;GO TO "BOT"
1572 004204 104413 017012  WRITE ,WLNK1 ;WRITE ONE BLOCK
1573 004210 104412          WFG ;FOLLOWED BY A FILE GAP
1574 004212 104415          BSBG ;BACK OVER THE DATA BLOCK
1575 004214 104414 016756  READ ,RLNK1 ;READ THE DATA
1576 004220 104423          BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1577 004222 016756          RLNK1 ;LINK TO THE READ BUFFER
1578 004224 017012          WLNK1 ;LINK TO THE WRITE BUFFER
1579
1580 ;*****
1581 ;*TEST 20 TEST BSBG AFTER WFG
1582 ;*****
1582 004226 000004          TST20: SCOPE
1583 004230 012737 004264 001172  MOV #TST21,$ESCAPE ;;ESCAPE TO TEST 21 ON ERROR
1584 004236 104422          SELDRV ;SELECT DRIVE FOR TESTING
1585 004240 104421          REWIND ;GO TO BOT
1586 004242 104413 017012  WRITE ,WLNK1 ;WRITE ONE BLOCK OF DATA
1587 004246 104412          WFG ;WRITE A FILE GAP
1588 004250 104416          BSBG ;BACK OVER THE DATA
1589 004252 104414 016756  READ ,RLNK1 ;READ ONE BLOCK
1590 004256 104423          BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1591 004260 016756          RLNK1 ;LINK TO THE READ BUFFER
1592 004262 017012          WLNK1 ;LINK TO THE WRITE BUFFER
1593
1594 ;*****
1595 ;*TEST 21 TEST WRITE AFTER WFG
1596 ;*****
1596 004264 000004          TST21: SCOPE
1597 004266 012737 004342 001172  MOV #TST22,$ESCAPE ;;ESCAPE TO TEST 22 ON ERROR
1598 004274 104422          SELDRV ;SELECT DRIVE FOR TESTING
1599 004276 104421          REWIND ;GO TO BOT
1600 004300 104413 017012  WRITE ,WLNK1 ;WRITE ONE BOCK
1601 004304 104421          REWIND ;GO TO BOT
1602 004306 104414 016756  READ ,RLNK1 ;READ ONE BLOCK
1603 004312 104423          BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1604 004314 016756          RLNK1 ;LINK TO THE READ BUFFER
1605 004316 017012          WLNK1 ;LINK TO THE WRITE BUFFER
1606 004320 104412          WFG ;WRITE A FILE GAP
1607 004322 104413 017016  WRITE ,WLNK2 ;WRITE ANOTHER BLOCK
1608 004326 104416          BSBG ;BACK OVER THE LAST BLOCK
1609 004330 104414 016762  READ ,RLNK2 ;READ THE SECOND BLOCK
1610 004334 104423          BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1611 004336 016762          RLNK2 ;LINK TO THE READ BUFFER
1612 004340 017016          WLNK2 ;LINK TO THE WRITE BUFFER
1613
1614 ;*****
1615 ;*TEST 22 TEST WFG AFTER READ AND BSBG AFTER WRITE
1616 ;*****
1616 004342 000004          TST22: SCOPE
1617 004344 012737 004420 001172  MOV #TST23,$ESCAPE ;;ESCAPE TO TEST 23 ON ERROR
1618 004352 104422          SELDRV ;SELECT DRIVE FOR TESTING
1619 004354 104421          REWIND ;GO TO "BOT"
1620 004356 104413 017012  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 004405 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 TST23: 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 TST24: 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 TST25: 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 004556 104413 017022 WRITE ,WLNK3

1677 004564 104421 REWIND ;GO TO "BOT"  
1678 004566 104417 SFFG ;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 ;\*TEST 26 TEST SFBG AFTER SFBG  
1687 ;\*\*\*\*\*  
1688 004606 000004 TST26: 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 SFFG ;SPACE OVER THE 1ST FILE  
1699 004650 104420 SFBG ;SPACE OVER TWO BLOCK  
1700 004652 104420 SFBG ;SPACE OVER TWO BLOCK  
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 ;\*TEST 27 TEST BSBG AFTER SFBG AND BSBG AFTER BSBG  
1707 ;\*\*\*\*\*  
1708 004666 000004 TST27: 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 SFFG ;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 ;\*TEST 30 TEST BSBG THRU A FILE GAP  
1726 ;\*\*\*\*\*  
1727 004740 000004 TST30: 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 ;WRITE TWO BLOCKS OF DATA
1735 004772 104421 REWIND ;GO TO "BOT"
1736 004774 104417 SFFG ;SPACE OVER 1ST FILE
1737 004776 104420 SFFG ;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 BSBG AFTER BSBG
1746 ;*****
1747 TST31: SCOPE
1748 005016 000004 MOV #TST32,$ESCAPE ;;ESCAPE TO TEST 32 ON ERROR
1749 005020 012737 005070 001172 SELDRV ;SELECT DRIVE FOR TESTING
1750 005026 104422 REWIND ;GO TO "BOT"
1751 005032 104413 017012 WRITE ,WLNK1 ;WRITE ONE BLOCK OF DATA
1752 005036 104412 WFG ;WRITE A FILE GAP
1753 005040 104413 017016 WRITE ,WLNK2 ;WRITE A BLOCK OF DATA
1754 005044 104421 REWIND ;GO TO "BOT"
1755 005046 104417 SFFG ;SPACE OVER FILE
1756 005050 104420 SFFG ;SPACE OVER 1ST BLOCK OF 2ND FILE
1757 005052 104416 BSBG ;BACK OVER BLOCK
1758 005054 104415 BSBG ;BACK OVER FILE
1759 005056 104414 016756 READ ,RLNK1 ;READ 1ST BLOCK ON TAPE
1760 005062 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1761 005064 016756 RLNK1 ;LINK TO THE READ BUFFER
1762 005066 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1763 ;*****
1764 ;*TEST 32 TEST BSBG AFTER BSBG
1765 ;*****
1766 TST32: SCOPE
1767 005070 000004 MOV #TST33,$ESCAPE ;;ESCAPE TO TEST 33 ON ERROR
1768 005072 012737 005134 001172 SELDRV ;SELECT DRIVE FOR TESTING
1769 005100 104422 REWIND ;GO TO "CLEAR LEADER"
1770 005102 104421 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 BSBG ;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 005132 017012 WLNK1 ;LINK TO THE WRITE BUFFER
1779 ;*****
1780 ;*TEST 33 TEST BSBG AFTER SFFG AND SFFG AFTER READ
1781 ;*****
1782 TST33: SCOPE
1783 005134 000004 MOV #TST34,$ESCAPE ;;ESCAPE TO TEST 34 ON ERROR
1784 005136 012737 005216 001172 SELDRV ;SELECT DRIVE FOR TESTING
1785 005144 104422 REWIND ;GO TO "BOT"
1786 005146 104421 017012 WRITE ,WLNK1 ;WRITE A BLOCK OF DATA
1787 005150 104413 WFG ;WRITE A FILE GAP
1788 005154 104412 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 BSBG ;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 TST34: SCOPE
1805 005216 000004 MOV #TST35,$ESCAPE ;;ESCAPE TO TEST 35 ON ERROR
1806 005220 012737 005274 001172 SELDRV ;SELECT DRIVE FOR TESTING
1807 005226 104422 REWIND ;GO TO BOT
1808 005230 104421 017012 WRITE ,WLNK1 ;WRITE A BLOCK
1809 005232 104413 WFG ;WRITE FILE GAP
1810 005236 104413 017016 WRITE ,WLNK2 ;WRITE A BLOCK
1811 005244 104412 WFG ;WRITE A FILE GAP
1812 005246 104413 017022 WRITE ,WLNK3 ;WRITE A BLOCK
1813 005252 104412 WFG ;WRITE A FILE GAP
1814 005254 104421 REWIND ;GO TO "BOT"
1815 005256 104417 SFFG ;SPACE OVER 1ST FILE
1816 005260 104417 SFFG ;SPACE OVER 2ND FILE
1817 005262 104414 016766 READ ,RLNK3 ;READ BLOCK OF 3RD FILE
1818 005266 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1819 005270 016766 RLNK3 ;LINK TO THE READ BUFFER
1820 005272 017022 WLNK3 ;LINK TO THE WRITE BUFFER
1821 ;*****
1822 ;*TEST 35 TEST SFFG AFTER SFFG
1823 ;*****
1824 TST35: SCOPE
1825 005274 000004 MOV #TST36,$ESCAPE ;;ESCAPE TO TEST 36 ON ERROR
1826 005276 012737 005350 001172 SELDRV ;SELECT DRIVE FOR TESTING
1827 005304 104422 REWIND ;GO TO "BOT"
1828 005306 104421 017012 WRITE ,WLNK1 ;WRITE A BLOCK
1829 005310 104413 017016 WRITE ,WLNK2 ;WRITE A BLOCK
1830 005314 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 SFFG ;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 2RD 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 SFFG
1842 ;*****
1843 TST36: SCOPE
1844 005350 000004 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 ;*****
1860 ;*TEST 37 TEST WRITE AFTER SFBG
1861 005422 000004 TST37: SCOPE
1862 005424 012737 005474 001172 MOV #TST40,$ESCAPE ;;ESCAPE TO TEST 40 ON ERROR
1863 005432 104422 SELDRV ;SELECT DRIVE FOR TESTING
1864 005434 104421 REWIND ;GO TO BOT
1865 005436 104413 017012 WRITE ,WLNK1 ;WRITE 2 BLOCK
1866 005442 104413 017016 WRITE ,WLNK2
1867 005446 104421 REWIND ;GO TO BOT
1868 005450 104420 SFBG ;SPACE OVER 1ST BLOCK
1869 005452 104413 017022 WRITE ,WLNK3 ;WRITE OVER 2ND BLOCK
1870 005456 104415 BSFG ;GO TO BEGINNING OF FILE
1871 005460 104420 WFG ;SPACE OVER 1ST BLOCK
1872 005462 104414 016766 READ ,RLNK3 ;READ THE NEW 2ND BLOCK
1873 005466 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1874 005470 016766 RLNK3 ;LINK TO THE READ BUFFER
1875 005472 017022 WLNK3 ;LINK TO THE WRITE BUFFER
1876
1877 ;*****
1878 ;*TEST 40 3 ONE BLOCK FILES---MULTI SPACING & TEST SFFG AFTER BSFG
1879 005474 000004 TST40: SCOPE
1880 005476 012737 005564 001172 MOV #TST41,$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 TST41: SCOPE
1905 005566 012737 005672 001172 MOV #TST42,$ESCAPE ;;ESCAPE TO TEST 42 ON ERROR
1906 005574 104422 SELDRV ;SELECT DRIVE FOR TESTING
1907 005576 104421 REWIND ;GO TO "BOT"
1908 005600 104413 017012 WRITE ,WLNK1 ;WRITE A 2 BLOCK FILE
1909 005604 104413 017016 WRITE ,WLNK2
1910 005610 104412 WFG
1911 005612 104413 017022 WRITE ,WLNK3 ;WRITE A 2 BLOCK FILE
1912 005616 104413 017026 WRITE ,WLNK4
1913 005622 104412 WFG
1914 005624 104413 017032 WRITE ,WLNK5 ;WRITE A 2 BLOCK FILE
1915 005630 104413 017036 WRITE ,WLNK6
1916 005634 104412 WFG
1917 005636 104421 REWIND ;GO TO "BOT"
1918 005640 104417 SFFG ;SPACE OVER 1ST FILE
1919 005642 104417 SFFG ;SPACE OVER 2ND FILE
1920 005644 104414 016776 READ ,RLNK5 ;READ 1ST BLOCK OF 3RD FILE
1921 005650 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1922 005652 016776 RLNK5 ;LINK TO THE READ BUFFER
1923 005654 017032 WLNK5 ;LINK TO THE WRITE BUFFER
1924 005656 104415 BSFG ;BACK TO BEGINNING OF 3RD FILE
1925 005660 104414 016778 READ ,RLNK5 ;READ 1ST BLOCK OF 3RD FILE
1926 005664 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1927 005666 016776 RLNK5 ;LINK TO THE READ BUFFER
1928 005670 017032 WLNK5 ;LINK TO THE WRITE BUFFER
1929
1930 ;*****
1931 ;*TEST 42 MULTI SPACING & TEST SFBG AFTER READ
1932 005672 000004 TST42: SCOPE
1933 005674 012737 006042 001172 MOV #TST43,$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
```



```

2056 .SBTTL END OF PASS ROUTINE
2057
2058 ;*****
2059 ;*INCREMENT THE PASS NUMBER ($PASS)
2060 ;*TYPE "END PASS"
2061 ;*IF THERES A MONITOR GO TO IT
2062 ;*IF THERE ISN'T JUMP TO START
2063 ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
2064 ;*$ENDMG CAN BE CHANGED TO 7.
2065
2066 006352 $EOP: SCOPE
2067 006352 000004 CLR STSTNM ;;ZERO THE TEST NUMBER
2068 006354 005037 001102 CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
2069 006360 005037 001170 INC $PASS ;;INCREMENT THE PASS NUMBER
2070 006364 005237 001100 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
2071 006370 042737 100000 001100 DEC (PC)+ ;;LODP?
2072 006376 005327 $EOPCT: .WORD 1
2073 006400 000001 BGT $D0AGN ;;YES
2074 006402 003017 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
2075 006404 012737 $ENDCT: .WORD 1
2076 006406 000001 $EOPCT
2077 006410 006400 TYPE , $ENDMG ;;TYPE "END PASS"
2078 006412 104401 006451 TYPE , $ENULL ;;TYPE A NULL CHARACTER
2079 006416 104401 006446 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
2080 006422 013700 000042 BEQ $D0AGN ;;BRANCH IF NO MONITOR
2081 006426 001405 RESET ;;CLEAR THE WORLD
2082 006430 000005 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
2083 006432 004710 NOP ;;SAVE ROOM
2084 006434 000240 NOP ;;FOR
2085 006436 000240 NOP ;;ACT11
2086 006440 000240 $D0AGN: JMP @(PC)+ ;;RETURN
2087 006442 $RTNAD: .WORD START
2088 006442 000137 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
2089 006444 002312 $ENDMG: .ASCIZ <15><12>/END PASS/
2090 006446 377 000
2091 006451 015 042412 042116
2092 006456 050040 051501 000123
    
```

```

2093 ;*****
2094 ;
2095 ;.SBTTL ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2096
2097 ;CALL:
2098 ; MOV #DRVKEY,R0
2099 ; JSR PC,@#EXAM ;R1 IS DESTROYED
2100 ; NORMAL RETURN
2101 ; ERROR RETURN
2102
2103 006464 013701 001220 EXAM: MOV @#TACSL,R1 ;PICKUP THE "CONTROL & STATUS" REG. ADR.
2104 006470 005011 CLR (R1) ;DRIVE="A", FUNCTION="WFG"
2105 006472 122710 000101 CMPB #'A,(R0) ;EXAMINE DRIVE "A"?
2106 006476 001402 BEQ 1$ ;BR IF YES
2107 006500 052711 000400 BIS #UNIT,(R1) ;SELECT DRIVE "B"
2108 006504 032711 000040 1$: BIT #READY,(R1) ;WAIT ON READY
2109 006510 001775 BEQ 1$
2110 006512 005711 TST (R1) ;ANY ERROR?
2111 006514 100024 BPL 4$ ;BR IF NO
2112 006516 032711 001000 BIT #OFFLINE,(R1) ;ERROR DUE TO "OFF LINE"?
2113 006522 001017 BNE 3$ ;BR IF YES
2114 006524 032711 010000 BIT #WRTLOCK,(R1) ;ERROR DUE TO "WRITE LOCK"?
2115 006530 001411 BEQ 2$ ;BR IF NO
2116 006532 122777 000201 172400 CMPB #BIT071BIT00,@SWR ;"READONLY" SELECTED? (RD1PAS)
2117 006540 001412 BEQ 4$ ;BR IF YES
2118 006542 122777 000203 172370 CMPB #BIT071BIT011BIT00,@SWR ;(RD2PAS)?
2119 006550 001406 BEQ 4$ ;BR IF YES
2120 006552 000403 BR 3$ ;TAKE THE ERROR EXIT
2121 006554 032711 020000 2$: BIT #LEADER,(R1) ;ERROR DUE TO "CLEAR LEADER"?
2122 006560 001002 BNE 4$ ;BR IF YES
2123 006562 062716 000002 3$: ADD #2,(SP) ;TAKE ERROR RETURN
2124 006566 000207 4$: RTS PC ;RETURN
    
```

```

.SBTTL SCOPE HANDLER ROUTINE
2125
2126
2127 ;*****
2128 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
2129 ;*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
2130 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
2131 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2132 ;*SW14=1 LOOP ON TEST
2133 ;*SW11=1 INHIBIT ITERATIONS
2134 ;*SW09=1 LOOP ON ERROR
2135 ;*CALL
2136 ;* SCOPE ;;SCOPE=IDT
2137
2138 $SCOPE:
2139 CKSWR @STKS ;;TEST FOR CHANGE IN SOFT-SWR
2140 TSTB BPL 100$ ;TTY FLAG UP?
2141 BPL 100$ ;BR IF NO
2142 RDCMR @#177060 ;GO GET ONE CHARACTER FROM TTY
2143 CMPB (SP)+,#3 ;IS IT A CONTROL/C ?
2144 BNE 100$ ;BR IF NO
2145 SETLOOP (PC)+ ;GO GET "TEST" AND "SCOPE LOOP" ADDRESSES
2146 100$: TST ;LOOP ON TEST?
2147 LOOPKEY: 0 ;BR IF YES
2148 BNE $OVER ;BR IF YES
2149 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
2150 BNE $OVER ;;YES IF SW14=1
2151 ;*****START OF CODE FOR THE XOR TESTER*****
2152 $XTSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
2153 ;THIS INSTRUCTION TO A "NOP" (NOP=240)
2154 MOV @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
2155 MOV #5$,@#ERRVEC ;;SET FOR TIMEOUT
2156 TST @#177060 ;;TIME OUT ON XOR?
2157 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
2158 BR $SVLAD ;;GO TO THE NEXT TEST
2159 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
2160 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
2161 BR 7$ ;;LOOP ON THE PRESENT TEST
2162 6$:*****END OF CODE FOR THE XOR TESTER*****
2163 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
2164 BEQ 3$ ;;BR IF NO
2165 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
2166 BHI 3$ ;;BR IF NO
2167 BIT #BIT09,@SWR ;;LOOP ON ERROR?
2168 BEQ 4$ ;;BR IF NO
2169 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
2170 BR $OVER
2171 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
2172 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
2173 BR 1$ ;;ESCAPE TO THE NEXT TEST
2174 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
2175 BNE 1$ ;;BR IF YES
2176 TST $PASS ;;IF FIRST PASS OF PROGRAM
2177 BEQ 1$ ;; INHIBIT ITERATIONS
2178 INC $ICNT ;;INCREMENT ITERATION COUNT
2179 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
2180 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
    
```

```

2181 006770 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
2182 006776 013737 001216 001170 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
2183 007004 105237 001102 $SVLAD: INCB $STNM ;;COUNT TEST NUMBERS
2184 007010 011637 001106 MOV (SP),$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 $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
2189 007040 013716 001106 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
2190 007044 000002 RTI ;;FIXES PS
    
```

```

2191 .SBTTL ERROR HANDLER ROUTINE
2192
2193 ;*****
2194 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2195 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2196 ;*AND GO TO TYPERR ON ERROR
2197 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2198 ;*SW15=1 HALT ON ERROR
2199 ;*SW13=1 INHIBIT ERROR TYPEDOUTS
2200 ;*SW10=1 BELL ON ERROR
2201 ;*SW09=1 LOOP ON ERROR
2202 ;*CALL
2203 ;*
2204 ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2205
2206 $ERROR:
2207 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
2208 MOV @TACS,@#$REG0 ;;SAVE THE TA11 STATUS
2209 MOV @TADB,@#$REG1 ;;SAVE THE DATA BUFFER
2210 TSTB @TACS ;;IF "XFER REQ" IS UP KNOCK IT DOWN
2211 BPL 100$
2212 MOV #ILBS,@TACS
2213 BR 101$
2214 BIT #READY,@TACS ;;IS READY SET?
2215 BNE 101$ ;;BR IF YES
2216 RESET ;;REGAIN CONTROL
2217 MOV @#$REG0,@TACS ;;RESTORE THE TACS
2218 CMP (SP)+,(SP)+ ;;POP THE STACK
2219 TST ASKKEY ;;MANUAL LOOPING?
2220 BGE 102$ ;;BR IF NO
2221 MOV (SP),$ESCAPE ;;SETUP FOR MANUAL ESCAPE
2222 INCB $ERFLG ;;SET THE ERROR FLAG
2223 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
2224 MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
2225 BIT #BIT:0,@SWR ;;BELL ON ERROR?
2226 BEQ 1$ ;;NO - SKIP
2227 TYPE $BELL ;;RING BELL
2228 INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
2229 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
2230 SUB #2,$ERRPC
2231 MOV @SERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
2232 BIT #BIT13,@SWR ;;SKIP TYPEDOUT IF SET
2233 BNE 20$ ;;SKIP TYPEDOUTS
2234 JSR PC,TYPERR ;;GO TO USER ERROR ROUTINE
2235 TYPE $CRLF
2236
2237 20$: TST @SWR ;;HALT ON ERROR
2238 BPL 3$ ;;SKIP IF CONTINUE
2239 HALT ;;HALT ON ERROR!
2240 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
2241 BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
2242 BEQ 4$ ;;BR IF NO
2243 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
2244 TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
2245 BEQ 5$ ;;BR IF NONE
2246 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
    
```

```

2247 5$: CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
2248 BNE 6$ ;;BRANCH IF NO
2249 HALT ;;YES
2250
2251 6$: RTI ;;RETURN
2252
    
```



```

2253 ;*****
2254 ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
2255 007276 104401 001201 TYPERR: TYPE ,$CRLF ;TYPE A CARRIAGE RETURN & LINE FEED
2256 007302 104401 TYPE ;TYPE THE FUNCTION BEING PERFORMED
2257 007304 000000 MFUNC: 0 ;MESSAGE POINTER GOES HERE
2258 007306 104401 001201 TYPE , $CRLF
2259 007312 162737 000002 001204 SUB #2, @#SAVPC ;ADJUST THE MAIN FLOW PC
2260 007320 010046 MOV RO, -(SP) ;SAVE RO
2261 007322 113700 MOVB @#$ITEMB, RO ;PICKUP THE ITEM INDEX
2262 007326 005300 DEC RO ;ADJUST THE INDEX
2263 007330 006300 ASL RO ;SO IT WILL WORK FOR
2264 007332 006300 ASL RO ;THE ERROR TABLE
2265 007334 006300 ASL RO
2266 007336 062700 ADD # $ERRTB, RO ;FORM THE TABLE POINTER
2267 007342 012037 007350 MOV (RO)+, 1$ ;PICKUP "ERROR MESSAGE" POINTER
2268 007346 104401 TYPE ;TYPE "ERROR MESSAGE"
2269 007350 000000 1$: 0 ;"ERROR MESSAGE POINTER" GOES HERE
2270 007352 012037 007362 MOV (RO)+, 2$ ;PICKUP "DATA HEADER" POINTER
2271 007356 001402 BEQ 3$ ;IF "0" DON'T TYPE
2272 007360 104401 TYPE ;TYPE "DATA HEADER"
2273 007362 000000 2$: 0 ;"DATA HEADER" POINTER GOES HERE
2274 007364 012000 3$: MOV (RO)+, RO ;PICKUP "DATA POINTER"
2275 007366 001004 BNE 5$ ;IF THERE IS DATA TO TYPE GO DO IT
2276 007370 012600 4$: MOV (SP)+, RO ;RESTORE RO
2277 007372 104401 001201 TYPE , $CRLF ;TYPE A CARRIAGE RETURN&LINE FEED
2278 007376 000207 RTS PC ;RETURN
2279 007400
2280 007400 013046 5$: MOV @ (RO)+, -(SP) ;SAVE @ (RO)+ FOR TYPEOUT
2281 ;TYPE DATA
2282 007402 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2283 007404 005710 TST (RO) ;TERMINATOR?
2284 007406 001770 BEQ 4$ ;BR IF YES
2285 007410 104401 007416 TYPE , 6$ ;TYPE 2 SPACES.
2286 007414 000771 BR 5$ ;LOOP
2287 007416 020040 000 6$: .ASCIZ / / ;ASCII STRING OF 2 SPACES
2288 007422 .EVEN
    
```

```

2289 ;*****
2290 ;SBTTL ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2291
2292 ;CALL
2293 ; JSR PC, @#ASKDRV ;NOTE: RO AND R1 ARE DESTROYED
2294 ;
2295 ; RETURN
2296
2297 007422 104401 015136 ASKDRV: TYPE ,MSGDRV ;<CRLF>"DRIVE(S)? "
2298 007426 005037 001240 CLR DRVKEY ;GO GET A DRIVE
2299 007432 104410 RDLIN ;SETUP TO CHECK FOR VALID DRIVE(S)
2300 007434 012600 MOV (SP)+, RO ;WAS A DRIVE SELECTED?
2301 007436 105710 TSTB @RO ;BR IF NO
2302 007440 001425 BEQ NOTLGL
2303 007442 012701 001240 MOV #DRVKEY, R1 ;WAS DRIVE "A" SELECTED?
2304 007446 122710 000101 LOOP: CMPB #'A, @RO ;BR IF NO
2305 007452 001002 BNE NDTA ;SET KEY FOR DRIVE "A"
2306 007454 112021 BR NEXT
2307 007456 000411 NOTA: CMPB #'B, @RO ;WAS DRIVE "B" SELECTED?
2308 007460 122710 000102 BNE NOTB ;BR IF NO
2309 007464 001002 MOVB (RO)+, (R1)+ ;SET KEY FOR DRIVE "B"
2310 007466 112021 BR NEXT
2311 007470 000404 NOTB: CMPB #'54, @RO ;WAS A COMMA TYPED?
2312 007472 122710 000054 BNE NOTLGL ;BR IF NO
2313 007476 001006 TSTB (RO)+ ;DUMP THE COMMA
2314 007500 105720 NEXT: TSTB @RO ;TERMINATOR?
2315 007502 105710 BEQ EXIT ;BR IF YES
2316 007504 001406 CMP #DRVKEY+2, R1 ;TWO DRIVES SELECTED?
2317 007506 022701 001242 BHI LOOP ;BR IF NO
2318 007512 101355 NOTLGL: TYPE , $QUES ;ILLEGAL INPUT DETECTED
2319 007514 104401 001200 BR ASKDRV ;GO TRY AGAIN
2320 007520 000740 EXIT: TST DRVKEY ;ANY DRIVE SELECTED?
2321 007522 005737 001240 BEQ NOTLGL ;BR IF NO
2322 007526 001772 RTS PC
2323 007530 000207
2324 ;*****
2325 ;CALL
2326 ; JSR PC, @#ASKADR
2327 ;
2328 ; RETURN
2329 007532 010046 ASKADR: MOV RO, -(SP) ;SAVE RO
2330 007534 104401 015152 1$: TYPE ,MSGASK ;"TACS?"
2331 007540 104411 RDOCT ;GET VALUE
2332 007542 012600 MOV (SP)+, RO ;PICK UP THE OCTAL NUMB'R
2333 007544 001423 BEQ 3$ ;IF "0" USE OLD VALUES
2334 007546 020027 160000 CMP RO, #160000 ;MAKE SURE IT IS A BUS ADDRESS
2335 007552 103770 BLD 1$
2336 007554 010037 001220 MOV RO, @#TACSL ;SAVE TOE TACS
2337 007560 062700 000002 ADD #2, RO ;STEP TO TADB ADDRESS
2338 007564 010037 001224 MOV RO, @#TADBL ;AND SAVE IT
2339 007570 013737 001220 001222 MOV @#TACSL, @#TACSH ;SET UP TACS UPPER
2340 007576 005237 001222 INC @#TACSH ;BYTE POINTER
2341
2342 007602 013737 001224 001226 MOV @#TADBL, @#TADBH ;SET UP TADB UPPER
2343 007610 005237 001226 INC @#TADBH ;BYTE POINTER
2344 007614 104401 015162 3$: TYPE ,MSGVEC ;"VECTOR?"
    
```

```

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$: TYPE ,MSGPRI ;ASK FOR PRIORITY
2354 007654 104411 RDOCT
2355 007656 012600 MOV (SP)+,RO
2356 007660 001413 6$ 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 #'C<340>,RO ;SAVE ONLY PRIORITY BITS
2364 007704 010037 001234 MOV RO,@#TAPRID ;STORE IT AWAY
2365 007710 104401 015204 6$: TYPE ,MTACS ;TACS=
2366 007714 013746 001220 MOV TACSL,-(SP) ;;SAVE TACSL FOR TYPEOUT
2367 007720 104402 TYPDC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2368 007722 104401 015213 TYPE ,MTADB ;"TADB="
2369 007726 013746 091224 MOV TADBL,-(SP) ;;SAVE TADBL FOR TYPEOUT
2370 007732 104402 TYPDC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2371 007734 104401 015222 TYPE ,MTAVEC ;"VECTOR="
2372 007740 013746 001230 MOV TAVEC,-(SP) ;;SAVE TAVEC FOR TYPEOUT
2373 007744 104402 TYPDC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2374 007746 104401 015233 TYPE ,MTAPRI ;"PRIORITY="
2375 007752 013746 001234 MOV TAPRID,-(SP) ;;SAVE TAPRID FOR TYPEOUT
2376 007756 104401 015246 TYPDC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2377 007760 104401 TYPE ,MSGOK ;"OK?"
2378 007764 104407 RDCHR (SP)-,RO ;GO READ ONE CHARACTER
2379 007766 012600 MOV (SP)-,RO ;GET IT
2380 007770 022700 000015 CMP #15,RO ;IS IT "CR"?
2381 007774 001406 BEQ 7$ ;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 ,SQUES ;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
2389
2390
2391 ;*****
2392 ;THIS ROUTINE WILL CHANGE AN .ASCII STRING TO AN OCTAL NUMBER
2393 ;
2394 ;REGISTER R1 MUST BE INITIALIZED BEFORE ENTERING
2395 ;
2396 ;CALL:
2397 JSR R3,@#A2OCT ;CALL THE ROUTINE
2398 ; POINTER ;ADDRESS OF THE FIRST CHARACTER IN THE STRING
2399 ; ;STRING MUST BE TERMINATED BY A BYTE OF ALL 0'S
2400 ; ;ERROR RETURN ;RETURN HERE IF ILLEGAL CHARACTER
    
```

```

2401 ;
2402 ; NORMAL RETURN ;OR STRING IS TO LONG
2403 ; ;OCTAL NUMBER IS IN R1
2404 010022 010046 A2OCT: MOV RO,-(SP) ;SAVE RO
2405 010024 012300 MOV (R3)+,RO ;PICK UP THE POINTER
2406 010026 012737 000006 010042 MOV #6,2$ ;MAX. NUMBER OF CHAR.'S ALLOWED
2407 010034 105710 1$: TSTB (RO) ;TERMINATOR?
2408 010036 001420 BEQ 3$ ;BR IF YES
2409 010040 005327 DEC (PC)+ ;COUNT THIS CHARACTER
2410 010042 000000 2$: 0
2411 010044 002416 BLT 4$ ;BR IF TO MANY CHARACTERS IN THE STRING
2412 010046 121027 000060 CMPB (RO),#'0 ;CHECK THIS CHAR. IS BETWEEN
2413 010052 002413 BLT 4$ ;0 AND 7
2414 010054 121027 000067 CMPB (RO),#'7
2415 010060 003010 BGT 4$
2416 010062 142710 177770 BICB #'C7,(RO) ;STRIP AWAY THE ASCII CODE
2417 010066 006301 ASL R1 ;POSITION THE FOR THIS CHAR
2418 010070 006301 ASL R1
2419 010072 006301 ASL R1
2420 010074 152001 BISB (RO)+,R1 ;SET IN THIS DIGIT
2421 010076 000756 BR 1$ ;GO GET THE NEXT CHARACTER
2422 010100 005723 3$: TST (R3)+ ;INCREMENT FOR NORMAL RETURN
2423 010102 012600 4$: MOV (SP)+,RO ;RESTORE RO
2424 010104 000203 RTS R3 ;RETURN TO USER
    
```

```

2425 ;*****
2426 ;THIS ROUTINE WILL INPUT THE ADDRESSES OF "TEST" AND "SCOPE LOOP" FROM THE TTY
2427
2428 T.SETLOOP:
2429 TYPE ,MTEST ;TYPE "TEST PC?"
2430 RDLIN ;GET AN ASCII STRING
2431 MOV (SP)+,11$ ;GET FIRST ADDRESS OF INPUT BUFFER
2432 TSTB @11$ ;FIRST CHAR. A "CR" ?
2433 BEQ 3$ ;BR IF YES---GO CLEAR THE LOOP KEY
2434 CMP (SP)+,(SP)+ ;POP POP
2435 MOV R1,SAVR1 ;SAVE R1
2436 CLR R1 ;SET PARTIAL TO ZERO
2437 JSR R3,@#A2OCT ;GO CHANGE ASCII TO OCTAL
2438 11$: 0 ;POINTER TO STRING
2439 BR T.SETLOOP ;ERROR RETURN---GO TRY AGAIN
2440 TST (R1)+ ;JUST INCREMENTING
2441 MOV R1,(SP) ;PUT "TEST PC" ON THE STACK
2442 TYPE ,MLOOP ;TYPE "LOOP PC ?"
2443 RDLIN ;GET A STRING OF ASCII
2444 MOV (SP)+,12$ ;GET FIRST ADDRESS OF INPUT BUFFER
2445 TSTB @12$ ;JUST A "CR" ?
2446 BEQ 1$ ;BR IF YES
2447 CLR R1 ;CLEAR PARTIAL
2448 JSR R3,@#A2OCT ;CHANGE TO OCTAL
2449 12$: 0
2450 BR 5$
2451 MOV R1,@#LPADR ;SAVE SCOPE LOOP ADDRESS
2452 BR 2$ ;GO SET LOOP KEY
2453 MOV (SP),@#LPADR ;SAVE SCOPE LOOP ADDRESS
2454 MOV (SP),@#LPERR ;ERROR LOOP SAME AS THE TEST ADDRESS
2455 MOV #-1,@#LOOPKEY ;SET THE LOOP KEY
2456 BR 4$
2457 CLR @#LOOPKEY ;CLEAR THE LOOP KEY
2458 MOV SAVR1,R1 ;RESTORE R1
2459 RTI ;BYE BYE
2460 SAVR1: 0 ;HOLD R1 HERE
    
```

```

2461 ;*****
2462 ;THIS ROUTINE WILL SELECT THE NEXT DRIVE TO TEST
2463
2464 T.SELDRV:
2465 BIT #SW07,@SWR ;LOCK ON CURRENT DRIVE?
2466 BNE 5$ ;BR IF YES
2467 MOV R1,-(SP) ;SAVE R1 ON THE STACK
2468 MOV DRIVE,R1 ;GET THE DRIVE
2469 MOVB (R1),@#CURDRV ;SAVE CURRENT DRIVE
2470 CMPB (R1),#'A ;IS IT DRIVE "A"?
2471 BEQ 1$ ;BR IF YES
2472 BIS #UNIT,@TACS ;SELECT DRIVE "B"
2473 MOV #RWDB,@#RWDFLG ;SET REWIND POINTER TO "B"
2474 BR 2$
2475 BIC #UNIT,@TACS ;SELECT DRIVE "A"
2476 MOV #RWDA,@#RWDFLG ;SET REWIND POINTER TO "A"
2477 INC R1 ;INCREMENT TO THE NEXT DRIVE
2478 TSTB (R1) ;OUT OF DRIVES TO TEST?
2479 BNE 3$ ;BR IF NO
2480 MOV #DRVKEY,R1 ;RESET TO THE FIRST DRIVE
2481 MOV R1,DRIVE ;SAVE FOR NEXT TIME
2482 BIT #SW09,@SWR ;IS SWITCH 9=1?
2483 BNE 4$ ;BR IF YES
2484 CLRB @#SERFLG ;CLEAR THE ERROR INDICATOR
2485 MOV (SP)+,R1 ;RESTORE R1
2486 RTI
    
```

```

2488 ;:*****
2489 ;THIS ROUTINE WILL COMPARE THE READ AND WRITE BUFFERS
2490 T.BLKCMP:
2491 MOV (SP),RO ;GET THE PC OF THE CALL
2492 MOV RO,@$SAVPC ;SAVE THE PC OF THE CALL
2493 MOV #MBUFCK,@$MFUNC ;POINTER FOR .ASCII MESSAGE
2494 MOV (R0)+,R3 ;PICKUP THE READ LINK ADDRESS
2495 MOV (R3)+,R1 ;PICKUP THE READ BUFFER ADDRESS
2496 MOV (R3),R3 ;PICKUP THE READ BUFFER SIZE
2497 MOV @ (R0)+,R2 ;PICKUP THE WRITE BUFFER ADDRFS5
2498 CLR 2$+2 ;ZERO THE OFFSET
2499 MOV RO,(SP) ;SET THE RETURN
2500 CLR @$BYTNUM ;ZERO THE BYTE NUMBER
2501 BIT #SW03,@$SWR ;COMPARE DATA BUFFERS?
2502 BEQ 1$ ;BR IF YES
2503 RTI ;NO---GO BACK
2504 1$: DEC R3 ;CHECK FOR THE TERMINATION
2505 BGE 4$ ;BR IF MORE BYTES TO COMPARE
2506 RTI ;RETURN
2507 4$: INC @$BYTNUM ;COUNT THIS BYTE
2508 2$: CMPB (R1)+,0(R2) ;COMPARE READ AND WRITE BUFFER
2509 BNE 3$ ;BR IF NOT EQUAL
2510 INC 2$+2 ;INCREMENT WRITE BUFFER INDEX
2511 BIC #'C3,2$+2 ;DON'T LET IT GET BIGGER THAN 3
2512 BR 1$
2513 3$: MOVB -(R1),@$SDDAT ;GET THE RECEIVED BYTE
2514 ADD 2$+2,R2 ;FORM ADDRESS
2515 MOVB (R2),@$GDDAT ;GET THE EXPECTED BYTE
2516 ERROR 10 ;RECEIVED AND EXPECTED DATA NOT EQUAL
2517 ;:*****
2518 ;THIS ROUTINE WILL WRITE A FILE GAP ON THE TAPE
2519 T.WFG: MOV (SP),@$SAVPC ;SAVE THE PC OF THE CALL
2520 MOV #MXWFG,@$MFUNC ;SET MESSAGE POINTER FOR "WFG"
2521 1$: JSR RO,@$DO.CMD ;GO START A "WFG"
2522 XWFGIINT.EN!GO
2523 CLR @$PS ;ALLOW INTERRUPTS
2524 JSR RO,@$WAITFLAG ;GO WAIT FOR A FLAG
2525 ERROR 3 ;NO FLAGS OCCURRED
2526 ERROR 4 ;NO INTERRUPT
2527 JSR RO,@$FLAGS ;NORMAL RETURN---FIND OUT WHAT FLAG
2528 ERROR 2 ;RETURN HERE IF "XFER REQ" AND "READY" =0
2529 ERROR 2 ;RETURN HERE IF "XFER REQ" =1
2530 RTI ;RETURN HERE IF "READY"=1 AND "ERROR" =0
2531 ERROR 2 ;"READY"=1 AND "ERROR"=1
    
```

```

2532 ;:*****
2533 ;THIS ROUTINE WILL WRITE A RECORD ON THE TAPE
2534 T.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 2$: JSR RO,@$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$: BISB 0(R1),RO ;PICKUP BYTE FOR TRANSFER
2561 MOVB RO,@TADB ;TRANSFER A BYTE TO THE TA11
2562 JSR PC,@$DO.CRC ;CALCULATE CRC FOR THIS BYTE
2563 INC 6$+2 ;INCREASE THE INDEX
2564 BIC #'C3,6$+2 ;KEEP IT LESS THAN 4
2565 MOV #SERV,@TAVEC ;SETUP TO SERVICE THE INTERRUPT
2566 BR 3$ ;LOOP
2567 5$: MOV @CRC.WD,4(R1) ;SAVE THE CRC WORD
2568 JSR RO,@$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"
    
```

```

2579          ;:*****
2580          ;THIS ROUTINE WILL READ A RECORD FROM THE TAPE
2581 010774 011637 001204          T.READ: MOV (SP),@#SAVPC          ;SAVE THE PC OF THE CALL
2582 011000 012737 015361 007304 MOV #MXREAD,@#MFUNC          ;SET POINTER OF .ASCIZ MESSAGE
2583 011006 017602 000000          MOV @ (SP),R2          ;GET THE READ LINK POINTER
2584 011012 062716 000002          ADD #2,(SP)          ;ADJUST FOR RETURN
2585 011016 010200          MOV R2,R0          ;PICKUP AND SAVE THE "CRC" THAT
2586 011020 016000 000034          MOV WLNK1-RLNK1(R0),R0          ; WAS CALCULATED ON THE WRITE
2587 011024 016037 000004 001210 MOV 4(R0),RCRC0          ; DATA FOR THIS BLOCK
2588 011032 012201          MOV (R2)+,R1          ;GET ADDRESS OF BUFFER
2589 011034 012202          MOV (R2)+,R2          ;GET NUMBER OF BYTES
2590 011036 003001          BGT 2$          ;MAKE SURE NUMBER IS LEGAL
2591 011040 104011          ERROR 11          ;ILLEGAL NUMBER OF BYTES
2592 011042 005037 012222          2$: CLR @#CRC.WD          ;CLEAR CRC WORD
2593 011046 004037 012224          JSR R0,@#DD.CMD          ;GO START A "READ"
2594 011052 000105          XREADIINT.ENIG0
2595 011054 005037 177776          3$: CLR @#PS          ;ALLOW INTERRUPTS
2596 011060 105011          CLR (R1)          ;SET TERMINATOR IF LAST BYTE
2597 011062 004037 011716          JSR R0,@#WAITFLAG          ;WAIT FOR A FLAG
2598 011066 104003          ERROR 3          ;NONE OCCURRED
2599 011070 104004          ERROR 4          ;NO INTERRUPT
2600 011072 005037 001254          CLR @#WAITKEY          ;CLEAR THE WAIT ON INTERRUPT KEY
2601 011076 012737 000340 177776 MOV #340,@#PS          ;LOCK OUT ALL INTERRUPTS
2602 011104 004037 012014          JSR R0,@#FLAGS          ;WHAT FLAG OCCURRED
2603 011110 104002          ERROR 2          ;UNKNOWN FLAG
2604 011112 000402          BR 4$          ;"XFER REQ"
2605 011114 104005          ERROR 5          ;"READY"
2606 011116 000413          BR 5$          ;"READY" WITH "ERROR"
2607 011120 005302          4$: DEC R2          ;READ A BYTE?
2608 011122 002431          BLT 5$          ;BR IF NO
2609 011124 005000          CLR R0
2610 011126 151500          BISB @TADB,R0          ;READ ONE BYTE
2611 011130 110021          MOV (R1)+          ;AND PUT IT IN THE BUFFER
2612 011132 004737 012102          JSR PC,@#DD.CRC          ;CALCULATE CRC FOR THIS BYTE
2613 011136 012777 012340 170064 MOV #SERV,@TAVEC          ;SETUP TO SERVICE THE INTERRUPT
2614 011144 000743          BR 8$          ;LOOP
2615 011146 010237 001166          8$: MOV R2,@#STMP0          ;GET NUMBER OF BYTES NOT READ
2616 011152 001414          BEQ 9$          ;SKIP OUT IF 0
2617 011154 013746 001172          MOV @#SESCAPE,-(SP)          ;SAVE THE ESCAPE ADDRESS
2618 011160 005037 001172          CLR @#SESCAPE          ;NO ESCAPE DESIRED
2619 011164 016646 000004          MOV 4(SP),-(SP)          ;COPY THE STACK
2620 011170 016646 000004          MOV 4(SP),-(SP)
2621 011174 104013          ERROR 13          ;BLOCK READ SHORT
2622 011176 012637 001172          MOV (SP)+,@#SESCAPE          ;RESTORE ESCAPE ADDRESS
2623 011202 000002          RTI          ;LEAVE
2624 011204 104002          ERROR 2          ;ERROR OCCURRED DURING READ
2625 011206 013737 012222 001212 5$: MOV @#CRC.WD,@#RCRC1          ;SAVE THE CALCULATED CRC
2626 011214 004037 012224          JSR R0,@#DD.CMD          ;DO AN ILBS
2627 011220 000124          XREADIINT.ENIILBS
2628 011222 111537 001214          MOV @TADB,@#RCRC2          ;SAVE 1ST BYTE OF CRC FROM TA11
2629 011226 005037 177776          CLR @#PS          ;ALLOW INTERRUPTS
2630 011232 004037 011716          JSR R0,@#WAITFLAG          ;WAIT ON FLAG
2631 011236 104003          ERROR 3          ;NONE OCCURRED
2632 011240 104004          ERROR 4          ;INTERRUPT FAILED
2633 011242 111537 001215          MOV @TADB,RCRC2+1          ;SAVE 2ND BYTE OF CRC FROM TA11
2634 011246 004037 012014          JSR R0,@#FLAGS          ;CHECK FLAGS
    
```

```

2635 011252 104002          ERROR 2          ;UNKNOWN FLAG
2636 011254 104002          ERROR 2          ;"XFER REQ"
2637 011256 000002          RTI          ;"READY"
2638 011260 032714 037000          BIT #LEADERIWRLOCKIFGAPITMERR:OFFLINE,@TACS
2639 011264 001401          BEQ 6$          ;BR IF CRC ERROR
2640 011266 104002          ERROR 2          ;ERROR OCCURRED
2641 011270 032777 000020 167642 6$: BIT #SW04,@SWR          ;IGNORE CRC ERRORS?
2642 011276 001001          BNE 7$          ;BR IF YES
2643 011300 104012          ERROR 12          ;"READY" AND "CRC ERROR
2644 011302 000002          7$: RTI          ;IT WAS A CRC ERROR AND WE ARE IGNORING THEM
2645          ; SO RETURN TO CALLER
    
```

```

2646 ;*****
2647 ;THIS ROUTINE WILL BACK SPACE ONE FILE GAP
2648 T.BSFG: MOV (SP),@#SAVPC ;SAVE PC OF CALL
2649 011304 011637 001204 007304 MOV #MXBSFG,@#MFUNC ;SAVE POINTER OF .ASCIZ NAME
2650 011310 012737 015433 JSR RO,@#DDO.CMD ;START A "BSFG"
2651 011316 004037 012224 1$: XBSFGIINT.ENIGO
2652 011322 000107 CLR @#PS ;ALLOW INTERRUPTS
2653 011324 005037 177776 JSR RO,@#WAITFLAG ;WAIT ON FLAG
2654 011330 004037 011716 ERROR 3 ;NO FLAG
2655 011334 104003 ERROR 4 ;FLAG DIDN'T INTERRUPT
2656 011336 104004 JSR RO,@#FLAGS ;FIND OUT WHAT FLAG OCCURRED
2657 011340 004037 012014 ERROR 2 ;UNKNOWN
2658 011344 104002 ERROR 2 ;"XFER REQ"
2659 011346 104002 BR 2$ ;"READY"
2660 011350 004001 ERROR 2 ;"READY" & "ERROR"
2661 011352 104002 JSR @#ASKKEY ;RUNNING UNDER MANUAL LOOPING?
2662 011354 032714 004000 2$: BIT #FGAP,@TACS ;BR IF NO
2663 011360 001401 BEQ 3$ ;RETURN
2664 011362 000002 RTI ;GO BACK
2665 011364 104005 3$: ERROR 6 ;"BSFG" DIDN'T STOP IN A FILE GAP
2666
2667 ;*****
2668 ;THIS ROUTINE WILL BACK SPACE ONE BLOCK GAP
2669 T.BSBG: MOV (SP),@#SAVPC ;SAVE THE PC OF THE CALL
2670 011372 012737 015457 007304 MOV #MXBSBG,@#MFUNC ;MESSAGE POINTER
2671 011400 004037 012224 1$: JSR RO,@#DDO.CMD ;START A "BSBG"
2672 011404 000111 XBSBGINT.ENIGO
2673 011406 005037 177776 CLR @#PS ;ALLOW INTERRUPTS
2674 011412 004037 011716 JSR RO,@#WAITFLAG ;WAIT FOR A FLAG
2675 011416 104003 ERROR 3 ;NO FLAG OCCURRED
2676 011420 104004 ERROR 4 ;FLAG DIDN'T INTERRUPT
2677 011422 004037 012014 JSR RO,@#FLAGS ;WHAT FLAG OCCURRED
2678 011426 104002 ERROR 2 ;UNKNOWN
2679 011430 104002 ERROR 2 ;"XFER REQ"
2680 011432 000002 RTI ;"READY"-- GO BACK TO USER
2681 011434 104002 ERROR 2 ;"READY" & "ERROR"
    
```

```

2682 ;*****
2683 ;THIS ROUTINE WILL SPACE FORWARD FOR ONE FILE GAP
2684 T.SFFG: MOV (SP),@#SAVPC ;SAVE PC OF CALL
2685 011442 012737 015366 007304 MOV #MXSFFG,@#MFUNC ;POINTER TO .ASCIZ MESSAGE
2686 011450 004037 012224 1$: JSR RO,@#DDO.CMD ;START A "SFFG"
2687 011454 000113 XSFFGIINT.ENIGO
2688 011456 005037 177776 CLR @#PS ;ALLOW INTERRUPTS
2689 011462 004037 011716 JSR RO,@#WAITFLAG ;WAIT FOR A FLAG
2690 011466 104003 ERROR 3 ;FLAG FAILED TO OCCUR
2691 011470 104004 ERROR 4 ;FLAG DIDN'T INTERRUPT
2692 011472 004037 012014 JSR RO,@#FLAGS ;WHAT FLAG OCCURRED
2693 011476 104002 ERROR 2 ;UNKNOWN
2694 011500 104002 ERROR 2 ;"XFER REQ"
2695 011502 004001 BR 2$ ;"READY"
2696 011504 104002 ERROR 2 ;"READY" AND "ERROR"
2697 011506 032714 004000 2$: BIT #FGAP,@TACS ;DID "SFFG" STOP IN A "FILE GAP"?
2698 011512 001401 BEQ 3$ ;BR IF NO
2699 011514 000002 RTI ;RETURN TO CALLER
2700 011516 104006 3$: ERROR 6 ;"SFFG" DIDN'T STOP IN A FILE GAP
2701
2702 ;*****
2703 ;THIS ROUTINE WILL SPACE FORWARD ONE BLOCK GAP
2704 T.SFBG: MOV (SP),@#SAVPC ;SAVE PC OF CALL
2705 011520 011637 001204 007304 MOV #MXSFBG,@#MFUNC ;POINTER TO .ASCIZ MESSAGE
2706 011524 012737 015411 1$: JSR RO,@#DDO.CMD ;START A "SFBG"
2707 011532 004037 012224 XSFBIINT.ENIGO
2708 011536 000115 CLR @#PS ;ALLOW INTERRUPTS
2709 011540 005037 177776 JSR RO,@#WAITFLAG ;WAIT FOR A FLAG
2710 011544 004037 011716 ERROR 3 ;NO FLAG
2711 011550 104003 ERROR 4 ;NO INTERRUPT
2712 011552 104004 JSR RO,@#FLAGS ;FIND OUT WHAT FLAG
2713 011554 004037 012014 ERROR 2 ;UNKNOWN
2714 011560 104002 ERROR 2 ;"XFER REQ"
2715 011562 104002 ERROR 2 ;"READY"
2716 011564 000002 RTI ;"READY" AND "ERROR"
2717 011566 005737 001246 TST @#ASKKEY ;RUNNING UNDER MANUAL LOOPING?
2718 011572 002001 BGE 2$ ;BR IF NO
2719 011574 000002 RTI ;RETURN
2720 011576 104002 2$: ERROR 2 ;"READY" AND "ERROR"
    
```

```

;*****
;THIS ROUTINE WILL REWIND THE TAPE
T.RWND: MOV (SP),@#SAVPC ;SAVE PC OF CALL
MOV @#MWRWND,@#MFWNC ;POINTER TO .ASCIZ MESSAGE
JSR R0,@#FLAGS ;CHECK THE FLAGS
ERROR 1 ;NO FLAGS
ERROR 1 ;"XFER REQ"
BR 1$ ;NORMAL RETURN
TST @RWDFLG ;WAS LAST FUNCTION A REWIND?
BEQ 1$ ;BR IF YES
BIT #LEADER,@TACS ;ON CLEAR LEADER?
BEQ 1$ ;BR IF NO
CLR @RWDFLG ;ALLOW THIS ERROR ONLY ONE TIME
ERROR 1 ;ON CLEAR LEADER BUT SHOULDN'T BE
JSR R0,@#DD.COMD ;GO START A "REWIND"
XRWIND INT.ENIGO
CLR @#PS ;ALLOW INTERRUPTS
JSR R0,@#WAITFLAG ;WAIT ON FLAG
ERROR 3 ;NO FLAG OCCURRED
ERROR 4 ;NO INTERRUPT
JSR R0,@#FLAGS ;WHAT FLAG OCCURRED
ERROR 2 ;UNKNOWN
ERROR 2 ;"XFER REQ"
BR 2$ ;"READY"
ERROR 2 ;"READY" & "ERROR"
BIT #LEADER,@TACS ;ON "CLEAR LEADER"?
BEQ 3$ ;BR IF NO
RTI ;YES---RETURN
ERROR 7 ;"REWIND" FAILED TO GO TO "CLEAR LEADER"
  
```

```

;*****
;THIS ROUTINE IS USED TO WAIT ON A FLAG
;CALL:
; JSR R0,@#WAITFLAG
; RETURN FOR NO FLAGS
; RETURN FOR NO INTERRUPT
; NORMAL RETURN
;
;WAITFLAG:
2750 CLR LOCNT ;ZERO THE LOW COUNTER
2751 MOV MAXCNT,HICNT ;SET THE HIGH COUNTER TO MAX.COUNT
2752 BIT #SW05,@SWR ;RUNNING WITH INTERRUPTS?
2753 BEQ 1$ ;BR IF YES
2754 MOV #-1,WAITKEY ;SET THE INTERRUPT OCCURRED FLAG
2755 BIT #TR.REQUIREADY,@TACS ;TEST FOR FLAGS
2756 BNE 2$ ;BR IF A FLAG IS UP
2757 INC LOCNT ;COUNT THE LOW COUNTER
2758 BNE 1$ ;LOOP IF NOT ZERO
2759 DEC HICNT ;COUNT THE HIGH COUNTER
2760 BPL 1$ ;LOOP IF NOT NEG.
2761 BR 3$ ;LEAVE
2762 TST (R0)+ ;INCREMENT THE RETURN ADDRESS
2763 TST WAITKEY ;LOOK AT THE WAIT ON INT. KEY
2764 BEQ 3$ ;BR IF NOT SET
2765 TST (R0)+ ;INCREMENT THE RETURN ADDRESS
2766 RTS R0 ;RETURN TO USER
2767 LOCNT: 0
2768 HICNT: 0
2769 MAXCNT: 0
  
```

```

2780
2781 ;*****
2782 ;THIS ROUTINE IS USED TO DETERMINE WHAT FLAG IS SET
2783 ;
2784 ;
2785 ;CALL:
2786 ; JSR RO,@#FLAGS
2787 ; RETURN FOR UNKNOWN FLAG
2788 ; RETURN FOR "TRANSFER REQUEST"
2789 ; RETURN FOR "READY"
2790 ; RETURN FOR "READY" & "ERROR"
2791 ;
2792 012014 032714 000240 FLAGS: BIT #TR.REQ|READY,@TACS ;CHECK FOR "XFER REQ" OR "READY"
2793 012020 001427 BEQ LVFLAG ;BR IF NO FLAGS
2794 012022 005720 TST (R0)+ ;INCREMENT RETURN ADDRESS
2795 012024 105714 TSTB @TACS ;CHECK FOR "XFER REQ"
2796 012026 100424 BMI LVFLAG ;BR IF SET
2797 012030 005720 TST (R0)+ ;INCREMENT RETURN ADDRESS
2798 012032 032777 000100 167100 BIT #SW06,@SWR ;STALL?
2799 012040 001414 BEQ 3$ ;BR IF NO
2800 012042 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
2801 012044 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
2802 012046 013700 001264 MOV @#STALL,R0 ;GET STALL TIME
2803 012052 013701 001266 MOV @#STALL+2,R1
2804 012056 062700 000001 1$: ADD #1,R0 ;STALL
2805 012062 005501 ADC R1
2806 012064 001374 BNE 1$ ;LOOP
2807 012066 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
2808 012070 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
2809 012072 005714 3$: TST @TACS ;IS "ERROR" SET
2810 012074 100001 BPL LVFLAG ;BR IN NO "ERROR"
2811 012076 005720 TST (R0)+ ;INCREMENT RETURN ADDRESS
2812 012100 000200 LVFLAG: RTS RO
    
```

```

2813 ;*****
2814 ;THIS ROUTINE WILL CALCULATE THE CRC
2815 ;CALL:
2816 ; JSR PC,@#DO.CRC ;R0=1 BYTE OF DATA
2817 ;
2818 012102 DO.CRC:
2819 012102 010046 MOV RO,-(SP) ;;PUSH RO 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 1$: MOV CRC.WD,R3 ;PICKUP THE CRC WORD
2826 012126 005001 CLR R1
2827 012130 010302 MOV R3,R2 ;GET THE PARTIAL CRC WORD
2828 012132 042702 177776 BIC #CBIT00,R2 ;STRIP AWAY EVERYTHING BUT BIT00
2829 012136 006000 ROR RO ;PULL OFF THIS BIT
2830 012140 006101 ROL R1 ;AND SETUP TO XOR IT
2831 012142 010104 MOV R1,R4 ;FORM THE XOR OF "R1" AND "R2"
2832 012144 040204 BIC R2,R4
2833 012146 040102 BIC R1,R2
2834 012150 050402 BIS R4,R2 ;RESULTS TO "R2"
2835 012152 006002 ROR R2
2836 012154 103006 BCC 2$
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$: ROR R3
2843 012174 005327 3$: DEC (PC)+
2844 012176 000000 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 ;;POP STACK INTO 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 PC
2853 012222 000000 CRC.WD: 0 ;CRC WORD
    
```



```

2854 ;*****
2855 ;THIS ROUTINE WILL LOAD THE LOW BYTE OF TACS WITH THE DESIRED FUNCTION
2856 ;
2857 ;CALL:
2858 ; JSR RO,@#DO,CMD
2859 ; DESIRED FUNCTIN
2860 ; RETURN
2861 ;
2862 012224 010146 DO.CMD: MOV R1,-(SP) ;SAVE R1
2863 012226 012737 000340 177776 MOV #340,@#PS ;LOCK OUT ALL INERRUPTS
2864 012234 012001 MOV (RO)+,R1 ;GET THE COMMAND
2865 012236 042701 177640 BIC #<INT.ENIILBSIFUNCTION!GO>,R1 ;STRIP AWAY JUNK
2866 012242 032777 000040 166670 BIT #SW05,@SWR ;WITH OR WITHOUT INTERRUPTS
2867 012250 001405 BEQ 4$ ;BR IF WITH
2868 012252 042701 000100 BIC #INT.EN,R1 ;CLEAR INTERRUPT ENABLE IF SET
2869 012256 012777 012356 166744 MOV #BADINT,@TAVEC ;SETUP TO CATCH BAD INTERRUPT
2870 012264 005037 001254 4$: CLR WAITKEY ;ZERO THE WAIT ON INTERRUPT KEY
2871 012270 032701 000100 BIT #INT.EN,R1 ;RUNNING WITH INTERRUPTS?
2872 012274 001403 BEO 3$ ;BR IF NO
2873 012276 012777 012340 166724 MOV #SERV,@TAVEC ;SETUP TO SERVICE THE INTERRUPT
2874 012304 110114 3$: MOVB R1,@TACS ;LOAD THE COMMAND
2875 012306 032701 000001 BIT #GO,R1 ;IS GO BIT ON A "1"?
2876 012312 001403 BEQ 2$ ;BR IF GO = 0
2877 012314 032714 000040 1$: BIT #READY,@TACS ;WAIT FOR "READY" TO CLEAR
2878 012320 001375 BNE 1$
2879 012322 005101 2$: COM R1 ;SET OR CLEAR REWIND FLAG
2880 012324 042701 177761 BIC #<FUNCTION>,R1 ;EXTRACT FUNCTION BITS
2881 012330 010177 166722 MOV R1,@RWDFLG ;AND SAVE AS THE REWIND FLAG
2882 012334 012601 MOV (SP)+,R1 ;GET R1 BACK
2883 012336 000200 RTS RO ;RETURN TO USER
2884 ;
2885 ;*****
2886 ;THIS ROUTINE WILL SERVICE ALL LEGAL INTERRUPTS
2887 ;
2888 012340 012737 177777 001254 SERV: MOV #-1,@WAITKEY ;SET THE WAIT KEY
2889 012346 012777 012356 166654 MOV #BADINT,@TAVEC ;SET TO CATCH ILLEGAL INTERRUPTS
2890 012354 000002 RTI
2891 ;
2892 ;*****
2893 ;THIS ROUTINE WILL CATCH ILLEGAL INTERRUPTS FROM THE TA11
2894 ;
2895 012356 011637 001204 BADINT: MOV (SP),@#SAVPC ;SAVE WHERE WE WERE AT
2896 012362 104014 ERROR 14 ;ILLEGAL INTERRUPT OCCURRED
2897 ;
    
```

```

2898 ;*****
2899 ;THE FOLLOWING ROUTINES CAN BE USED TO MAKE ADJUSTMENTS TO THE TUGO
2900 ;NOTE: ### BEFORE USING ANY OF THE ROUTINES LOAD AND START AT 214 ###
2901 ;** NOTE: IF USING SOFTWARE SWITCH REGISTER THE
2902 ; PROGRAM MUST HAVE BEEN STARTED AT A
2903 ; NORMAL STARTING ADDRESS FIRST SO THAT
2904 ; THE AUTO-SIZING ROUTINE CAN SET UP ADDRESS
2905 ; 176 AS THE SOFTWARE SWITCH REGISTER
2906 ; PLACE VALUE INTO 176
2907 ;*****
2908 ;
2909 ;*****
2910 ;WRITE FILE GAPS FROM "BOT" TO "EOT"
2911 ;START AT 220
2912 ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND
2913 ;THE "WRITE DELAY MONO".
2914 ;*****
2915 WFGSUB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2916 MOV @#TACSL,TACS ;SETUP THE TA11 STATUS AND
2917 MOV @#TADBL,TADB ;DATA BUFFER REGISTERS
2918 012364 012706 001100 RESET ;RESET THE WORLD
2919 012370 013704 001220 MOV #WFGSUB,@#LPERR ;SETUP THE LOOP ON ERROR ADDRESS
2920 012374 013705 001224 SELDRV ;SELECT THE DRIVE
2921 012400 000005 MOV #XRWDIGO,@TACS ;SEND TAPE TO "BOT"
2922 012402 012737 012364 001110 BIT #READY,@TACS ;WAIT ON READY
2923 012410 104422 BEQ 100$
2924 012412 112714 000017 100$: WFG ;WRITE A FILE GAP
2925 012416 032714 000040 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2926 012422 001775 BEQ 1$ ;BR IF NO
2927 012424 104412 WFG ;WRITE A FILE GAP
2928 012426 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2929 012432 001774 BEQ 1$ ;BR IF NO
2930 012434 000000 HALT ;STOP IF YES
2931 012436 000752 BR WFGSUB ;LOOP ON CONT.
2932 ;
2933 ;*****
2934 ;WRITE CONTINUOUS BLOCKS OF DATA
2935 ;START AT 224
2936 ;THE PROGRAM WILL HALT THREE(3) TIMES
2937 ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
2938 ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
2939 ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
2940 ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
2941 ;THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"
2942 ;THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
2943 ;*****
2944 WRTSUB: JSR PC,@#SETBUF ;GET BLOCK SIZE AND PATTERN
2945 012440 004737 013106 MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2946 012444 012706 001100 WLOOP: MOV @#TACSL,TACS ;SETUP THE TA11 STATUS AND
2947 012450 013704 001220 MOV @#TADBL,TADB ;DATA BUFFER REGISTERS
2948 012454 013705 001224 RESET ;RESET THE WORLD
2949 012460 000005 MOV #WLOOP,@#LPERR ;SETUP THE LOOP ON ERROR ADDRESS
2950 012462 012737 012444 001110 SELDRV ;SELECT THE DRIVE
2951 012470 104422 MOVB #XRWDIGO,@TACS ;SEND TAPE TO "BOT"
2952 012472 112714 000017 100$: BIT #READY,@TACS ;WAIT ON READY
2953 012476 032714 000040 ;
    
```

```
2954 012502 001775 BEQ 1005 ;WRITE A BLOCK
2955 012504 104413 1$: WRITE ,WLNKX ;AT "CLEAR LEADER"?
2956 012510 032714 020000 BIT #LEADER,@TACS ;BR IF NO
2957 012514 001773 BEQ 1$ ;STOP IF "EOT"
2958 012516 000000 HALT ;LOOP IF CONT.
2959 012520 000751 BR WLOOP
2960
2961 ;*****
2962 ; READ AND COMPARE CONTINUOUS BLOCKS OF DATA
2963 ;
2964 ;START AT 230
2965 ;THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
2966 ;AND THE "THRESHOLD POT".
2967 ;*****
2968 012522 012706 001100 RDSUB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2969 012526 013704 001220 MOV @#TACSL,TACS ;SETUP THE TA11 STATUS AND
2970 012532 013705 001224 MOV @#TADBL,TADB ;DATA BUFFER REGISTERS
2971 012536 000005 RESET ;RESET THE WORLD
2972 012540 012737 001252 001110 MOV #RDSUB,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2973 012546 104422 SELDRV ;SELECT THE DRIVE
2974 012550 112714 000017 MOV #XRWDIGO,@TACS ;SEND TAPE TO "BOT"
2975 012554 032714 000040 100$: BIT #READY,@TACS ;WAIT ON READY
2976 012560 001775 BEQ 1005
2977 012562 104414 017006 1$: READ ,RLNKX ;READ A BLOCK
2978 012566 012706 001100 MOV #STACK,SP ;INIT. IN CASE OF ERROR
2979 012572 032714 020900 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2980 012576 001351 BNE RDSUB ;BR IF YES---LOOP
2981 012600 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
2982 012602 017006 RLNKX ;LINK TO THE READ BUFFER
2983 012604 017042 WLNKX ;LINK TO THE WRITE BUFFER
2984 012606 000765 BR 1$
2985
2986 ;*****
2987 ; WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
2988 ;
2989 ;START AT 234
2990 ;THE PROGRAM WILL HALT THREE(3) TIMES
2991 ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
2992 ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
2993 ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
2994 ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
2995 ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
2996 ;AND THE "GAP TIME MONO".
```

```
2997 ;THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
2998 ;*****
2999 012610 004737 013106 WGPBLK: JSR PC,@#SETBUF ;GET BLOCK SIZE AND PATTERN
3000 012614 012706 001100 WGBLOP: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
3001 012620 013704 001220 MOV @#TACSL,TACS ;SETUP THE TA11 STATUS AND
3002 012624 013705 001224 MOV @#TADBL,TADB ;DATA BUFFER REGISTERS
3003 012630 000005 RESET ;RESET THE WORLD
3004 012632 012737 012614 001110 MOV #WGBLOP,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3005 012640 104422 SELDRV ;SELECT THE DRIVE
3006 012642 112714 000017 MOV #XRWDIGO,@TACS ;SEND TAPE TO "BOT"
3007 012646 032714 000040 100$: BIT #READY,@TACS ;WAIT ON READY
3008 012652 001775 BEQ 1005
3009 012654 104412 1$: WFG ;WRITE A FILE GAP
3010 012656 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
3011 012662 001005 BNE 2$ ;BR IF YES
3012 012664 104413 017042 WRITE ,WLNKX ;WRITE A BLOCK
3013 012670 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
3014 012674 001767 BEQ 1$ ;BR IF NO
3015 012676 000000 2$: HALT ;STOP AT "EOT"
3016 012700 000745 BR WGBLOP ;START OVER ON CONT.
3017
3018 ;*****
3019 ; READ AND COMPARE A BLOCK OF DATA AND A FILE GAP
3020 ;
3021 ;START AT 240
3022 ;THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
3023 ;IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
3024 ;AND THE "TAPE BLANK MONO".
3025 ;*****
3026 012702 012706 001100 RGPBLK: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
3027 012706 013704 001220 MOV @#TACSL,TACS ;SETUP THE TA11 STATUS AND
3028 012712 013705 001224 MOV @#TADBL,TADB ;DATA BUFFER REGISTERS
3029 012716 000005 RESET ;RESET THE WORLD
3030 012720 012737 012702 001110 MOV #RGPBLK,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3031 012726 104422 SELDRV ;SELECT THE DRIVE
3032 012730 112714 000017 MOV #XRWDIGO,@TACS ;SEND TAPE TO "BOT"
3033 012734 032714 000040 100$: BIT #READY,@TACS ;WAIT ON READY
3034 012740 001775 BEQ 1005
3035 012742 104414 017006 1$: READ ,RLNKX ;READ A BLOCK OF DATA
3036 012746 012706 001100 MOV #STACK,SP ;KEEP OUT OF THE WAY
3037 012752 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
3038 012756 001351 BNE RGPBLK ;BR IF YES
3039 012760 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
3040 012762 017006 RLNKX ;LINK TO THE READ BUFFER
3041 012764 017042 WLNKX ;LINK TO THE WRITE BUFFER
3042 012766 104420 SFBG ;GET INTO FILE GAP
3043 012770 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
3044 012774 001342 BNE RGPBLK ;BR IF YES
3045 012776 000761 BR 1$ ;LOOP
3046
3047 ;*****
3048 ; SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
3049 ;
3050 ;START AT 244
3051 ;THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
3052 ;SPACE FOWARD (TAPE BLANK MONO CAN BE ADJUSTED). OR AFTER READ OR
```

```

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 TA11 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,@#$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
3061 013024 104422 SELDRV ;SELECT THE DRIVE
3062 013026 112714 000017 MOVB #XRWNDIGO,@TACS ;SEND TAPE TO "BOT"
3063 013032 032714 000040 100$: 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 ; BACK SPACE FILE GAP
3073 ; START AT 250
3074 ; THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
3075 ;*****
3076 ;*****
3077 013054 000005 BSFGS: RESET ;RESET THE WORLD
3078 013056 012737 013054 001110 MOV #BSFGS,@#$LPERR ;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 BSFGS ;START OVER ON CONT.
3086
3087 ;*****
3088 ; SETUP READ AND WRITE LINKS FOR SUBROUTINES
3089 ;*****
3090 013106 005000 SETBUF: CLR R0
3091 013110 000000 HALT
3092 013112 022737 000176 001140 CMP #SWREG,SWR ;OPERATOR PUTS BYTE COUNT IN THE SWR
3093 013120 001001 BNE 20$ ;USING S/W SWITCH REG?
3094 013122 104405 GTSWR ;NO- GET OUT
3095 013124 20$: ;GET VALUE
3096 013124 157700 166010 BISB @SWR,R0 ;PICKUP THE BYTE COUNT
3097 013130 001006 BNE 2$ ;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,R0 ;SET FOR MAX ALLOWED
3101 013144 005200 1$: INC R0 ;MAKE IT 377 OR 1
3102 013146 010037 017010 2$: MOV R0,@#RLNKX+2 ;SETUP READ LINK
3103 013152 010037 017044 MOV R0,@#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 MOV @SWR,R0 ;PICKUP THE PATTERN
3110 013176 012701 017112 MOV #WBUF,R1 ;PICKUP FIRST ADDRESS OF BUFFER
3111 013202 110021 MOVB R0,(R1)+ ;FILL THE BUFFER
3112 013204 110021 MOVB R0,(R1)+
3113 013206 110021 MOVB R0,(R1)+
3114 013210 110021 MOVB R0,(R1)+
3115 013212 000000 HALT ;SET OPERATIONAL SWITCHES
3116 013214 022737 000176 001140 CMP #SWREG,SWR ;USING S/W SWITCH REG?
3117 013222 001001 BNE 22$ ;NO- GET OUT
3118 013224 104405 GTSWR ;GET VALUE
3119 013226 22$: ;CONTINUE
3120 013226 000207 RTS PC ;RETURN
    
```

```
.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 $TPFLG ;;IS THERE A TERMINAL?
BPL 1$ ;;BR IF YES
HALT ;;HALT HERE IF NO TERMINAL
BR 3$ ;;LEAVE
1$: MOV RO,-(SP) ;;SAVE RO
MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
2$: MOVB (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$: CMPB #HT,(SP) ;;BRANCH IF <HT>
BEQ 8$
CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
BNE 5$
TST (SP)+ ;;POP <CR><LF> EQUIV
TYPE ;;TYPE A CR AND LF
$CRLF
CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
BR 2$ ;;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
```

```
3177 013376 000724 BR 2$ ;;GET NEXT CHARACTER
3178 013400 105777 165544 $TYPEC: TSTB @STPS ;;WAIT UNTIL PRINTER IS READY
BPL $TYPEC
3180 013406 116677 000002 165536 MOVB 2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3181 013414 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
BNE 1$ ;;BRANCH IF NO
3182 013422 001003 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
3183 013424 105037 013444 BR $TYPEC ;;EXIT
3184 013430 000406 BR $TYPEC
3185 013432 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
BEQ $TYPEC ;;BRANCH IF YES
INCB (PC)+ ;;COUNT THE CHARACTER
$CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
3189 013446 000207 $TYPEC: RTS PC

.SBTTL TTY INPUT ROUTINE
*****
.ENABL LSB
*****
;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
;SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
;WHEN OPERATING IN TTY FLAG MODE.
3201 013450 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
BNE 15$ ;;BRANCH IF NO
TSTB @STKS ;;CHAR THERE?
BPL 15$ ;;IF NO, DON'T WAIT AROUND
MOVB @STKB,-(SP) ;;SAVE THE CHAR
BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
3207 013476 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
BNE 15$ ;;NO, RETURN TO USER
3209 013504 123727 001134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
BEQ 15$ ;;BRANCH IF YES
3212 013514 104401 014175 TYPE ,SCNTLG ;;ECHO THE CONTROL-G (^G)
3213 013520 104401 014202 $GTSWR: TYPE ,MSWR ;;TYPE CURRENT CONTENTS
MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEDUT
3215 013530 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3216 013532 104401 014213 TYPE ,SMNEW ;;PROMPT FOR NEW SWR
19$: CLR -(SP) ;;CLEAR COUNTER
CLR -(SP) ;;THE NEW SWR
7$: TSTB @STKS ;;CHAR THERE?
BPL 7$ ;;IF NOT TRY AGAIN
3222 013550 117746 165372 MOVB @STKB,-(SP) ;;PICK UP CHAR
3223 013554 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
3224
3225
3226
3227 013560 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
BNE 10$ ;;BRANCH IF NOT
TYPE ,SCNTLU ;;YES, ECHO CONTROL-U (^U)
3230 013572 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
BR 19$ ;;LET'S TRY IT AGAIN
3232
```

```

3233      013600 021627 000015      10$:  CMP      (SP),#15      ;;IS IT A <CR>?
3234      013600 001022      BNE      16$              ;;BRANCH IF NO
3235      013604 001022      TST      4(SP)            ;;YES, IS IT THE FIRST CHAR?
3236      013606 005766 000004      BEQ      11$              ;;BRANCH IF YES
3237      013612 001403      MOV      2(SP),@SWR      ;;SAVE NEW SWR
3238      013614 016677 000002 165316      ADD      #6,SP            ;;CLEAR UP STACK
3239      013622 062706 000006      TYPE    $SCRLF           ;;ECHO <CR> AND <LF>
3240      013626 104401 001201      14$:  CMPB    $INTAG,#1      ;;RE-ENABLE TTY KBD INTERRUPTS?
3241      013632 123727 001135 000001      BNE      15$              ;;BRANCH IF NOT
3242      013640 001003      MOV      #100,@STKS      ;;RE-ENABLE TTY KBD INTERRUPTS
3243      013642 012777 000100 165274      BNE      15$              ;;RETURN
3244      013650 000002      JSR     PC,$TYPEC        ;;ECHO CHAR
3245      013652 004737 013400      16$:  CMP      (SP),#60      ;;CHAR < 0?
3246      013656 021627 000060      BLT     18$              ;;BRANCH IF YES
3247      013662 002420      CMP      (SP),#67      ;;CHAR > ??
3248      013664 021627 000067      BGT     18$              ;;BRANCH IF YES
3249      013670 003015      BIC     #60,(SP)+        ;;STRIP-OFF ASCII
3250      013672 042726 000060      TST     2(SP)            ;;IS THIS THE FIRST CHAR
3251      013676 005766 000002      BEQ     17$              ;;BRANCH IF YES
3252      013702 001403      ASL     (SP)              ;;NO, SHIFT PRESENT
3253      013704 006316      ASL     (SP)              ;; CHAR OVER TO MAKE
3254      013706 006316      ASL     (SP)              ;; ROOM FOR NEW ONE.
3255      013710 006316      ASL     (SP)              ;;KEEP COUNT OF CHAR
3256      013712 005266 000002      17$:  INC     2(SP)            ;;SET IN NEW CHAR
3257      013716 056616 177776      BIS     -2(SP),(SP)      ;;GET THE NEXT ONE
3258      013722 000707      BR      7$                ;;TYPE ?<CR><LF>
3259      013724 104401 001200      18$:  TYPE    $QUES          ;;SIMULATE CONTROL-U
3260      013730 000720      BR      20$
3261      .DSABL  LSB
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
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     #'C<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     #'C<177>,2(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
3295
3296
3297
3298
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  (SP)+,(R3)        ;;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   -(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      .BLKB  B.                ;;RESERVE B BYTES FOR TTY INPUT
3324      014170 052536 005015 000      SCNTLU: .ASCIZ  '/U<15><12>'  ;;CONTROL "U"
3325      014175 136 006507 000012 000012      SCNTLG: .ASCIZ  '/G<15><12>'  ;;CONTROL "G"
3326      014202 005015 053523 020122      SMSWR:  .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
3334
3335
3336
3337
3338
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      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      MOV R1,12(SP)  ;;SAVE THE RESULT
3362 014310 010237      MOV R2,SHIOCT
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      SHIOCT: .WORD 0 ;;HIGH ORDER BITS GO HERE
    
```

```

3368                                     .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
3369
3370                                     ;;*****
3371                                     ;;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3372                                     ;;OCTAL (ASCII) NUMBER AND TYPE IT.
3373                                     ;;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3374                                     ;;*CALL:
3375                                     ;;*   MOV   NUM,-(SP)      ;;NUMBER TO BE TYPED
3376                                     ;;*   TYPOS      ;;CALL FOR TYPEOUT
3377                                     ;;*   .BYTE  N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3378                                     ;;*   .BYTE  M          ;;M=1 OR 0
3379                                     ;;*                                     ;;1=TYPE LEADING ZEROS
3380                                     ;;*                                     ;;0=SUPPRESS LEADING ZEROS
3381                                     ;;*
3382                                     ;;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3383                                     ;;*$TYPOS OR $TYPOC
3384                                     ;;*CALL:
3385                                     ;;*   MOV   NUM,-(SP)      ;;NUMBER TO BE TYPED
3386                                     ;;*   TYPON      ;;CALL FOR TYPEOUT
3387                                     ;;*
3388                                     ;;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3389                                     ;;*CALL:
3390                                     ;;*   MOV   NUM,-(SP)      ;;NUMBER TO BE TYPED
3391                                     ;;*   TYPOC      ;;CALL FOR TYPEOUT
3392                                     ;;*
3393 014326 017646 000000      STYPOS: MOV @ (SP),-(SP) ;;PICKUP THE MODE
3394 014332 116837 000001 014551 MOVB 1(SP),SOFILL ;;LOAD ZERO FILL SWITCH
3395 014340 112637 014553 MOVB (SP)+,SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
3396 014344 062716 000002      ADD #2,(SP) ;;ADJUST RETURN ADDRESS
3397 014350 000406      BR $TYPON
3398 014352 112737 000001 014551 $TYPOC: MOVB #1,SOFILL ;;SET THE ZERO FILL SWITCH
3399 014360 112737 000006 014553 MOVB #6,SOMODE+1 ;;SET FOR SIX(6) DIGITS
3400 014366 112737 000005 014550 $TYPON: MOVB #5,SOCNT ;;SET THE ITERATION COUNT
3401 014374 010346      MOV R3,-(SP) ;;SAVE R3
3402 014376 010446      MOV R4,-(SP) ;;SAVE R4
3403 014400 010546      MOV R5,-(SP) ;;SAVE R5
3404 014402 113704 014553      MOVB $OMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
3405 014406 005404      NEG R4
3406 014410 062704 000006      ADD #6,R4 ;;SUBTRACT IT FOR MAX. ALLOWED
3407 014414 110437 014552      MOVB R4,SOMODE ;;SAVE IT FOR USE
3408 014420 113704 014551      MOVB $OFILL,R4 ;;GET THE ZERO FILL SWITCH
3409 014424 016605 000012      MOV 12(SP),R5 ;;PICKUP THE INPUT NUMBER
3410 014430 005003      CLR R3 ;;CLEAR THE OUTPUT WORD
3411 014432 006105      1$: ROL R5 ;;ROTATE MSB INTO "C"
3412 014434 000404      BR 3$ ;;GO DO MSB
3413 014436 006105      2$: ROL R5 ;;FORM THIS DIGIT
3414 014440 006105      ROL R5
3415 014442 006105      ROL R5
3416 014444 010503      MOV R5,R3
3417 014446 006103      3$: ROL R3 ;;GET LSB OF THIS DIGIT
3418 014450 105337 014552      DECB $OMODE ;;TYPE THIS DIGIT?
3419 014454 100016      BPL 7$ ;;BR IF NO
3420 014456 042703 177770      BIC #177770,R3 ;;GET RID OF JUNK
3421 014462 001002      BNE 4$ ;;TEST FOR 0
3422 014464 005704      TST R4 ;;SUPPRESS THIS 0?
3423 014466 001403      BEQ 5$ ;;BR IF YES
    
```

```

3424 014470 005204      4$: INC R4 ;;DON'T SUPPRESS ANYMORE 0'S
3425 014472 052703 000060      BIS #'0,R3 ;;MAKE THIS DIGIT ASCII
3426 014476 052703 000040      5$: BIS #' ,R3 ;;MAKE ASCII IF NOT ALREADY
3427 014502 110337 014546      MOVVB R3,B$ ;;SAVE FOR TYPING
3428 014506 104401 014546      TYPE ,B$ ;;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
    
```

```

3445      .SBTTL TRAP DECODER
3446
3447      ;;*****
3448      ;;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3449      ;;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3450      ;;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3451      ;;GO TO THAT ROUTINE.
3452
3453 014554 010046      $TRAP: MOV RO,-(SP) ;;SAVE RO
3454 014556 016600 000002      MOV 2(SP),RO ;;GET TRAP ADDRESS
3455 014562 005740      TST -(RO) ;;BACKUP BY 2
3456 014564 111000      MOVVB (RO),RO ;;GET RIGHT BYTE OF TRA
3457 014566 006300      ASL RO ;;POSITION FOR INDEXING
3458 014570 016000 014610      MOV $TRPAD(RO),RO ;;INDEX TO TABLE
3459 014574 000200      RTS RO ;;GO TO ROUTINE
3460
3461      ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3462
3463
3464 014576 011646      $TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
3465 014600 016666 000004 000002      MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
3466 014606 000002      RTI ;;RESTORE THE PSW
3467
3468      .SBTTL TRAP TABLE
3469
3470      ;;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3471      ;;BY THE "TRAP" INSTRUCTION.
3472
3473      ; ROUTINE
3474      ; -----
3475 014610 014576      $TRPAD: .WORD $TRAP2
3476 014612 013230      STYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
3477 014614 014352      STYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3478 014616 014326      STYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
3479 014620 014366      STYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
3480
3481 014622 013520      SGTSWR ;;CALL=GTSWR TRAP+5(104405) GET SOFT-SWR SETTING
3482
3483 014624 013450      SCKSWR ;;CALL=CKSWR TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
3484 014626 013732      SRDCHR ;;CALL=RDCHR TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
3485 014630 014052      SRDLIN ;;CALL=RDLIN TRAP+10(104410) TTY TYPEIN STRING ROUTINE
3486 014632 014224      SRDOCT ;;CALL=RDDOCT TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
3487 014634 010504      T.WFG ;;CALL=WFG TRAP+12(104412) ROUTINE TO WRITE A FILE GAP
3488 014636 010554      T.WRIT ;;CALL=WRITE TRAP+13(104413) ROUTINE TO WRITE A BLOCK OF DATA
3489 014640 010774      T.READ ;;CALL=READ TRAP+14(104414) ROUTINE TO READ A BLOCK OF DATA
3490 014642 011304      T.BSFG ;;CALL=BSFG TRAP+15(104415) ROUTINE TO BACK SPACE A FILE GAP
3491 014644 011366      T.BSBG ;;CALL=BSBG TRAP+16(104416) ROUTINE TO BACK SPACE A BLOCK GAP
3492 014646 011436      T.SFFG ;;CALL=SFFG TRAP+17(104417) ROUTINE TO SPACE FWD A FILE GAP
3493 014650 011520      T.SFBG ;;CALL=SFBG TRAP+20(104420) ROUTINE TO SPACE FWD A BLOCK GAP
3494 014652 011600      T.RWND ;;CALL=REWIND TRAP+21(104421) ROUTINE TO REWIND THE TAPE
3495 014654 010246      T.SELDRV ;;CALL=SELDRV TRAP+22(104422) ROUTINE TO SELECT THE NEXT DRIVE
3496 014656 010362      T.BLKCMP ;;CALL=BLKCMP TRAP+23(104423) CCOMPARE READ AND WRITE BUFFERS
3497 014660 010106      T.SETLOOP ;;CALL=SETLOOP TRAP+24(104424) SETUP TO LOOP AS PER THE TTY
    
```

```

3498 .SBTTL POWER DOWN AND UP ROUTINES
3499
3500 ;*****
3501 ;POWER DOWN ROUTINE
3502 014662 012737 015026 000024 $PWRDN: MOV #SILLUP,@#PWRVEC ;;SET FOR FAST UP
3503 014670 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
3504 014676 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
3505 014700 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
3506 014702 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
3507 014704 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
3508 014706 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
3509 014710 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
3510 014712 017746 164222 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
3511 014716 010637 015032 MOV SP,$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 ;POWER UP ROUTINE
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 $SAVR6 ;;WAIT LOOP FOR THE TTY
3521 014752 005237 015032 1$: INC $SAVR6 ;;WAIT FOR THE INC
3522 014756 001375 BNE 1$ ;;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 $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
3534 015020 012716 MOV (PC)+,(SP) ;;RESTART AT PWRST
3535 015022 002300 $PWRAD: .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: .ASCIZ <15><12>"POWER"
3541 015042 000122 .EVEN
3542

```

```

3543 ;*****
3544 ;DATA POINTERS
3545
3546 DT1: .WORD SAVPC,$ERRPC,$REG0,$REG1,0
3547 015044 001204 001116 001162
3548 015052 001164 000000
3549 DT2: .WORD SAVPC,$ERRPC,$REG0,$DDAT,$BDDAT, BYTNUM,0
3550 015056 001204 001116 001162
3551 015064 001124 001126 001206
3552 015072 000000
3553 DT3: .WORD SAVPC,$ERRPC,$REG0,RCRC0,RCRC1,RCRC2,0
3554 015074 001204 001116 001162
3555 015102 001210 001212 001214
3556 015110 000000
3557 DT4: .WORD SAVPC,$ERRPC,$REG0,$REG1,$TMP0,0
3558 015112 001204 001116 001162
3559 015120 001164 001166 000000
3560 DT5: .WORD SAVPC,$ERRPC,$REG0,0
3561 015126 001204 001116 001162
3562 015134 000000
3563

```



```

3564 ;*****
3565 ;MESSAGES
3566
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 015244 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 MLOOP: .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 040505 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 047440 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
3667 016104 042524 052123 020040
3668 016112 020040 051105 047522
3669 016120 006522 012
3670 016123 120 020103 020040
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          DH2: .ASCII <15><12>
3677 016164 042524 052123 020040 .ASCII /TEST  ERROR          EXPT'D  RCV'D  BYTE/<15><12>
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 .ASCIZ /PC  PC  TACS  DATA  DATA  NUMBER/<15><12>
3686 016250 020040 041520 020040
3687 016256 020040 020040 040524
3688 016264 051503 020040 020040
3689 016272 040504 040524 020040
3690 016308 020040 040504 040524
3691 016308 020040 020040 052516
3692 016314 041115 051105 005015
3693 016322 000
3694 016323 015 012
3695 016325 124 051505 020124 DH3: .ASCII <15><12>
3696 016332 020040 042440 051122 .ASCII /TEST  ERROR          WRITE  READ  TU60/<15><12>
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 .ASCIZ /PC  PC  TACS  CRC  CRC  CRC/<15><12>
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 DH4: .ASCII <15><12>
3712 016463 124 051505 020124 .ASCII /TEST  ERROR          BYTES/<15><12>
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 .ASCIZ /PC  PC  TACS  TADB  LEFT/<15><12>
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 DH5: .ASCII <15><12>
3727 016603 124 051505 020124 .ASCII /TEST  ERROR/<15><12>
3728 016610 020040 042440 051122
3729 016616 051117 005015
3730 016622 041520 020040 020040 .ASCIZ /PC  PC  TACS/<15><12>
3731 016630 020040 041520 020040
```

```
3732 016636 020040 020040 040524
3733 016644 051503 005015 000
3734 016652 016652          .EVEN
3735 016652 001116 001220 000000 DT201: .WORD $ERRPC,TACSL,0
3736
3737 016660 001116 000000 DT202: .WORD $ERRPC,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 ;BUFFER LINKS
3754 ;THESE LINKS POINT TO THE STARTING ADDRESS OF THE BUFFERS
3755 ;AND ALSO INDICATE HOW MANY BYTES TO READ OR WRITE
3756
3757 016756 017120 000200 RLNK1: RBUF1,128.
3758 016762 017322 000200 RLNK2: RBUF2,128.
3759 016766 017524 000010 RLNK3: RBUF3,8.
3760 016772 017726 000040 RLNK4: RBUF4,32.
3761 016776 020130 000020 RLNK5: RBUF5,16.
3762 017002 020332 000100 RLNK6: RBUF6,64.
3763 017006 017120 000000 RLNKX: RBUF1,0.
3764 017012 017046 000200 WLNK1: WBUF1,128.
3765 017016 017054 000200 WLNK2: WBUF2,128.
3766 017022 017062 000010 WLNK3: WBUF3,8.
3767 017026 017070 000040 WLNK4: WBUF4,32.
3768 017032 017076 000020 WLNK5: WBUF5,16.
3769 017036 017104 000100 WLNK6: WBUF6,64.
3770 017042 017112 000000 WLNKX: WBUF6,0.
3771
3772 ;BUFFERS
3773
3774 017046 000002 WBUF1:
3775 017046 314 .BYTE ^B11001100
3776 017047 063 .BYTE ^B00110011
3777 017050 314 .BYTE ^B11001100
3778 017051 063 .BYTE ^B00110011
3779 017052 000000 WCRC1: .WORD 0 ;CRC STORAGE FOR WBUF1
3780
3781 017054 000002 WBUF2:
3782 017054 377 .BYTE ^B11111111
3783 017055 000 .BYTE ^B00000000
3784 017056 377 .BYTE ^B11111111
3785 017057 000 .BYTE ^B00000000
3786 017060 000000 WCRC2: .WORD 0 ;CRC STORAGE FOR WBUF2
3787
3788 017062 000002 WBUF3:
3789 017062 231 .BYTE ^B10011001
3790 017063 146 .BYTE ^B01100110
3791 017064 231 .BYTE ^B10011001
3792 017065 146 .BYTE ^B01100110
3793 017066 000000 WCRC3: .WORD 0 ;CRC STORAGE FOR WBUF3
3794
3795 017070 000002 WBUF4:
3796 017070 360 .BYTE ^B11110000
3797 017071 017 .BYTE ^B00001111
3798 017072 360 .BYTE ^B11110000
3799 017073 017 .BYTE ^B00001111
3800 017074 000000 WCRC4: .WORD 0 ;CRC STORAGE FOR WBUF4
3801
3802 017076 000002 WBUF5:
3803 017076 252 .BYTE ^B10101010
3804 017077 125 .BYTE ^B01010101
3805 017100 252 .BYTE ^B10101010
3806 017101 125 .BYTE ^B01010101
3807 017102 000000 WCRC5: .WORD 0 ;CRC STORAGE FOR WBUF5
    
```

```

3808
3809 017104 000001 WBUF6:
3810 017104 377 .BYTE ^B11111111
3811 017105 377 .BYTE ^B11111111
3812 017106 000 .BYTE ^B00000000
3813 017107 000 .BYTE ^B00000000
3814 017110 000000 WCRC6: .WORD 0 ;CRC STORAGE FOR WBUF6
3815
3816 017112 000004 WBUF7:
3817 017112 000 .BYTE 0
3818 017113 000 .BYTE 0
3819 017114 000 .BYTE 0
3820 017115 000 .BYTE 0
3821 017116 000000 WCRCX: .WORD 0
3822
3823
3824 017120 000202 RBUF1: .BLKB 130.
3825 017322 000202 RBUF2: .BLKB 130.
3826 017524 000202 RBUF3: .BLKB 130.
3827 017726 000202 RBUF4: .BLKB 130.
3828 020130 000202 RBUF5: .BLKB 130.
3829 020332 000202 RBUF6: .BLKB 130.
3830
3831 020534 LASTADDRESS=
3832
3833 000001 .END
    
```

Table with columns for symbols (e.g., ASKADR, ASKDRV, ASKKEY, A2OCT, BADINT, BEGINX, BEGIN1, BEGIN2, BEGIN3, BEGIN4, BGNCMN, BIT0, BIT00, BIT01, BIT02, BIT03, BIT04, BIT05, BIT06, BIT07, BIT08, BIT09, BIT1, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15, BIT2, BIT3, BIT4, BIT5, BIT6, BIT7, BIT8, BIT9, BLKMP, BPTVEC, BSG, BSFG, BSFGB, BYNUM, CHKADR, CHKDRV, CKSWR, CR, CRCERR) and corresponding numerical values.

Table with columns for symbols (e.g., CRC.WD, CRLF, CURDRV, DDISP, DH1, DH2, DH201, DH202, DH3, DH4, DH5, DISPRA, DISPRE, DD.CMD, DD.CRC, DRIVE, DRVKEY, DRVPNT, DSWR, DT1, DT2, DT201, DT202, DT3, DT4, DTS, EMTVEC, EM1, EM10, EM11, EM12, EM13, EM14, EM2, EM201, EM202, EM3, EM4, EM5, EM6, EM7, ERROR, ERRVEC, EXAM, EXIT, FATAL, FGAP, FLAGS, FUNCT1, FUNC0, FUNC1, FUNC2, GNS, GO) and corresponding numerical values.







\$MNEW 014213	3216	3328#														
\$MSWR 014202	3213	3326#														
\$MXCNT 001216	882#	1207*	1219#	2182	2191											
\$NULL 001154	860#	3162	3191													
\$NWTST= 000001	1285#	1298#	1311#	1324#	1342#	1360#	1375#	1394#	1409#	1424#	1439#	1455#	1470#			
	1520#	1565#	1579#	1593#	1613#	1633#	1649#	1666#	1685#	1705#	1724#	1744#	1763#			
	1779#	1801#	1821#	1840#	1858#	1876#	1901#	1929#	1972#	2005#						
\$OCNT 014550	3400*	3429*	3442#													
\$OMODE 014552	3395*	3399*	3404	3407*	3418*	3444#										
\$OVER 007032	2148	2150	2170	2183	2188#											
\$PASS 001100	835#	1229	2070*	2071*	2090	2176	2191									
\$POWER 015034	3533	3540#														
\$PWRAD 015022	3535#															
\$PWRDN 014662	1049	3502#	3530													
\$PWRMG 015016	3533#															
\$PWRUP 014734	3512	3518#														
\$QUES 001200	872#	2253	2319	2384	3191	3259	3308	3324								
\$RDCHR 013732	3272#	3484														
\$RDDEC= ***** U	3487															
\$RDLIN 014052	3300#	3485														
\$RDODCT 014224	3340#	3486														
\$RDSZ = 000010	3293#															
\$REGAD 001160	864#															
\$REGO 001162	866#	2207*	2216	3547	3550	3554	3558	3561								
\$REG1 001164	867#	2208*	3547	3558												
\$RTNAD 006444	2089#															
\$R2A = ***** U	3487															
\$SAVRE= ***** U	3487															
\$SAVR6 015032	3511*	3519	3520*	3521*	3539#											
\$SCOPE 006570	1043	2138#														
\$SETUP= 000137	1022#	1042	1043	1045	1047	1049	1051	1052	1053	1055	1077	1080	2068			
	2139	2206	2240	2248	3196	3330										
\$SS = 000000	1285#															
\$STUP = 177777	1022#															
\$SVLAD 007004	2158	2183#														
\$SWR = 167000	582#	592	642	643	644	645	646	647	648	869	870	871	1052			
	1053	1055	1056	1289	1302	1315	1328	1346	1364	1379	1398	1413	1428			
	1443	1459	1474	1524	1569	1583	1597	1617	1637	1653	1670	1689	1709			
	1728	1748	1767	1783	1805	1825	1844	1852	1880	1905	1933	1976	2009			
	2061	2069	2082	2088	2090	2131	2132	2133	2134	2135	2149	2161	2163			
	2164	2165	2172	2173	2174	2185	2188	2191	2197	2198	2199	2200	2201			
	2225	2232	2237	2241	2253	2483	3536									
	2135															
\$SWRMK= 000000	869#	1052*	2069*	2172*	2179	2182*	2191									
\$TIMES 001170	857#	3194	3205	3222	3276	3282										
\$TKB 001146	856#	2140	3194	3203	3219	3243*	3274	3280								
\$TKS 001144	868#	2615*	3558													
\$TMPD 001166	582#	592	1285	1289#	1298	1302#	1311	1315#	1324	1328#	1342	1346#	1360			
\$TN = 000045	1364#	1375	1379#	1394	1398#	1409	1413#	1424	1428#	1439	1443#	1455	1459#			
	1470	1474#	1520	1524#	1565	1569#	1579	1583#	1593	1597#	1613	1617#	1633			
	1637#	1649	1650	1653#	1666	1670#	1685	1689#	1705	1709#	1724	1728#	1744	1748#		
	1763	1767#	1779	1783#	1801	1805#	1821	1825#	1840	1844#	1858	1862#	1876			
	1880#	1901	1905#	1929	1933#	1972	1976#	2005	2009#							
	859#	3180*	3191													
\$TPB 001152	853#	3138	3191													
\$TPFLG 001157	858#	3178	3191													
\$TPS 001150																

\$TRAP 014554	1047	3453#														
\$TRAP2 014576	3464#	3475														
\$TRP = 000025	3468#	3477#	3478#	3479#	3480#	3481	3482#	3483	3484#	3485#	3486#	3487#	3488#			
	3489#	3490#	3491#	3492#	3493#	3494#	3495#	3496#	3497#	3498#						
	3458	3475#														
\$TRPAD 014610	836#	1205*	2068*	2130	2183*	2188	2191	2224	2253							
\$TSTNM 001102	3301	3302	3319	3323#												
\$TTYIN 014160	3480															
\$TYPBN= ***** U	3480															
\$TYPDS= ***** U	3138#	3468	3476													
\$TYPE 013230	3159	3166	3173	3178#	3179	3245										
\$TYPEC 013400	3184	3186	3189#													
\$TYPEX 013446	3398#	3477														
\$TYPDC 014352	3397	3400#	3479													
\$TYPDN 014366	3393#	3478														
\$TYPDS 014326	2152#															
\$XTSTR 006630	2082#															
\$GET4= 000000	3394*	3398*	3408	3443#												
\$OFILL 014551	2149	2234														
\$OCCAT= ***** U	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:C2TADD,DSKZ:C2TADD.SQ=DSKZ:C2TADD.P11  
RUN-TIME: 20 14 1 SECONDS  
RUN-TIME RATIO: 213/36=5.8  
CORE USED: 23K (45 PAGES)

DOCUMENT PAGES: 97