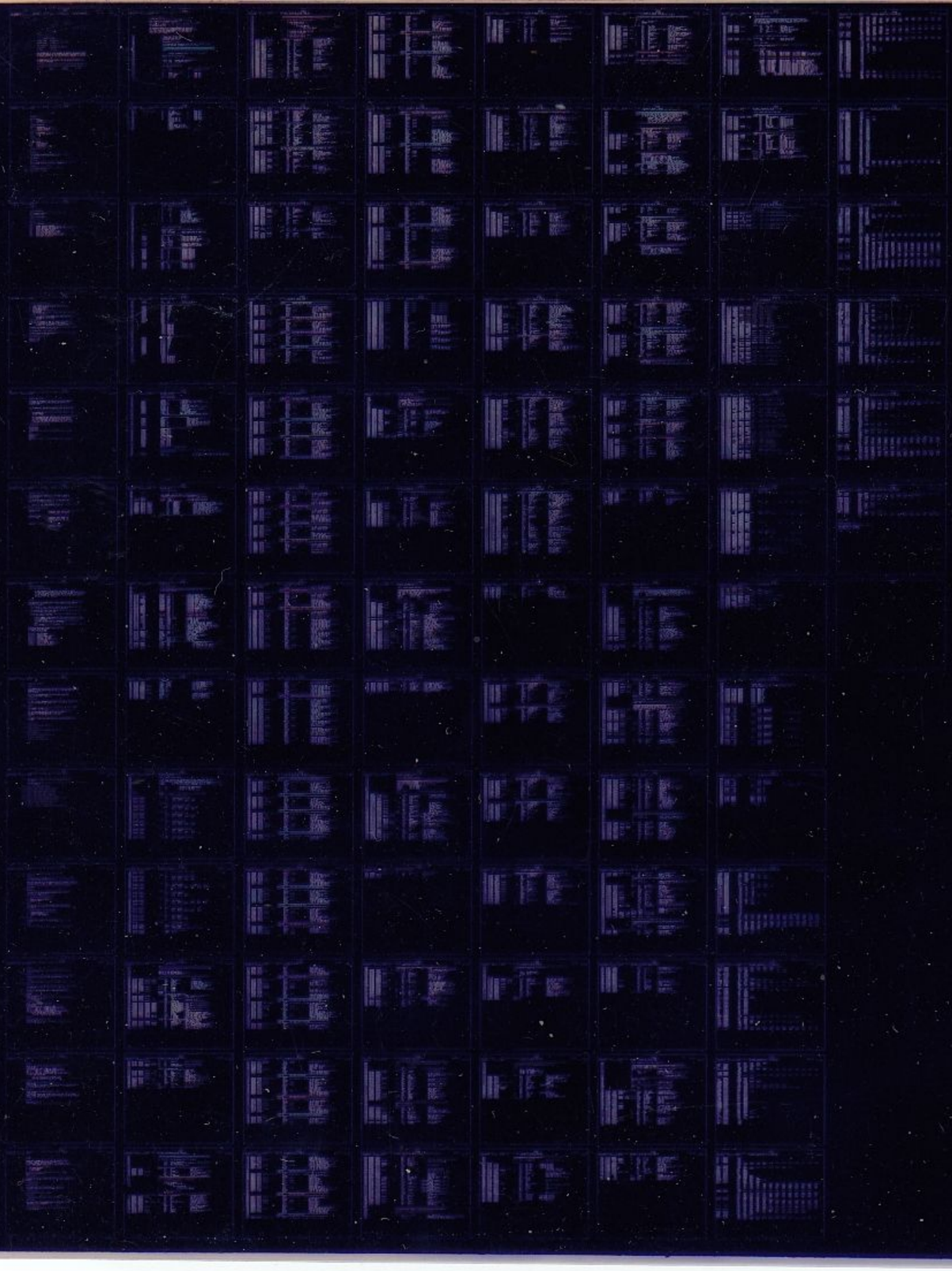
TU60

TA11 MOTION TEST
CZTADD0

AH-9358D-MC

FICHE 1 OF 1

JUN 1978
digital
MADE IN USA



EOF!CZR!IFSEQ 00010000 780519 PDP10 411 0 HDR!CZTADDSEQ 0001C000 780519
SEQ 000@DP10 411 TA11 MCTION TEST CZTAD-D MACY11 30A(1052) 23-MAR-78 08:36 PAGE 1
CZTADD P11 23-MAR-78 08:34 SEQ 0001

.REM @

IDENTIFICATION

PRODUCT CODE: AC-9357D-MC
PRODUCT NAME: CZTADDO TA11 MO'!ON TST
DATE REVISED: FEB 1978
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: JIM LACEY

CONTENTS

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 CONTINOUS BLOCKS OF DATA
230 READ CONTINOUS 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 RESPONSES 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              ;OPERATOR TYPES "2242""CARRIAGE RETURN "(CR)"
LOOP PC?                    ;OPERATOR TYPES "CR" WHICH
                            ;IMPLIES "LOOP PC"="TEST PC"

;OPERATOR TYPES "↑C"
↑TEST PC? 3000              ;OPERATOR TYPES "3000""CR"
LOOP PC? 3020               ;OPERATOR TYPES "3020""CR"
                            ;PROGRAM USES THIS AS THE
                            ;FIRST ADDRESS OF THE SCOPE
                            ;LOOP

;OPERATOR TYPES "↑C"
↑TEST PC?                   ;OPERATOR TYPES "CR"
                            ;PROGRAM CONTINUES
                            ;FROM THIS POINT
```

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

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

CONTROL:

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

1) TYPE CONTROL G (↑G); THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.

2) THE MACHINE WILL THEN TYPE: SWR=XXXXXXNEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)

3) AFTER THE ''NEW='' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:

   A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A (CR). (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED)
   IF A (CR) IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.

   B) IF A CONTROL U (↑U) IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

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

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

5.2 SUBROUTINE ABSTRACTS

5.2.1 SCOPE

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

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

5.2.2 TRAPCATCHER

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

5.2.3 ERROR

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

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

5.2.4 TRAP

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

5.2.4.1 TYPE ($TYPE)

TYPE AN ASCIZ STRING ON THE TTY

5.2.4.2 RDCHR ($RDCHR)

READ A SINGLE ASCII CHARACTER FROM THE TTY

5.2.4.3 RDLIN ($RDLIN)

READ AN ASCII STRING FROM THE TTY

5.2.4.4 SETLOOP (T.SETLOOP)

SETUP TO LOOP AS PER THE TTY

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

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

5.2.5.1 WFG (T.WFG) WRITE A FILE GAP

5.2 5.2 WRITE (T.WRIT) WRITE A BLOCK OF DATA

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

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

5.2.5 5 BSBG (T.BSBG) BACK SPACE A BLOCK GAP

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

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

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

5.2.5.9 SELDRV (T.SELDRV) SELECT A DRIVE

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

5.2.6 THE FOLLOWING SUBROUTINES ARE CALLED BY A JSR

5.2.6.1 DO.CMD

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

WHEN THE FUNCTION IS TO BE PERFORMED WITH "INTERRUPT ENABLE"=1 THE TA11 VECTOR IS SET TO "SERV". WHEN THE FUCTION IS TO BE PERFORMED WITH "INTERRUPT ENABLE"=0 THE VECTOR IS SET TO "BADINT"

NOTE SWR<5> PROVIDES OVERRIDE CAPABLITIES 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. "TRANFER REQUEST"=0 AND "READY"=0
2. "TRANFER REQUEST"=1
3. "READY"=1 AND "ERROR"=0
4. "READY"=1 AND "ERROR"=1

5.2.6.3 DO.CRC

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

5.2.6.4 A2OCT

THIS ROUTINE CHANGES AN ASCII STRING TO AN OCTAL NUMBER.

5.2.6.5 GETDRV

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

5.2.6.6 ASKADR

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

5.2.6.7 TYPERR

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

5.2.6.8 EXAM

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

5.2.7 THE FOLLOW ROUTINES ARE USED TO MAKE ADJUSTMENTS TO
THE 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 MON". THE THRESHOLD POT"
AND THE "TAPE BLANK MONO".

5.2.7.6 SFFGSB

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

5.2.7.7 BSFGSB

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

6. ERRORS

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

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

REFER TO THE LISTING UNDER SCRATB 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 APPOXIMATELY 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
564                     ;*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                     .REM!
596
597                             GENERAL INFORMATION ABOUT THE TA11/TU60 CASSETTE
598
599                     ADDRESS MNEMONIC DESCRIPTION
600                     ----------------------------
601
602
603                     777500  TACS    CONTROL AND STATUS RECISTER
604                     777502  TADB    DATA BUFFER REGISTER
605                        260  TAVEC   INTERRUPT VECTOR
606
607
608                     TACS REGISTER DESCRIPTION
609                     -------------------------
610                                                     INIT
611                     BIT     NAME                    STATE   READ AND/OR WRITE?
612                     ---     ----                    -----   -----------------
613
614                     15      ERROR                   ?       READ ONLY
615                     14      BLOCK CHECK ERROR       0       READ ONLY
616                     13      C EAR LEADER            ?       READ ONLY
617                     12      WRITE LOCK              ?       READ ONLY
618                     11      FILE GAP                0       READ ONLY
619                     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'

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


                        .SBTTL  BASIC DEFINITIONS

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

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

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

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
```

```
                        
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7
                        
                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0
                        
                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
```

```
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

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

                        ;**********TA11 FUNCTIONS**********
000000                  XWFG=   0                       ;WRITE FILE GAP FUNCTION
000002                  XWRITE= 2                       ;WRITE FUNCTION
000004                  XREAD=  4                       ;READ FUNCTION
000006                  XBSFG=  6                       ;BACK SPACE FILE GAP FUNCTION
000010                  XBSBG=  10                      ;BACK SPACE BLOCK GAP FUNCTION
000012                  XSFFG=  12                      ;SPACE FWD FILE GAP FUNCTION
000014                  XSFBG=  14                      ;SPACE FWD BLOCK GAP FUNCTION
000016                  XRWND=  16                      ;REWIND FUNCTION

                        ;**********TA11 BIT ASSIGNMENT**********
100000                  ERROR=  BIT15
040000                  CRCERR= BIT14
020000                  LEADER= BIT13
010000                  WRTLOCK=BIT12
004000                  FGAP=   BIT11
002000                  TIMERR= BIT10
001000                  OFFLINE=BIT09
000400                  UNIT=   BIT08
000200                  TR.REQ= BIT07
000100                  INT.EN= BIT06
000040                  READY=  BIT05
000020                  ILBS=   BIT04
000010                  FUNC2=  BIT03
000004                  FUNC1=  BIT02
000002                  FUNC0=  BIT01
000001                  GO=     BIT00
000016                  FUNCTION=       FUNC2+FUNC1+FUNC0

                        ;////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////
                        ;SPECIAL REGISTERS

000004                  TACS=   %4                      ;R4 IS USED AS A POINTER TO THE TACS REGISTER
000005                  TADB=   %5                      ;R5 IS USED AS A POINTER TO THE TADB REGISTER.
                        ;////////////////////////////////////////////////////////////
```

```
                                .SBTTL  TRAP CATCHER

                        000000          .=0
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                        000174          .=174
000174  000000          DISPREG:  .WORD  0              ;;SOFTWARE DISPLAY REGISTER
000176  000000          SWREG:    .WORD  0              ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  001450          JMP     @#BEGIN1        ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  001502          JMP     @#BEGIN2        ;SELECT DRIVE(S) BEFORE STARTING TEST
000210  000137  001510          JMP     @#BEGIN3        ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
000214  000137  002036          JMP     @#BEGINX        ;SETUP FOR MANUAL LOOPING
000220  000137  012364          JMP     @#WFGSUB        ;WRITE FILE GAP FROM BOT TO EOT
000224  000137  012440          JMP     @#WRTSUB        ;WRITE CONTINUOUS BLOCKS OF DATA
000230  000137  012522          JMP     @#RDSUB         ;READ CONTINUOUS BLOCKS OF DATA
000234  000137  012610          JMP     @#WGPBLK        ;WRITE FILE GAP AND A BLOCK OF DATA
000240  000137  012702          JMP     @#RGPBLK        ;READ BLOCK OF DATA AND INTO A FILE GAP
000244  000137  013000          JMP     @#SFFGSB        ;SPACE FWD FILE GAP FROM BOT TO EOT
000250  000137  013054          JMP     @#BSFGSB        ;BACK SPACE FILE GAPS
                        ;;**************************************************
```

```
                                .SBTTL  COMMON TAGS

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

                        001100                  .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102      000         $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103      000         $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114      000         $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115      001         $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134      000         $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135      000         $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY:.WORD   DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154      000         $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155      002         $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156      012         $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157      000         $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000          $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001162  000000          $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164  000000          $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001166  000000          $TMP0:  .WORD   0               ;;USER DEFINED
001170  000000          $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001172  000000          $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001174  177607  000377  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001200      077         $QUES:  .ASCII  /?/             ;;QUESTION MARK
001201      015         $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001202  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                        ;;*****************************************************************
001204  000000          SAVPC:  .WORD   0               ;STORAGE FOR THE PC
001206  000000          BYTNUM: .WORD   0               ;THE NUMBER OF BYTES "READ" OR "WRITTEN
001210  000000          RCRC0:  .WORD   0               ;CRC CALCULATED FOR "WRITE"
001212  000000          RCRC1:  .WORD   0               ;CRC CALCULATED FOR "READ"
001214  000000          RCRC2:  .WORD   0               ;CRC FROM THE TA11 FOR "READ"

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           TAPRIO: 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:        0       ;WAIT ON INTERRUPT KEY
  895  001256  001260           RWDFLG: RWDA            ;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
```

```
                                .SBTTL  ERROR POINTER TABLE

                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT


001270                          $ERRTB:
                                ;NOTE:  ALL NUMBERS WILL BE TYPED AS 6 DIGIT OCTAL NUMBERS

                                ;ITEM   1
001270  015540                  EM1     ;IMPROPER FLAG SETTING
001272  016102                  DH1     ;TEST   ERROR
                                        ;PC     PC      TACS    TADB
001274  015044                  DT1     ;SAVPC  $ERRPC  $REG0   $REG1
001276  000000                  0

                                ;ITEM   2
001300  015566                  EM2     ;IMPROPER FLAG OCCURRED
001302  016102                  DH1     ;TEST   ERROR
                                        ;PC     PC      TACS    TADB
001304  015044                  DT1     ;SAVPC  $ERRPC  $REG0   $REG1
001306  000000                  0

                                ;ITEM   3
001310  015615                  EM3     ;MISSED A FLAG
001312  016102                  DH1     ;TEST   ERROR
                                        ;PC     PC      TACS    TADB
001314  015044                  DT1     ;SAVPC  $ERRPC  $REG0   $REG1
001316  000000                  0

                                ;ITEM   4
001320  015633                  EM4     ;INTERRUPT FAILED
001322  016102                  DH1     ;TEST   ERROR
                                        ;PC     PC      TACS    TADB
001324  015044                  DT1     ;SAVPC  $ERRPC  $REG0   $REG1
001326  000000                  0

                                ;ITEM   5
001330  015654                  EM5     ;PREMATURE READY OCCURRED
001332  016102                  DH1     ;TEST   ERROR
                                        ;PC     PC      TACS    TADB
001334  015044                  DT1     ;SAVPC  $ERRPC  $REG0   $REG1
001336  000000                  0

                                ;ITEM   6
001340  015705                  EM6     ;DIDN'T STOP IN A FILE GAP
001342  016102                  DH1     ;TEST   ERROR
                                        ;PC     PC      TACS    TADB
```

```
  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           EXPT'D  RCV'D   BYTE
  968                                             ;PC     PC      TACS    DATA    DATA    NUMBER
  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           WRITE   READ    TU60
  982                                             ;PC     PC      TACS    CRC     CRC     CRC
  983  001404  015074                    DT3     ;SAVPC  $ERRPC  $REGO   RCRC0   RCRC1   RCRC2
  984  001406  000000                    0
  985
  986                          ;ITEM   13
  987  001410  016047                    EM13    ;SHORT RECORD
  988  001412  016461                    DH4     ;TEST   ERROR                           BYTES
  989                                             ;PC     PC      TACS    TADB    LEFT
  990  001414  015112                    DT4     ;SAVPC  $ERRPC  $REGO   $REG1   $TMPO
  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  $REGO
  998  001426  000000                    0
  999
 1000  001430                  ITEMS2:           ;ITEMS 201-202
 1001
 1002  001430  016664                    EM201   ;TA11 FAILED TO RESPOND
 1003  001432  016736                    DH201   ;PC     TACS
 1004  001434  016652                    DT201   ;$ERRPC TACS
 1005  001436  000000                    0       ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS
 1006
 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     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1048  001600  012737  000340  000036           MOV     #340,@#TRAPVEC+2;LEVEL 7
1049  001606  012737  014662  000024           MOV     #$PWRDN,@#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$,@#ERRVEC   ;;SET UP ERROR VECTOR
1061  001674  012737  177570  001140           MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
1062  001702  012737  177570  001142           MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
1063  001710  023727  177777  177222           CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
1064  001716  001012                           BNE     66$             ;;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     #$ENDAD,@#42    ;;PCT-11?
1078  001764  001424                          BEQ     HERE            ;;BRANCH IF YES
1079  001766  104401  002024                  TYPE    ,MSGID          ;;TYPE ASCIZ STRING
1080                                  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1081  001772  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
1082  001776  001006                          BNE     67$             ;;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$:
1089  002022  000405                          BR      HERE            ;;GET OVER THE ASCIZ
1090                                  ;;MSGID:        .ASCIZ  <CRLF>/CZTAD-D/<CRLF>
1091  002036                          HERE:
```

```
                        ;;*****************************************************************
                        ;;*****************************************************************
                        ;
                        ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
                        ;
                        ;       R5=3    MANUAL OPERATIONS
                        ;       R5=2    ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
                        ;       R5=1    ASK FOR DRIVE(S)
                        ;       R5=0    DON'T ASK FOR ANYTHING
                        ;
                        ;*****************************************************************
002036  010504          BEGINX: MOV     R5,R4                   ;COPY R5
002040  005305                  DEC     R5                      ;ASK FOR DRIVES?
002042  002406                  BLT     CHKADR                  ;BR IF NO
002044  004737  007422          JSR     PC,@#ASKDRV             ;GO GET DRIVES TO BE TESTED
002050  005305                  DEC     R5                      ;ASK FOR ADDRESSES?
002052  002402                  BLT     CHKADR                  ;BR IF NO
002054  004737  007532          JSR     PC,@#ASKADR             ;GO GET TA11 ADDRESSES
                        ;;*****************************************************************
                        ;;*****************************************************************
                        ;
                        ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
                        ;
                        ;I.     TIMEOUT OCCURRED
                        ;       A. TYPE ERROR MESSAGE
                        ;       B. EXAMINE R4
                        ;               1. R4>0 GOTO BEGINX
                        ;               2. R4=0 EXAMINE (42)
                        ;                       A. (42)=0 GOTO BEGINX
                        ;                       B. (42)>0 GOTO $ENDAD
                        ;
                        ;II.    TIMEOUT DIDN'T OCCUR
                        ;       A. CONTINUE
                        ;
                        ;*****************************************************************
002060  012737  002076  000004  CHKADR: MOV     #1$,@#ERRVEC            ;IN CASE C- TIMEOUTS
002066  005000                  CLR     R0                      ;USE AS A SWITCH
002070  005777  177124          TST     @TACSL                  ;SEE IF TA11 RESPONDS
002074  000402                  BR      2$                      ;BR IF NO TIMEOUT
002076  005200          1$:     INC     R0                      ;COME HERE ON TIMEOUT
002100  022626                  CMP     (SP)+,(SP)+             ;CLEANUP THE STACK
002102  012737  000006  000004  2$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TIMEOUT VECTOR
002110  003700                  TST     R0                      ;DID A TIMEOUT OCCUR?
002112  001412                  BEQ     3$                      ;BR IF NO
002114  104201                  ERROR   201                     ;TA11 FAILED TO RESPOND
002116  012705  000002          MOV     #2,R5                   ;DRIVES & ADDRESSES
002122  005704                  TST     R4                      ;OPERATOR INPUTS?
002124  001344                  BNE     BEGINX                  ;BR IF YES
002126  013700  000042          MOV     @#42,R0                 ;GET MONITOR RETURN ADDRESS
002132  001741                  BEQ     BEGINX                  ;BR IF NO MONITOR
002134  000137  006432          JMP     @#$ENDAD                ;GO TO END
002140                  3$:
```

```
                        ;;*****************************************************************
                        ;;*****************************************************************
                        ;
                        ;MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED
                        ;
                        ;I.      DESIRED DRIVES CAN NOT BE TESTED
                        ;        A. TYPE ERROR MESSAGE
                        ;        B. EXAMINE R4
                        ;                1. R4>0 GOTO BEGINX
                        ;                2. R4=0 EXAMINE (42)
                        ;                        A. (42)=0 GOTO BEGINX
                        ;                        B. (42)>0 GOTO $ENDAD
                        ;
                        ;II.     BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
                        ;        A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
                        ;        B. CONTINUE IN PROGRAM
                        ;
                        ;III.    DESIRED DRIVE(S) CAN BE TESTED
                        ;        A. CONTINUE IN PROGRAM
                        ;
                        ;*****************************************************************
002140  012700  001240  CHKDRV: MOV     #DRVKEY,R0          ;PICKUP ADDRESS OF ASCII DRIVE KEY
002144  004737  006464          JSR     PC,@#EXAM           ;GO EXAMINE FIRST DRIVE
002150  000410                  BR      1$                  ;OK TO TEST---GO CHECK NEXT
002152  116010  000001          MOVB    1(R0),(R0)          ;REPLACE 1ST WITH 2ND
002156  001412                  BEQ     2$                  ;BR IF NO 2ND DRIVE SELECTED
002160  004737  006464          JSR     PC,@#EXAM           ;GO EXAMINE DRIVE
002164  000407                  BR      2$                  ;OK TO TEST
002166  005010                  CLR     (R0)                ;CLEAR DRIVE CODES
002170  000405                  BR      2$
002172  005200          1$:     INC     R0                  ;POINT TO 2ND
002174  004737  006464          JSR     PC,@#EXAM           ;GO EXAMINE DRIVE
002200  000401                  BR      2$                  ;OK TO TEST
002202  105010                  CLRB    (R0)                ;CLEAR 2ND
002204  012700  001240  2$:     MOV     #DRVKEY,R0          ;RESET ADDRESS POINTERS
002210  010037  001244          MOV     R0,@#DRVPNT
002214  121060  000001          CMPB    (R0),1(R0)          ;1ST = 2ND?
002220  001002                  BNE     3$                  ;BR IF NO
002222  105060  000001          CLRB    1(R0)               ;YES---CLEAR 2ND
002226  005710          3$:     TST     (R0)                ;ANY DRIVES?
002230  001401                  BEQ     5$                  ;BR IF NO
002232  000412                  BR      MANUAL
002234  104202          5$:     ERROR   202                 ;NO DRIVES AVAILABLE
002236  012705  000002          MOV     #2,R5               ;DRIVES & ADDRESS
002242  005704                  TST     R4                  ;OPERATOR INPUTS?
002244  001274                  BNE     BEGINX              ;BR IF YES
002246  013700  000042          MOV     @#42,R0             ;GET MONITOR RETURN ADDRESS
002252  001671                  BEQ     BEGINX              ;NO MONITOR
002254  000137  006432          JMP     @#$ENDAD            ;GO TO END
002260  020427  000003  MANUAL: CMP R4,#3
002264  001002                  BNE     OK
002266  013704  177777          MOV     -1,R4
002272  010437  001246  OK:     MOV     R4,@#ASKKEY
002276  000405                  BR      START
002300  104401  002024  PWRST:  TYPE    ,MSGID              ;POWER FAIL RESTART
002304  012737  001240  001244  MOV     #DRVKEY,@#DRVPNT
```

```
1200  002312  012777  013356  176710  START:  MOV     #BADINT,@TAVEC          ;SETUP TA11 TRAP VECTOR
1201  002320  013777  001234  176704          MOV     @#TAPRIO,@TAVEC+2
1202  002326  012737  000340  177776          MOV     #340,@#PS               ;LOCKOUT ALL I/O INT
1203  002334  013704  001 20                  MOV     @#TACSL,TACS            ;SETUP TACS
1204  002340  013705  001[illegible]          MOV     @#TADBL,TADB            ;SETUP TADB
1205  002344  005037  001102                  CLR     @#$TSTNM                ;ZERO THE TEST NUMBER
1206  002350  005037  006614                  CLR     @#LOOPKEY               ;CLEAR THE LOOP UNDER TTY CONTROL KEY
1207  002354  012737  000001  001216          MOV     #1,@#$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                  MOVB    (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     @#$MXCNT                ;SET FOR TWO ITERATIONS PER TEST (ONE PER DRIVE)
1220  002430  005201                          INC     R1                      ;INCREMENT THE POINTER
1221  002432  111137  001250                  MOVB    (R1),@#CURDRV           ;SET SECOND DRIVE FOR TYPE-OUT
1222  002436  001004                          BNE     1$                      ;GO TYPE IT
1223  002440  012701  001240                  MOV     #DRVKEY,R1              ;RESET POINTER
1224  002444  111137  001250                  MOVB    (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$,@#$LPADR            ;SETUP THE SCOPE LOOP ADDRESS
1229  002472  005737  001100                  TST     @#$PASS                 ;ONCE ONLY
1230  002476  001131                          BNE     TST1
1231                                          ;////////////////////////////////////////////////////////////
1232                                          ;ON THE FIRST PASS OF THE PROGRAM DETERMINE THE AMOUNT OF TIME TO
1233                                          ;WAIT ON A FLAG ("READY"/"TRANSFER REQUEST") AND THE STALL TIME TO USE
1234                                          ;WHEN STALLING AFTER A FUNCTION IS COMPLETED
1235  002500  104422                          SELDRV                          ;SELECT DRIVE
1236  002502  012737  002702  001172          MOV     #FATAL,@#$ESCAPE        ;SETUP ESCAPE ON ERROR
1237  002510  005077  176542                  CLR     @RWDFLG                 ;SET FOR NO REWIND ERRORS
1238  002514  012737  077777  012012          MOV     #^CBIT15,@#MAXCNT       ;SETUP IMPOSSIBLE COUNT
1239  002522  112714  000016                  MOVB    #XRWND,@^ACS            ;INSURE NO ERROR DUE TO "CLEAR LEADER"
1240  002526  104421                          REWIND                          ;GO TO BOT
1241  002530  10441[illegible]                WFG                             ;NOW TIME A WFG
1242  002532  163737  012010  012012          SUB     @#HICNT,@#MAXCNT        ;GET THE TIME IT TOOK TO WFG
1243  002540  005237  012012                  INC     @#MAXCNT                ;MAKE IT BIGGER
1244  002544  006137  012012                  ROL     @#MAXCNT
1245  002550  006137  012012                  ROL     @#MAXCNT
1246  002554  005000                          CLR     R0
1247  002556  005001                          CLR     R1
1248  002560  012777  002616  176442          MOV     #4$,@TAVEC              ;SETUP TA11 VECTOR
1249  002566  112714  000101                  MOVB    #XWFG!INT.EN!GO,@TACS   ;START A "WFG"
1250  002572  005037  177776                  CLR     @#PS                    ;ALLOW INTERRUPTS
1251  002576  162700  000001          3$:     SUB     #1,R0                   ;START TIMING HOW LONG
1252  002602  005601                          SBC     R1                      ;    IT TAKES TO WRITE A FILE GAP
1253  002604  001374                          BNE     3$
1254  002606  0[illegible]737  000340  177776          MOV     #340,@#PS       ;LOCK OUT INTERRUPTS
1255  002614  0[illegible]432                  BR      FATAL                   ;IT TOOK TO LONG
```

```
1256  002616  006201                    4$:   ASR     R1                      ;DIVIDE THE TIME BY 2
1257  002620  006000                          ROR     R0
1258  002622  010037  001264                  MOV     R0,@#STALL              ;AND SAVE IT AS THE STALL TIME
1259  002626  010137  001266                  MOV     R1,@#STALL+2
1260  002632  012777  012356  176370          MOV     #BADINT,@TAVEC          ;SET TRAP CATCHER
1261  002640  105737  001241                  TSTB    @#DRVKEY+1              ;TWO DRIVES?
1262  002644  001407                          BEQ     5$                      ;BR IF NO
1263  002646  104422                          SELDRV                          ;SELECT THE OTHER DRIVE
1264  002650  005077  176402                  CLR     @RWDFLG                 ;SET REWIND FLAG
1265  002654  112714  000016                  MOVB    #XRWND,@TACS            ;AVOID "CLEAR LEADER" ERROR
1266  002660  104421                          REWIND                          ;GO TO BOT
1267  002662  104412                          WFG                             ;GO TO OXIDE
1268  002664  005737  001246          5$:     TST     @#ASKKEY                ;GO START TESTING IF NO MANUAL
1269  002670  002034                          BGE     TST1                    ;  OPERATIONS REQUESTED
1270  002672  005000                          CLR     R0
1271  002674  006137  012012                  ROL     @#MAXCNT                ;INCREASE THE WAIT TIME
1272  002700  000000                          HALT                            ;GIVE CONTROL TO THE OPERATOR
1273
1274  002702                          FATHL:
1275  002702  104401  002710                  TYPE    ,65$                ;;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,R0                 ;CHECK FOR A MONITOR
1281  002750  001402                          BEQ     1$                      ;BR IF NONE
1282  002752  000137  006432                  JMP     @#$ENDAD                ;LEAVE
1283  002756  000000                  1$:     HALT                            ;ERROR OCCURRED BEFORE TESTING STARTED
1284  002760  000776                          BR      1$                      ;HANGUP
```

```
                                        ;;*****************************************************************
                                        ;*TEST 1         REWIND,WRITE,REWIND,READ
                                        ;;*****************************************************************
003016  000004                  TST1:   SCOPE
003020  012737  003052  001172          MOV     #TST2,$ESCAPE   ;;ESCAPE TO TEST 2 ON ERROR
003026  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
003030  104421                          REWIND                  ;GO TO "CLEAR LEADER"
003032  104413  017012                  WRITE   ,WLNK1          ;WRITE ON TAPE
003036  104421                          REWIND                  ;GO TO BEGINNING OF TAPE
003040  104414  016756                  READ    ,RLNK1          ;READ
003044  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003046  016756                          RLNK1                   ;LINK TO THE READ BUFFER
003050  017012                          WLNK1                   ;LINK TO THE WRITE BUFFER
                                        ;;*****************************************************************
                                        ;*TEST 2         TEST BSBG AFTER WRITE AND READ AFTER BSBG
                                        ;;*****************************************************************
003052  000004                  TST2:   SCOPE
003054  012737  003106  001172          MOV     #TST3,$ESCAPE   ;;ESCAPE TO TEST 3 ON ERROR
003062  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
003064  104421                          REWIND                  ;GO TO "BOT"
003066  104413  017012                  WRITE   ,WLNK1          ;WRITE ONE BLOCK
003072  104415                          BSBG                    ;BACK OVER THE BLOCK
003074  104414  016756                  READ    ,RLNK1          ;NOW READ
003100  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003102  016756                          RLNK1                   ;LINK TO THE READ BUFFER
003104  017012                          WLNK1                   ;LINK TO THE WRITE BUFFER
                                        ;;*****************************************************************
                                        ;*TEST 3         TEST BSFG AFTER WRITE AND READ AFTER BSFG
                                        ;;*****************************************************************
003106  000004                  TST3:   SCOPE
003110  012737  003156  001172          MOV     #TST4,$ESCAPE   ;;ESCAPE TO TEST 4 ON ERROR
003116  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
003120  104421                          REWIND                  ;GO TO "BOT"
003122  104413  017012                  WRITE   ,WLNK1          ;WRITE ONE BLOCK
003126  104416                          BSFG                    ;BACK OVER THE BLOCK
003130  104414  016756                  READ    ,RLNK1          ;NOW READ
003134  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003136  016756                          RLNK1                   ;LINK TO THE READ BUFFER
003140  017012                          WLNK1                   ;LINK TO THE WRITE BUFFER
003142  104416                          BSBG                    ;BACK UP
003144  104414  016756                  READ    ,RLNK1          ;READ
003150  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003152  016756                          RLNK1                   ;LINK TO THE READ BUFFER
                                        ;;*****************************************************************
                                        ;*TEST 4         TEST BSBG AFTER READ
                                        ;;*****************************************************************
003154  000004                  TST4:   SCOPE
003156  012737  003202  001172          MOV     #TST5,$ESCAPE   ;;ESCAPE TO TEST 5 ON ERROR
003164  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
003166  104421                          REWIND                  ;GO TO "BOT"
003170  104413  017012                  WRITE   ,WLNK1          ;WRITE ONE BLOCK
003174  104421                          REWIND                  ;GO TO "BOT"
003176  104414  016756                  READ    ,RLNK1          ;READ
003202  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003204  016756                          RLNK1                   ;LINK TO THE READ BUFFER
003206  017012                          WLNK1                   ;LINK TO THE WRITE BUFFER
003210  104416                          BSBG                    ;BACK UP
003212  104414  016756                  READ    ,RLNK1          ;READ
003216  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003220  016756                          RLNK1                   ;LINK TO THE READ BUFFER
```

```
1341  003154  017012                          WLNK1                           ;LINK TO THE WRITE BUFFER
1342                                  ;;*****************************************************************
1343                                  ;*TEST 5         TEST BSFG AFTER READ
1344                                  ;;*****************************************************************
1345  003156  000004                  TST5:   SCOPE
1346  003160  012737  003226  001172          MOV     #TST6,$ESCAPE   ;;ESCAPE TO TEST 6 ON ERROR
1347  003166  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
1348  003170  104421                          REWIND                  ;GO TO "BOT"
1349  003172  104413  017012                  WRITE   ,WLNK1          ;WRITE A BLOCK
1350  003176  104421                          REWIND                  ;GO TO "BOT"
1351  003200  104414  016756                  READ    ,RLNK1          ;READ A BLOCK
1352  003204  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
1353  003206  016756                          RLNK1                   ;LINK TO THE READ BUFFER
1354  003210  017012                          WLNK1                   ;LINK TO THE WRITE BUFFER
1355  003212  104415                          BSFG                    ;BACK OVER THE DATA
1356  003214  104414  016756                  READ    ,RLNK1          ;NOW READ THE DATA
1357  003220  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
1358  003222  016756                          RLNK1                   ;LINK TO THE READ BUFFER
1359  003224  017012                          WLNK1                   ;LINK TO THE WRITE BUFFER
1360                                  ;;*****************************************************************
1361                                  ;*TEST 6         TEST SFBG AFTER REWIND AND BSFG AFTER SFBG
1362                                  ;;*****************************************************************
1363  003226  000004                  TST6:   SCOPE
1364  003230  012737  003266  001172          MOV     #TST7,$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                          BSFG                    ;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                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
1386  003316  0..756                          RLNK1                   ;LINK TO THE READ BUFFER
1387  003320  0..012                          WLNK1                   ;LINK TO THE WRITE BUFFER
1388  003322  10'.413  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,$ESCAPE  ;;ESCAPE TO TEST 11 ON ERROR
1399  003352  104422                                SELDRV                  ;SELECT DRIVE FOR TESTING
1400  003354  104421                                REWIND                  ;GO TO "CLEAR LEADER"
1401  003356  104413  017012                        WRITE   ,WLNK1          ;WRITE TWO(2) BLOCKS
1402  003362  104413  017016                        WRITE   ,WLNK2
1403  003366  104421                                REWIND                  ;GO TO BOT
1404  003370  104414  016756                        READ    ,RLNK1          ;READ BOTH BLOCKS
1405  003374  104414  016762                        READ    ,RLNK2
1406  003400  104423                                BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
1407  003402  016762                                RLNK2                   ;LINK TO THE READ BUFFER
1408  003404  017016                                WLNK2                   ;LINK TO THE WRITE BUFFER
1409                                        ;;**********************************************************************
1410                                        ;*TEST 11        TEST BSBG AFTER BSBG
1411                                        ;;**********************************************************************
1412  003406  000004                        TST11:  SCOPE
1413  003410  012737  003450  001172                MOV     #TST12,$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                                        ;*TEST 12        TEST READ AFTER SFBG
1426                                        ;;**********************************************************************
1427  003450  000004                        TST12:  SCOPE
1428  003452  012737  003512  001172                MOV     #TST13,$ESCAPE  ;;ESCAPE TO TEST 13 ON ERROR
1429  003460  104422                                SELDRV                  ;SELECT DRIVE FOR TESTING
1430  003462  104421                                REWIND                  ;GO TO "CLEAR LEADER"
1431  003464  104413  017012                        WRITE   ,WLNK1          ;WRITE TWO(2) BLOCKS
1432  003470  104413  017016                        WRITE   ,WLNK2
1433  003474  104421                                REWIND                  ;BOT
1434  003476  104420                                SFBG                    ;SPACE OVER THE 1ST BLOCK
1435  003500  104414  016762                        READ    ,RLNK2          ;READ THE 2ND BLOCK
1436  003504  104423                                BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
1437  003506  016762                                RLNK2                   ;LINK TO THE READ BUFFER
1438  003510  017016                                WLNK2                   ;LINK TO THE WRITE BUFFER
1439                                        ;;**********************************************************************
1440                                        ;*TEST 13        TEST SFBG AFTER BSBG
1441                                        ;;**********************************************************************
1442  003512  000004                        TST13:  SCOPE
1443  003514  012737  003556  001172                MOV     #TST14,$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                                  ;*TEST 14        TEST SFBG AFTER BSFG
1457                                  ;;**************************************************************
1458  003556  000004                  ↑ST14:  SCOPE
1459  003560  012737  003620  001172          MOV     #TST15,$ESCAPE  ;;ESCAPE TO TEST 15 ON ERROR
1460  003566  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1461  003570  104421                          REWIND                          ;GO TO "CLEAR LEADER"
1462  003572  104413  017012                  WRITE   ,WLNK1                  ;WRITE TWO BLOCKS
1463  003576  104413  017016                  WRITE   ,WLNK2
1464  003602  104415                          BSFG                            ;SPACE REV FILE
1465  003604  104420                          SFBG                            ;SPACE FWD BLOCK
1466  003606  104414  016762                  READ    ,RLNK2                  ;READ THE 2ND BLOCK
1467  003612  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1468  003614  016762                          RLNK2                           ;LINK TO THE READ BUFFER
1469  003616  017016                          WLNK2                           ;LINK TO THE WRITE BUFFER
1470                                  ;;**************************************************************
1471                                  ;*TEST 15        REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER WRITE
1472                                  ;;**************************************************************
1473  003620  000004                  ↑ST15:  SCOPE
1474  003622  012737  004012  001172          MOV     #TST16,$ESCAPE  ;;ESCAPE TO TEST 16 ON ERROR
1475  003630  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1476  003632  104421                          REWIND                          ;GO TO "CLEAR LEADER"
1477  003634  104413  017012                  WRITE   ,WLNK1                  ;WRITE 2 BLOCKS
1478  003640  104413  017016                  WRITE   ,WLNK2
1479  003644  104416                          BSBG                            ;BACK OVER BOTH OF THEM
1480  003646  104416                          BSBG
1481  003650  104414  016756                  READ    ,RLNK1                  ;READ THE FIRST ONE
1482  003654  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1483  003656  016756                          RLNK1                           ;LINK TO THE READ BUFFER
1484  003660  017012                          WLNK1                           ;LINK TO THE WRITE BUFFER
1485  003662  104414  016762                  READ    ,RLNK2                  ;NOW READ THE 2ND ONE
1486  003666  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1487  003670  016762                          RLNK2                           ;LINK TO THE READ BUFFER
1488  003672  017016                          WLNK2                           ;LINK TO THE WRITE BUFFER
1489  003674  104416                          BSBG                            ;BACK OVER THE 2ND ONE
1490  003676  104413  017022                  WRITE   ,WLNK3                  ;WRITE OVER THE 2ND BLOCK
1491  003702  104416                          BSBG                            ;NOW BACK UP AND
1492  003704  104414  016766                  READ    ,RLNK3                  ;READ IT
1493  003710  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1494  003712  016766                          RLNK3                           ;LINK TO THE READ BUFFER
1495  003714  017022                          WLNK3                           ;LINK TO THE WRITE BUFFER
1496  003716  104416                          BSBG                            ;BACK OVER IT AGAIN
1497  003720  104413  017026                  WRITE   ,WLNK4                  ;AND WRITE OVER IT AGAIN
1498  003724  104416                          BSBG                            ;BACK UP AND
1499  003726  104414  016772                  READ    ,RLNK4                  ;READ THE BLOCK
1500  003732  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1501  003734  016772                          RLNK4                           ;LINK TO THE READ BUFFER
1502  003736  017026                          WLNK4                           ;LINK TO THE WRITE BUFFER
1503  003740  104416                          BSBG                            ;BACK OVER THE BLOCK AGAIN
1504  003742  104413  017016                  WRITE   ,WLNK2                  ;AND WRITE OVER IT AGAIN
1505  003746  104416                          BSBG                            ;NOW BACK UP
1506  003750  104414  016762                  READ    ,RLNK2                  ;AND READ IT
1507  003754  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1508  003756  016762                          RLNK2                           ;LINK TO THE READ BUFFER
```

```
1509  003760  017016                        WLNK2                           ;LINK TO THE WRITE BUFFER
1510  003762  104416                        BSBG                            ;BACK OVER BOTH BLOCKS
1511  003764  104416                        BSBG
1512  003766  104414  016756                READ    ,RLNK1                  ;AND READ THE 1ST ONE
1513  003772  104423                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1514  003774  016756                        RLNK1                           ;LINK TO THE READ BUFFER
1515  003776  017012                        WLNK1                           ;LINK TO THE WRITE BUFFER
1516  004000  104414  016762                READ    ,RLNK2                  ;AND THE 2ND ONE
1517  004004  104423                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1518  004006  016762                        RLNK2                           ;LINK TO THE READ BUFFER
1519  004010  017016                        WLNK2                           ;LINK TO THE WRITE BUFFER
1520                                ;;*****************************************************************
1521                                ;*TEST 16       REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER READ
1522                                ;;*****************************************************************
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                        WLNK1                           ;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
```

```
                                        ;;*****************************************************************
                                        ;*TEST 17        TEST WFG AFTER WRITE AND BSFG AFTER WFG
                                        ;;*****************************************************************
004170  000004                          ↑ST17:  SCOPE
004172  012737  004226  001172                  MOV     #TST20,$ESCAPE  ;;ESCAPE TO TEST 20 ON ERROR
004200  104422                                 SELDRV                  ;SELECT DRIVE FOR TESTING
004202  104421                                 REWIND                  ;GO TO "BOT"
004204  104413  017012                         WRITE   ,WLNK1          ;WRITE ONE BLOCK
004210  104412                                 WFG                     ;FOLLOWED BY A FILE GAP
004212  104415                                 BSFG                    ;BACK OVER THE DATA BLOCK
004214  104414  016756                         READ    ,RLNK1          ;READ THE DATA
004220  104423                                 BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
004222  016756                                 RLNK1                   ;LINK TO THE READ BUFFER
004224  017012                                 WLNK1                   ;LINK TO THE WRITE BUFFER
                                        ;;*****************************************************************
                                        ;*TEST 20        TEST BSBG AFTER WFG
                                        ;;*****************************************************************
004226  000004                          ↑ST20:  SCOPE
004230  012737  004264  001172                  MOV     #TST21,$ESCAPE  ;;ESCAPE TO TEST 21 ON ERROR
004236  104422                                 SELDRV                  ;SELECT DRIVE FOR TESTING
004240  104421                                 REWIND                  ;GO TO BOT
004242  104413  017012                         WRITE   ,WLNK1          ;WRITE ONE BLOCK OF DATA
004246  104412                                 WFG                     ;WRITE A FILE GAP
004250  104416                                 BSBG                    ;BACK OVER THE DATA
004252  104414  016756                         READ    ,RLNK1          ;READ ONE BLOCK
004256  104423                                 BLKCMP                  ;COMPARE THE READ AND WRITE B FFERS
004260  016756                                 RLNK1                   ;LINK TO THE READ BUFFER
004262  017012                                 WLNK1                   ;LINK TO THE WRITE BUFFER
                                        ;;*****************************************************************
                                        ;*TEST 21        TEST WRITE AFTER WFG
                                        ;;*****************************************************************
004264  000004                          ↑ST21:  SCOPE
004266  012737  004342  001172                  MOV     #TST22,$ESCAPE  ;;ESCAPE TO TEST 22 ON ERROR
004274  104422                                 SELDRV                  ;SELECT DRIVE FOR TESTING
004276  104421                                 REWIND                  ;GO TO BOT
004300  104413  017012                         WRITE   ,WLNK1          ;WRITE ONE BOCK
004304  104421                                 REWIND                  ;GO TO BOT
004306  104414  016756                         READ    ,RLNK1          ;READ ONE BLOCK
004312  104423                                 BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
004314  016756                                 RLNK1                   ;LINK TO THE READ BUFFER
004316  017012                                 WLNK1                   ;LINK TO THE WRITE BUFFER
004320  104412                                 WFG                     ;WRITE A FILE GAP
004322  104413  017016                         WRITE   ,WLNK2          ;WRITE ANOTHER BLOCK
004326  104416                                 BSBG                    ;BACK OVER THE LAST BLOCK
004330  104414  016762                         READ    ,RLNK2          ;READ THE SECOND BLOCK
004334  104423                                 BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
004336  016762                                 RLNK2                   ;LINK TO THE READ BUFFER
004340  017016                                 WLNK2                   ;LINK TO THE WRITE BUFFER
                                        ;;*****************************************************************
                                        ;*TEST 22        TEST WFG AFTER READ AND BSFG AFTER WRITE
                                        ;;*****************************************************************
004342  000004                          ↑ST22:  SCOPE
004344  012737  004420  001172                  MOV     #TST23,$ESCAPE  ;;ESCAPE TO TEST 23 ON ERROR
004352  104422                                 SELDRV                  ;SELECT DRIVE FOR TESTING
004354  104421                                 REWIND                  ;GO TO "BOT"
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  004406  104414  016762                  READ    ,RLNK2          ;READ SECOND BLOCK
1630  004412  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
1631  004414  016762                          RLNK2                   ;LINK TO THE READ BUFFER
1632  004416  017016                          WLNK2                   ;LINK TO THE WRITE BUFFER
1633                                  ;;*******************************************************************
1634                                  ;*TEST 23        TEST SFFG AFTER REWIND AND READ AFTER SFFG
1635                                  ;;*******************************************************************
1636  004420  000004                  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  004560  104413  017022                  WRITE   ,WLNK3
```

```
                                        REWIND                          ;GO TO "BOT"
                                        SFFG                            ;SPACE OVER 1ST FILE
                                        SFBG                            ;SPACE OVER BLOCK
                                        BSBG                            ;BACK OVER THE BLOCK
                                        READ    ,RLNK2                  ;READ THE LAST BLOCK
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK2                           ;LINK TO THE READ BUFFER
                                        WLNK2                           ;LINK TO THE WRITE BUFFER
                                ;;*******************************************************
                                ;*TEST 26        TEST SFBG AFTER SFBG
                                ;;*******************************************************
                                TST26:  SCOPE
                                        MOV     #TST27,$ESCAPE  ;;ESCAPE TO TEST 27 ON ERROR
                                        SELDRV                          ;SELECT DRIVE FOR TESTING
                                        REWIND                          ;GO TO "BOT"
                                        WRITE   ,WLNK1                  ;WRITE A BLOCK OF DATA
                                        WFG                             ;WRITE A FILE GAP
                                        WRITE   ,WLNK2                  ;WRITE 3 BLOCKS OF DATA
                                        WRITE   ,WLNK3
                                        WRITE   ,WLNK4
                                        REWIND                          ;GO TO "BOT"
                                        SFFG                            ;SPACE OVER THE 1ST FILE
                                        SFBG                            ;SPACE OVER TWO BLOCK
                                        SFBG
                                        READ    ,RLNK4                  ;READ LAST BLOCK OF DATA
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK4                           ;LINK TO THE READ BUFFER
                                        WLNK4                           ;LINK TO THE WRITE BUFFER
                                ;;*******************************************************
                                ;*TEST 27        TEST BSFG AFTER SFBG AND BSFG AFTER BSFG
                                ;;*******************************************************
                                TST27:  SCOPE
                                        MOV     #TST30,$ESCAPE  ;;ESCAPE TO TEST 30 ON ERROR
                                        SELDRV                          ;SELECT DRIVE FOR TESTING
                                        REWIND                          ;GO TO "BOT"
                                        WRITE   ,WLNK1                  ;WRITE ONE BLOCK OF DATA
                                        WFG                             ;WRITE A FILE GAP
                                        WRITE   ,WLNK2                  ;WRITE A BLOCK OF DATA
                                        REWIND                          ;GO TO "BOT"
                                        SFFG                            ;SPACE OVER 1ST FILE
                                        SFBG                            ;SPACE OVER 1ST BLOCK OF 2ND FILE
                                        BSFG                            ;BACK TO BEGINNING OF 2ND FILE
                                        BSFG                            ;BACK TO BEGINNING OF 1ST FILE
                                        READ    ,RLNK1                  ;READ 1ST BLOCK OF 1ST FILE
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK1                           ;LINK TO THE READ BUFFER
                                        WLNK1                           ;LINK TO THE WRITE BUFFER
                                ;;*******************************************************
                                ;*TEST 30        TEST BSBG THRU A FILE GAP
                                ;;*******************************************************
                                TST30:  SCOPE
                                        MOV     #TST31,$ESCAPE  ;;ESCAPE TO TEST 31 ON ERROR
                                        SELDRV                          ;SELECT DRIVE FOR TESTING
                                        REWIND                          ;GO TO "BOT"
                                        WRITE   ,WLNK1                  ;WRITE ONE BLOCK OF DATA
                                        WFG                             ;WRITE A FILE GAP
```

```
1733  004762  104413  017016                  WRITE   ,WLNK2                  ;WRITE TWO BLOCKS OF DATA
1734  004766  104413  017022                  WRITE   ,WLNK3
1735  004772  104421                          REWIND                          ;GO TO "BOT"
1736  004774  104417                          SFFG                            ;SPACE OVER 1ST FILE
1737  004776  104420                          SFBG                            ;SPACE OVER 1ST BLOCK OF 2ND FILE
1738  005000  104416                          BSBG                            ;BACK OVER 1 BLOCK
1739  005002  104416                          BSBG                            ;BACK OVER 1 BLOCK
1740  005004  104414  016756                  READ    ,RLNK1                  ;READ
1741  005010  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1742  005012  016756                          RLNK1                           ;LINK TO THE READ BUFFER
1743  005014  017012                          WLNK1                           ;LINK TO THE WRITE BUFFER
1744                                  ;;*****************************************************************
1745                                  ;*TEST 31        TEST BSFG AFTER BSBG
1746                                  ;;*****************************************************************
1747  005016  000004                  TST31:  SCOPE
1748  005020  012737  005070  001172          MOV     #TST32,$ESCAPE  ;;ESCAPE TO TEST 32 ON ERROR
1749  005026  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1750  005030  104421                          REWIND                          ;GO TO "BOT"
1751  005032  104413  017012                  WRITE   ,WLNK1                  ;WRITE ONE BLOCK OF DATA
1752  005036  104412                          WFG                             ;WRITE A FILE GAP
1753  005040  104413  017016                  WRITE   ,WLNK2                  ;WRITE A BLOCK OF DATA
1754  005044  104421                          REWIND                          ;GO TO "BOT"
1755  005046  104417                          SFFG                            ;SPACE OVER FILE
1756  005050  104420                          SFBG                            ;SPACE OVER 1ST BLOCK OF 2ND FILE
1757  005052  104416                          BSBG                            ;BACK OVER BLOCK
1758  005054  104415                          BSFG                            ;BACK OVER FILE
1759  005056  104414  016756                  READ    ,RLNK1                  ;READ 1ST BLOCK ON TAPE
1760  005062  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1761  005064  016756                          RLNK1                           ;LINK TO THE READ BUFFER
1762  005066  017012                          WLNK1                           ;LINK TO THE WRITE BUFFER
1763                                  ;;*****************************************************************
1764                                  ;*TEST 32        TEST BSBG AFTER BSFG
1765                                  ;;*****************************************************************
1766  005070  000004                  TST32:  SCOPE
1767  005072  012737  005134  001172          MOV     #TST33,$ESCAPE  ;;ESCAPE TO TEST 33 ON ERROR
1768  005100  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1769  005102  104421                          REWIND                          ;GO TO "CLEAR LEADER"
1770  005104  104413  017012                  WRITE   ,WLNK1                  ;WRITE BLOCK
1771  005110  104412                          WFG                             ;WRITE A FILE GAP
1772  005112  104413  017016                  WRITE   ,WLNK2                  ;FOLLOW WITH ANOTHER BLOCK
1773  005116  104415                          BSFG                            ;BACK UP 1 FILE
1774  005120  104416                          BSBG                            ;BACK UP 1 BLOCK
1775  005122  104414  016756                  READ    ,RLNK1                  ;READ 1ST BLOCK ON TAPE
1776  005126  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1777  005130  016756                          RLNK1                           ;LINK TO THE READ BUFFER
1778  005132  017012                          WLNK1                           ;LINK TO THE WRITE BUFFER
1779                                  ;;*****************************************************************
1780                                  ;*TEST 33        TEST BSFG AFTER SFFG AND SFFG AFTER READ
1781                                  ;;*****************************************************************
1782  005134  000004                  TST33:  SCOPE
1783  005136  012737  005216  001172          MOV     #TST34,$ESCAPE  ;;ESCAPE TO TEST 34 ON ERROR
1784  005144  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1785  005146  104421                          REWIND                          ;GO TO "BOT"
1786  005150  104413  017012                  WRITE   ,WLNK1                  ;WRITE A BLOCK OF DATA
1787  005154  104412                          WFG                             ;WRITE A FILE GAP
1788  005156  104413  017016                  WRITE   ,WLNK2                  ;WRITE A BLOCK OF DATA
```

```
1789   005162  104421                          REWIND                  ;GO TO "BOT"
1790   005164  104417                          SFFG                    ;SPACE OVER 1ST FILE
1791   005166  104415                          BSFG                    ;BACK TO BEGINNING OF 1ST FILE
1792   005170  104414  016756                  READ    ,RLNK1          ;READ FIRST DATA BLOCK
1793   005174  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
1794   005176  016756                          RLNK1                   ;LINK TO THE READ BUFFER
1795   005200  017012                          WLNK1                   ;LINK TO THE WRITE BUFFER
1796   005202  104417                          SFFG                    ;SPACE TO BEGINNING ON 2ND FILE
1797   005204  104414  016762                  READ    ,RLNK2          ;READ 1ST BLOCK OF 2ND FILE
1798   005210  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
1799   005212  016762                          RLNK2                   ;LINK TO THE READ BUFFER
1800   005214  017016                          WLNK2                   ;LINK TO THE WRITE BUFFER
1801                                  ;;*****************************************************************
1802                                  ;*TEST 34        TEST SFFG AFTER SFFG
1803                                  ;*****************************************************************
1804   005216  000004                 TST34:  SCOPE
1805   005220  012737  005274  001172          MOV     #TST35,$ESCAPE  ;;ESCAPE TO TEST 35 ON ERROR
1806   005226  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
1807   005230  104421                          REWIND                  ;GO TO BOT
1808   005232  104413  017012                  WRITE   ,WLNK1          ;WRITE F BLOCK
1809   005236  104412                          WFG                     ;WRITE FILE GAP
1810   005240  104413  017016                  WRITE   ,WLNK2          ;WRITE A BLOCK
1811   005244  104412                          WFG                     ;WRITE A FILE GAP
1812   005246  104413  017022                  WRITE   ,WLNK3          ;WRITE A BLOCK
1813   005252  104412                          WFG                     ;WRITE A FILE GAP
1814   005254  104421                          REWIND                  ;GO TO "BOT"
1815   005256  104417                          SFFG                    ;SPACE OVER 1ST FILE
1816   005260  104417                          SFFG                    ;SPACE OVER 2ND FILE
1817   005262  104414  016766                  READ    ,RLNK3          ;READ BLOCK OF 3RD FILE
1818   005266  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
1819   005270  016766                          RLNK3                   ;LINK TO THE READ BUFFER
1820   005272  017022                          WLNK3                   ;LINK TO THE WRITE BUFFER
1821                                  ;;*****************************************************************
1822                                  ;*TEST 35        TEST SFFG AFTER SFBG
1823                                  ;*****************************************************************
1824   005274  000004                 TST35:  SCOPE
1825   005276  012737  005350  001172          MOV     #TST36,$ESCAPE  ;;ESCAPE TO TEST 36 ON ERROR
1826   005304  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
1827   005306  104421                          REWIND                  ;GO TO "BOT"
1828   005310  104413  017012                  WRITE   ,WLNK1          ;WRITE A BLOCK
1829   005314  104413  017016                  WRITE   ,WLNK2          ;WRITE A BLOCK
1830   005320  104412                          WFG                     ;WRITE A GAP
1831   005322  104413  017022                  WRITE   ,WLNK3          ;WRITE A BLOCK
1832   005326  104412                          WFG                     ;WRITE A GAP
1833   005330  104421                          REWIND                  ;GO TO "BOT"
1834   005332  104420                          SFBG                    ;SPACE OVER THE 1ST BLOCK(AND 1ST FILE)
1835   005334  104417                          SFFG                    ;SPACE TO BEGINNING OF 2ND FILE
1836   005336  104414  016766                  READ    ,RLNK3          ;READ 1ST BLOCK OF 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 SFBG
1842                                  ;*****************************************************************
1843   005350  000004                 TST36:  SCOPE
1844   005352  012737  005422  001172          MOV     #TST37,$ESCAPE  ;;ESCAPE TO TEST 37 ON ERROR
```

```
1845  005360  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1846  005362  104421                          REWIND
1847  005364  104413  017012                  WRITE   ,WLNK1                  ;WRITE 2 BLOCKS
1848  005370  104413  017016                  WRITE   ,WLNK2
1849  005374  104421                          REWIND                          ;GO TO "BOT"
1850  005376  104420                          SFBG                            ;GET OVER THE 1ST BLOCK
1851  005400  104412                          WFG                             ;WRITE A GAP OVER 2ND BLOCK
1852  005402  104413  017022                  WRITE   ,WLNK3                  ;WRITE A BLOCK
1853  005406  104415                          BSFG                            ;BACK OVER THE BLOCK
1854  005410  104414  016766                  READ    ,RLNK3                  ;AND READ IT
1855  005414  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1856  005416  016766                          RLNK3                           ;LINK TO THE READ BUFFER
1857  005420  017022                          WLNK3                           ;LINK TO THE WRITE BUFFER
1858                                  ;;**********************************************************************
1859                                  ;*TEST 37        TEST WRITE AFTER SFBG
1860                                  ;;**********************************************************************
1861  005422  000004                  TST37:  SCOPE
1862  005424  012737  005474  001172          MOV     #TST40,$ESCAPE  ;;ESCAPE TO TEST 40 ON ERROR
1863  005432  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1864  005434  104421                          REWIND                          ;GO TO BOT
1865  005436  104413  017012                  WRITE   ,WLNK1                  ;WRITE 2 BLOCK
1866  005442  104413  017016                  WRITE   ,WLNK2
1867  005446  104421                          REWIND                          ;GO TO BOT
1868  005450  104420                          SFBG                            ;SPACE OVER 1ST BLOCK
1869  005452  104413  017022                  WRITE   ,WLNK3                  ;WRITE OVER 2ND BLOCK
1870  005456  104415                          BSFG                            ;GO TO BEGINNING OF FILE
1871  005460  104420                          SFBG                            ;SPACE OVER 1ST BLOCK
1872  005462  104414  016766                  READ    ,RLNK3                  ;READ THE NEW 2ND BLOCK
1873  005466  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1874  005470  016766                          RLNK3                           ;LINK TO THE READ BUFFER
1875  005472  017022                          WLNK3                           ;LINK TO THE WRITE BUFFER
1876                                  ;;**********************************************************************
1877                                  ;*TEST 40        3 ONE BLOCK FILES---MULTI SPACING & TEST SFFG AFTER BSFG
1878                                  ;;**********************************************************************
1879  005474  000004                  TST40:  SCOPE
1880  005476  012737  005564  001172          MOV     #TST41,$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                    ;ST41:  SCOPE
1905  005566  012737  005672  001'72             MOV     #TST42,$ESCAPE  ;;ESCAPE TO TEST 42 ON ERROR
1906  005574  104422                           SELDRV                          ;SELECT DRIVE FOR TESTING
1907  005576  104421                           REWIND                          ;GO TO "BOT"
1908  005600  104413  017012                   WRITE   ,WLNK1                  ;WRITE A 2 BLOCK FILE
1909  005604  104413  017016                   WRITE   ,WLNK2
1910  005610  104412                           WFG
1911  005612  104413  017022                   WRITE   ,WLNK3                  ;WRITE A 2 BLOCK FILE
1912  005616  104413  017026                   WRITE   ,WLNK4
1913  005622  104412                           WFG
1914  005624  104413  017032                   WRITE   ,WLNK5                  ;WRITE A 2 BLOCK FILE
1915  005630  104413  017036                   WRITE   ,WLNK6
1916  005634  104412                           WFG
1917  005636  104421                           REWIND                          ;GO TO "BOT"
1918  005640  104417                           SFFG                            ;SPACE OVER 1ST FILE
1919  005642  104417                           SFFG                            ;SPACE OVER 2ND FILE
1920  005644  104414  016776                   READ    ,RLNK5                  ;READ 1ST BLOCK OF 3RD FILE
1921  005650  104423                           BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1922  005652  016776                           RLNK5                           ;LINK TO THE READ BUFFER
1923  005654  017032                           WLNK5                           ;LINK TO THE WRITE BUFFER
1924  005656  104415                           BSFG                            ;BACK TO BEGINNING OF 3RD FILE
1925  005660  104414  016776                   READ    ,RLNK5                  ;READ 1ST BLOCK OF 3RD FILE
1926  005664  104423                           BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1927  005666  016776                           RLNK5                           ;LINK TO THE READ BUFFER
1928  005670  017032                           WLNK5                           ;LINK TO THE WRITE BUFFER
1929                                    ;;*****************************************************************
1930                                    ;*TEST 42        MULTI SPACING & TEST SFBG AFTER READ
1931                                    ;;*****************************************************************
1932  005672  000004                    ;ST42:  SCOPE
1933  005674  012737  006042  001172             MOV     #TST43,$ESCAPE  ;;ESCAPE TO TEST 43 ON ERROR
1934  005702  104422                           SELDRV                          ;SELECT DRIVE FOR TESTING
1935  005704  104421                           REWIND                          ;GO TO "BOT"
1936  005706  104413  017012                   WRITE   ,WLNK1                  ;WRITE A 3 BLOCK FILE
1937  005712  104413  017016                   WRITE   ,WLNK2
1938  005716  104413  017022                   WRITE   ,WLNK3
1939  005722  104412                           WFG
1940  005724  104413  017026                   WRITE   ,WLNK4                  ;WRITE A 2 BLOCK FILE
1941  005730  104413  017032                   WRITE   ,WLNK5
1942  005734  104412                           WFG
1943  005736  104421                           REWIND                          ;GO TO "BOT"
1944  005740  104417                           SFFG                            ;SPACE OVER THE 1ST FILE
1945  005742  104417                           SFFG                            ;SPACE OVER THE 2ND FLE
1946  005744  104415                           BSFG                            ;BACK TO BEGINNING OF 2 FILE
1947  005746  104414  016772                   READ    ,RLNK4                  ;READ 1ST BLOCK OF 2ND FILE
1948  005752  104423                           BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1949  005754  016772                           RLNK4                           ;LINK TO THE READ BUFFER
1950  005756  017026                           WLNK4                           ;LINK TO THE WRITE BUFFER
1951  005760  104415                           BSFG                            ;BACK TO BEGINNING OF 2ND FILE
1952  005762  104415                           BSFG                            ;BACK OVER 1ST FILE
1953  005764  104414  016756                   READ    ,RLNK1                  ;READ 1ND BLOCK OF 1ST FILE
1954  005770  104423                           BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1955  005772  016756                           RLNK1                           ;LINK TO THE READ BUFFER
1956  005774  017012                           WLNK1                           ;LINK TO THE WRITE BUFFER
```

```
 1957  005776  104415                    BSFG                    ;GO TO BEGINNING OF 1ST FILE
 1958  006000  104414  016756            READ    ,RLNK1          ;READ 1ST BLOCK
 1959  006004  104423                    BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1960  006006  016756                    RLNK1                   ;LINK TO THE READ BUFFER
 1961  006010  017012                    WLNK1                   ;LINK TO THE WRITE BUFFER
 1962  006012  104420                    SFBG                    ;GET OVER 2ND BLOCK
 1963  006014  104414  016766            READ    ,RLNK3          ;READ 3RD BLOCK
 1964  006020  104423                    BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1965  006022  016766                    RLNK3                   ;LINK TO THE READ BUFFER
 1966  006024  017022                    WLNK3                   ;LINK TO THE WRITE BUFFER
 1967  006026  104417                    SFFG                    ;GO TO BEGINNING OF 2ND FILE
 1968  006030  104414  016772            READ    ,RLNK4          ;READ 1ST BLOCK OF 2ND FILE
 1969  006034  104423                    BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1970  006036  016772                    RLNK4                   ;LINK TO THE READ BUFFER
 1971  006040  017026                    WLNK4                   ;LINK TO THE WRITE BUFFER
 1972                          ;;*****************************************************************
 1973                          ;*TEST 43        TWO BLOCK FILES--MULTI SPACING & TEST SFFG AFTER BSBG
 1974                          ;;*****************************************************************
 1975  006042  000004          TST43:  SCOPE
 1976  006044  012737  006162  001172    MOV     #TST44,$ESCAPE  ;;ESCAPE TO TEST 44 ON ERROR
 1977  006052  104422                    SELDRV                  ;SELECT DRIVE FOR TESTING
 1978  006054  104421                    REWIND                  ;GO TO BOT
 1979  006056  104413  017012            WRITE   ,WLNK1          ;WRITE A 2 BLOCK FILE
 1980  006062  104413  017016            WRITE   ,WLNK2
 1981  006066  104412                    WFG
 1982  006070  104413  017022            WRITE   ,WLNK3          ;WRITE A 2 BLOCK FILE
 1983  006074  104413  017026            WRITE   ,WLNK4
 1984  006100  104412                    WFG
 1985  006102  104413  017032            WRITE   ,WLNK5          ;WRITE A 2 BLOCK FILE
 1986  006106  104413  017036            WRITE   ,WLNK6
 1987  006112  104412                    WFG
 1988  006114  104421                    REWIND                  ;BOT
 1989  006116  104417                    SFFG                    ;SPACE OVER 3 FILES
 1990  006120  104417                    SFFG
 1991  006122  104417                    SFFG
 1992  006124  104415                    BSFG                    ;BACK OVER 2 FILES
 1993  006126  104415                    BSFG
 1994  006130  104416                    BSBG                    ;BACK TO 2ND BLOCK OF 1ST FILE
 1995  006132  104417                    SFFG                    ;GO TO BEGINNING OF 2ND FILE
 1996  006134  104414  016766            READ    ,RLNK3          ;READ 1ST RECORD OF 2ND FILE
 1997  006140  104423                    BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1998  006142  016766                    RLNK3                   ;LINK TO THE READ BUFFER
 1999  006144  017022                    WLNK3                   ;LINK TO THE WRITE BUFFER
 2000  006146  104417                    SFFG                    ;GO TO 3RD FILE
 2001  006150  104414  016776            READ    ,RLNK5          ;READ 1ST RECORD OF 3RD FILE
 2002  006154  104423                    BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 2003  006156  016776                    RLNK5                   ;LINK TO THE READ BUFFER
 2004  006160  017032                    WLNK5                   ;LINK TO THE WRITE BUFFER
 2005                          ;;*****************************************************************
 2006                          ;*TEST 44        6 ONE BLOCK FILES---MULTI SPACING
 2007                          ;;*****************************************************************
 2008  006162  000004          TST44:  SCOPE
 2009  006164  012737  006352  001172    MOV     #$EOP,$ESCAPE   ;;ESCAPE TO $EOP ON ERROR
 2010  006172  104422                    SELDRV                  ;SELECT DRIVE FOR TESTING
 2011  006174  104421                    REWIND                  ;GO TO BOT
 2012  006176  104413  017012            WRITE   ,WLNK1          ;WRITE A 1 RECORD FILE
```

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

```
                                .SBTTL  END OF PASS ROUTINE

                                ;;*****************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*TYPE "END PASS"
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO START
                                ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
                                ;*$ENDMG CAN BE CHANGED TO 7.

006352                          $EOP:
006352  000004                          SCOPE
006354  005037  001102                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
006360  005037  001170                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
006364  005237  001100                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
006370  042737  100000  001100          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
006376  005327                          DEC     (PC)+           ;;LOOP?
006400  000001                  $EOPCT: .WORD   1
006402  003017                          BGT     $DOAGN          ;;YES
006404  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
006406  000001                  $ENDCT: .WORD   1
006410  006400                          $EOPCT
006412  104401  006451                  TYPE    ,$ENDMG         ;;TYPE "END PASS"
006416  104401  006446                  TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
006422  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
006426  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
006430  000005                          RESET                   ;;CLEAR THE WORLD
006432  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
006434  000240                          NOP                     ;;SAVE ROOM
006436  000240                          NOP                     ;;FOR
006440  000240                          NOP                     ;;ACT11
006442                          $DOAGN:
006442  000137                          JMP     @(PC)+          ;;RETURN
006444  002312                  $RTNAD: .WORD   START
006446     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
006451     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS/
006456  050040  051501  000123
```

```
2093                                    ;;*****************************************************************
2094                                    ;
2095                                    ;SBTTL          ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABLITY
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    #BIT07!BIT00,@SWR       ;"READONLY" SELECTED? (RD1PAS)
2117  006540  001412                           BEQ     4$                      ;BR IF YES
2118  006542  122777  000203  172370          CMPB    #BIT07!BIT01!BIT00,@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

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

006570                  $SCOPE:
006570  104406                 CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
006572  105777  172346         TSTB    @$TKS           ;TTY FLAG UP?
006576  100005                 BPL     100$            ;BR IF NO
006600  104407                 RDCHR                   ;GO GET ONE CHARACTER FROM TTY
006602  122627  000003         CMPB    (SP)+,#3        ;IS IT A CONTROL/C ?
006606  001001                 BNE     100$            ;BR IF NO
006610  104424                 SETLOOP                 ;GO GET "TEST" AND "SCOPE LOOP" ADDRESSES
006612  005727         100$:   TST     (PC)+           ;LOOP ON TEST?
006614  000000         LOOPKEY:        0
006616  001105                 BNE     $OVER           ;BR IF YES
006620  032777  040000  172312 1$:     BIT     #BIT14,@$WR     ;;LOOP ON PRESENT TEST?
006626  001101                 BNE     $OVER           ;;YES IF SW14=1
                        ;#####START OF CODE FOR THE XOR TESTER#####
006630  000416         $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
006632  013746  000004         MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
006636  012737  006656  000004 MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
006644  005737  177060         TST     @#177060        ;;TIME OUT ON XOR?
006650  012637  000004         MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
006654  000453                 BR      $SVLAD          ;;GO TO THE NEXT TEST
006656  022626         5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
006660  012637  000004         MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
006664  000413                 BR      7$              ;;LOOP ON THE PRESENT TEST
006666                  6$:;#####END OF CODE FOR THE XOR TESTER#####
006666  105737  001103 2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
006672  001421                 BEQ     3$              ;;BR IF NO
006674  123737  001115  001103        CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
006702  101015                 BHI     3$              ;;BR IF NO
006704  032777  001000  172226        BIT     #BIT09,@$WR     ;;LOOP ON ERROR?
006712  001404                 BEQ     4$              ;;BR IF NO
006714  013737  001110  001106 7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
006722  000443                 BR      $OVER
006724  105037  001103 4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
006730  005037  001170         CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
006734  000415                 BR      1$              ;;ESCAPE TO THE NEXT TEST
006736  032777  004000  172174 3$:     BIT     #BIT11,@$WR     ;;INHIBIT ITERATIONS?
006744  001011                 BNE     1$              ;;BR IF YES
006746  005737  001100         TST     $PASS           ;;IF FIRST PASS OF PROGRAM
006752  001406                 BEQ     1$              ;;      INHIBIT ITERATIONS
006754  005237  001104         INC     $ICNT           ;;INCREMENT ITERATION COUNT
006760  023737  001170  001104        CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
006766  002021                 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    $TSTNM            ;;COUNT TEST NUMBERS
 2184  007010  011637  001106                   MOV     (SP),$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     $TSTNM,@DISPLAY   ;;DISPLAY TEST NUMBER
 2189  007040  013716  001106                   MOV     $LPADR,(SP)       ;;FUDGE RETURN ADDRESS
 2190  007044  000002                           RTI                       ;;FIXES PS
```

```
                                .SBTTL  ERROR HANDLER ROUTINE

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

007046                          $ERROR:
007046  104406                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
007050  011437  001162                  MOV     @TACS,@#$REGO           ;SAVE THE TA11 STATUS
007054  011537  001164                  MOV     @TADB,@#$REG1           ;SAVE THE DATA BUFFER
007060  105714                          TSTB    @TACS                   ;IF "XFER REQ" IS UP KNOCK IT DOWN
007062  100003                          BPL     100$
007064  112714  000020                  MOVB    #ILBS,@TACS
007070  000406                          BR      101$
007072  032714  000040          100$:   BIT     #READY,@TACS            ;IS READY SET?
007076  001003                          BNE     101$                    ;BR IF YES
007100  000005                          RESET                           ;REGAIN CONTROL
007102  013714  001162                  MOV     @#$REGO,@TACS           ;RESTORE THE TACS
007106  022626                  101$:   CMP     (SP)+,(SP)+             ;POP THE STACK
007110  005737  001246                  TST     ASKKEY                  ;MANUAL LOOPING?
007114  002002                          BGE     102$                    ;BR IF NO
007116  011637  001172                  MOV     (SP),$ESCAPE            ;SETUP FOR MANUAL ESCAPE
007122  016616  177774          102$:   MOV     -4(SP),(SP)             ;GET PC+2 OF ERROR
007126  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
007132  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
007134  013777  001102  172000          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
007142  032777  002000  171770          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
007150  001402                          BEQ     1$              ;;NO - SKIP
007152  104401  001174                  TYPE    ,$BELL          ;;RING BELL
007156  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
007162  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
007166  162737  000002  001116          SUB     #2,$ERRPC
007174  117737  171716  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
007202  032777  020000  171730          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
007210  001004                          BNE     20$             ;;SKIP TYPEOUTS
007212  004737  007276                  JSR     PC,TYPERR       ;;GO TO USER ERROR ROUTINE
007216  104401  001201                  TYPE    ,$CRLF
007222                          20$:
007222  005777  171712          2$:     TST     @SWR            ;;HALT ON ERROR
007226  100002                          BPL     3$              ;;SKIP IF CONTINUE
007230  000000                          HALT                    ;;HALT ON ERROR!
007232  104406                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
007234  032777  001000  171676  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
007242  001402                          BEQ     4$              ;;BR IF NO
007244  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
007250  005737  001172          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
007254  001402                          BEQ     5$              ;;BR IF NONE
007256  013716  001172                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
```

```
2247  007262                          5$:
2248  007262  022737  006432  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
2249  007270  001001                          BNE     6$              ;;BRANCH IF NO
2250  007272  000000                          HALT                    ;;YES
2251  007274                          6$:
2252  007274  000002                          RTI                     ;;RETURN
```

SEQ 0050

```
                                        ;;*****************************************************************
                                        ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
007276  104401  001201                  TYPERR: TYPE    ,$CRLF                  ;TYPE A CARRIAGE RETURN & LINE FEED
007302  104401                                  TYPE                            ;TYPE THE FUNCTION BEING PERFORMED
007304  000000                          MFUNC:  0                               ;MESSAGE POINTER GOES HERE
007306  104401  001201                          TYPE    ,$CRLF
007312  162737  000002  001204                  SUB     #2,@#SAVPC              ;ADJUST THE MAIN FLOW PC
007320  010046                                  MOV     R0,-(SP)                ;SAVE R0
007322  113700  001114                          MOVB    @#$ITEMB,R0             ;PICKUP THE ITEM INDEX
007326  005300                                  DEC     R0                      ;ADJUST THE INDEX
007330  006300                                  ASL     R0                      ;SO IT WILL WORK FOR
007332  006300                                  ASL     R0                      ;THE ERROR TABLE
007334  006300                                  ASL     R0
007336  062700  001270                          ADD     #$ERRTB,R0              ;FORM THE TABLE POINTER
007342  012037  007350                          MOV     (R0)+,1$                ;PICKUP "ERROR MESSAGE" POINTER
007346  104401                                  TYPE                            ;TYPE "ERROR MESSAGE"
007350  000000                          1$:     0                               ;"ERROR MESSAGE POINTER" GOES HERE
007352  012037  007362                          MOV     (R0)+,2$                ;PICKUP "DATA HEADER" POINTER
007356  001402                                  BEQ     3$                      ;IF "0" DON'T TYPE
007360  104401                                  TYPE                            ;TYPE "DATA HEADER
007362  000000                          2$:     0                               ;"DATA HEADER" POINTER GOES HERE
007364  012000                          3$:     MOV     (R0)+,R0                ;PICKUP "DATA POINTER"
007366  001004                                  BNE     5$                      ;IF THERE IS DATA TO TYPE GO DO IT
007370  012600                          4$:     MOV     (SP)+,R0                ;RESTORE R0
007372  104401  001201                          TYPE    ,$CRLF                  ;TYPE A CARRAGE RETURN&LINE FEED
007376  000207                                  RTS     PC                      ;RETURN
007400                                  5$:
007400  013046                                  MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
                                                                        ;;TYPE DATA
007402  104402                                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
007404  005710                                  TST     (R0)                    ;TERMINATOR?
007406  001770                                  BEQ     4$                      ;BR IF YES
007410  104401  007416                          TYPE    ,6$                     ;TYPE 2 SPACES
007414  000771                                  BR      5$                      ;LOOP
007416  020040     000                  6$:     .ASCIZ  /  /                    ;ASCII STRING OF 2 SPACES
        007422                                  .EVEN
```

```
                                ;;*******************************************************************
                                .SBTTL          ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST

                                ;CALL
                                ;       JSR     PC,@#ASKDRV
                                ;       RETURN                  ;NOTE: R0 AND R1 ARE DESTROYED

007422  104401  015136         ASKDRV: TYPE    ,MSGDRV         ;<CRLF>"DRIVE(S)? "
007426  005037  001240                 CLR     DRVKEY
007432  104410                         RDLIN                   ;GO GET A DRIVE
007434  012600                         MOV     (SP)+,R0        ;SETUP TO CHECK FOR VALID DRIVE(S)
007436  105710                         TSTB    @R0             ;WAS A DRIVE SELECTED?
007440  001425                         BEQ     NOTLGL          ;BR IF NO
007442  012701  001240                 MOV     #DRVKEY,R1
007446  122710  000101         LOOP:   CMPB    #'A,@R0         ;WAS DRIVE "A" SELECTED?
007452  001002                         BNE     NOTA            ;BR IF NO
007454  112021                         MOVB    (R0)+,(R1)+     ;SET KEY FOR DRIVE "A"
007456  000411                         BR      NEXT
007460  122710  000102         NOTA:   CMPB    #'B,@R0         ;WAS DRIVE "B" SELECTED?
007464  001002                         BNE     NOTB            ;BR IF NO
007466  112021                         MOVB    (R0)+,(R1)+     ;SET KEY FOR DRIVE "B"
007470  000404                         BR      NEXT
007472  122710  000054         NOTB:   CMPB    #54,@R0         ;WAS A COMMA TYPED?
007476  001006                         BNE     NOTLGL          ;BR IF NO
007500  105720                         TSTB    (R0)+           ;DUMP THE COMMA
007502  105710                 NEXT:   TSTB    @R0             ;TERMINATOR?
007504  001406                         BEQ     EXIT            ;BR IF YES
007506  022701  001242                 CMP     #DRVKEY+2,R1    ;TWO DRIVES SELECTED?
007512  101355                         BHI     LOOP            ;BR IF NO
007514  104401  001200         NOTLGL: TYPE    ,$QUES          ;ILLEGAL INPUT DETECTED
007520  000740                         BR      ASKDRV          ;GO TRY AGAIN
007522  005737  001240         EXIT:   TST     DRVKEY          ;ANY DRIVE SELECTED?
007526  001772                         BEQ     NOTLGL          ;BR IF NO
007530  000207                         RTS     PC

                                ;;*******************************************************************
                                ;CALL
                                ;JSR    PC,@#ASKADR

007532  010046                 ASKADR: MOV     R0,-(SP)        ;SAVE R0
007534  104401  015152         1$:     TYPE    ,MSGASK         ;"TACS?"
007540  104411                         RDOCT                   ;GET VALUE
007542  012600                         MOV     (SP)+,R0        ;PICK UP THE OCTAL NUMBER
007544  001423                         BEQ     3$              ;IF "0" USE OLD VALUES
007546  020027  160000                 CMP     R0,#160000      ;MAKE SURE IT IS A BUS ADDRESS
007552  103770                         BLO     1$
007554  010037  001220                 MOV     R0,@#TACSL      ;SAVE TOE TACS
007560  062700  000002                 ADD     #2,R0           ;STEP TO TADB ADDRESS
007564  010037  001224                 MOV     R0,@#TADBL      ;AND SAVE IT
007570  013737  001220  001222         MOV     @#TACSL,@#TACSH ;SET UP TACS UPPER
007576  005237  001222                 INC     @#TACSH         ;BYTE POINTER

007602  013737  001224  001226         MOV     @#TADBL,@#TADBH ;SET UP TADB UPPER
007610  005237  001226                 INC     @#TADBH         ;BYTE POINTER
007614  104401  015162         3$:     TYPE    ,MSGVEC         ;"VECTOR?"
```

```
2345  007620  104411                  RDOCT
2346  007622  012600                  MOV     (SP)+,R0
2347  007624  001411                  BEQ     5$
2348  007626  020027  001000          CMP     R0,#1000                ;MAKE SURE ADDRESS IS IN VECTOR AREA
2349  007632  103370                  BHIS    3$
2350  007634  010037  001230          MOV     R0,@#TAVEC              ;SAVE AS VECTOR ADDRESS
2351  007640  062700  000002          ADD     #2,R0
2352  007644  010037  001232          MOV     R0,@#TAVEC+2
2353  007650  104401  015172  5$:     TYPE    ,MSGPRI                 ;ASK FOR PRIORITY
2354  007654  104411                  RDOCT
2355  007656  012600                  MOV     (SP)+,R0
2356  007660  001413                  BEQ     6$                      ;IF "0" USE OLD VALUE
2357  007662  020027  000007          CMP     R0 #7                   ;MAKE SURE ITS VALID
2358  007666  101370                  BHI     5$
2359  007670  000300                  SWAB    R0                      ;PUT INTO HIGH BYTE
2360  007672  006200                  ASR     R0                      ;AND SHIFT
2361  007674  006200                  ASR     R0                      ;INTO PROPER
2362  007676  006200                  ASR     R0                      ;POSITION
2363  007700  042700  177437          BIC     #^C<340>,R0             ;SAVE ONLY PRIORITY BITS
2364  007704  010037  001234          MOV     R0,@#TAPRIO             ;STORE IT AWAY
2365  007710  104401  015204  6$:     TYPE    ,MTACS                  ;TACS="
2366  007714  013746  001220          MOV     @TACSL,-(SP)    ;;SAVE TACSL FOR TYPEOUT
2367  007720  104402                  TYPOC                   ;;GO TYPE -OCTAL ASCII(ALL DIGITS)
2368  007722  104401  015213          TYPE    ,MTADB                  ;TADB="
2369  007726  013746  001224          MOV     @TADBL,-(SP)    ;;SAVE TADBL FOR TYPEOUT
2370  007732  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2371  007734  104401  015222          TYPE    ,MTAVEC                 ;VECTOR="
2372  007740  013746  001230          MOV     @TAVEC,-(SP)    ;;SAVE TAVEC FOR TYPEOUT
2373  007744  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2374  007746  104401  015233          TYPE    ,MTAPRI                 ;PRIORITY="
2375  007752  013746  001234          MOV     @TAPRIO,-(SP)   ;;SAVE TAPRIO FOR TYPEOUT
2376  007756  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2377  007760  104401  015246          TYPE    ,MSGOK                  ;OK?"
2378  007764  104407                  RDCHR                   ;GO READ ONE CHARACTER
2379  007766  012600                  MOV     (SP)+,R0                ;GET IT
2380  007770  022700  000015          CMP     #15,R0                  ;IS IT "CR"?
2381  007774  001406                  BEQ     7$                      ;BRANCH IF YES
2382  007776  022700  000131          CMP     #'Y,R0                  ;IS IT "Y"?
2383  010002  001403                  BEQ     7$                      ;IT WAS
2384  010004  104401  001200          TYPE    ,$QUES                  ;TYPE "?"
2385  010010  000651                  BR      1$                      ;AND LET HIM CORRECT THEM
2386  010012  104401  015254  7$:     TYPE    ,MYES                   ;TYPE OUT "YES"
2387  010016  012600                  MOV     (SP)+,R0                ;RESTORE R0
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 INITILIZED BEFORE ENTERING
2395                          ;
2396                          ;CALL:
2397                          ;       JSR     R5,@#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                                    ;                       ;OR STRING IS TO LONG
2402                                    ;       NORMAL RETURN   ;OCTAL NUMBER IS IN R1
2403
2404   010022  010046                   A2OCT:  MOV     R0,-(SP)        ;SAVE R0
2405   010024  012300                           MOV     (R3)+,R0        ;PICK UP THE POINTER
2406   010026  012737  000006  010042           MOV     #6,2$           ;MAX. NUMBER OF CHAR.'S ALLOWED
2407   010034  105710                   1$:     TSTB    (R0)            ;TERMINATOR?
2408   010036  001420                           BEQ     3$              ;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    (R0),#'0        ;CHECK THIS CHAR. IS BETWEEN
2413   010052  002413                           BLT     4$              ;0 AND 7
2414   010054  121027  000067                   CMPB    (R0),#'7
2415   010060  003010                           BGT     4$
2416   010062  142710  177770                   BICB    #^C7,(R0)       ;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    (R0)+,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)+,R0        ;RESTORE R0
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  010106                            T.SETLOOP:
2429  010106  104401  015316                   TYPE    ,MTEST                  ;TYPE "TEST PC?"
2430  010112  104410                           RDLIN                           ;GET AN ASCII STRING
2431  010114  012637  010142                   MOV     (SP)+,11$               ;GET FIRST ADDRESS OF INPUT BUFFER
2432  010120  105777  000016                   TSTB    @11$                    ;FIRST CHAR. A "CR" ?
2433  010124  001442                           BEQ     3$                      ;BR IF YES---GO CLEAR THE LOOP KEY
2434  010126  022626                           CMP     (SP)+,(SP)+             ;POP POP
2435  010130  010137  010244                   MOV     R1,SAVR1                ;SAVE R1
2436  010134  005001                           CLR     R1                      ;SET PARTIAL TO ZERO
2437  010136  004337  010022                   JSR     R3,@#A2OCT              ;GO CHANGE ASCII TO OCTAL
2438  010142  000000                    11$:   0                               ;POINTER TO STRING
2439  010144  000760                           BR      T.SETLOOP               ;ERROR RETURN--GO TRY AGAIN
2440  010146  005721                           TST     (R1)+                   ;JUST INCREMENTING
2441  010150  010116                           MOV     R1,(SP)                 ;PUT "TEST PC"; ON THE STACK
2442  010152  104401  015331            5$:   TYPE    ,MLOOP                  ;TYPE "LOOP PC ?"
2443  010156  104410                           RDLIN                           ;GET A STRING OF ASCII
2444  010160  012637  010200                   MOV     (SP)+,12$               ;GET FIRST ADDRESS OF INPUT BUFFER
2445  010164  105777  000010                   TSTB    @12$                    ;JUST A "CR" ?
2446  010170  001410                           BEQ     1$                      ;BR IF YES
2447  010172  005001                           CLR     R1                      ;CLEAR PARTIAL
2448  010174  004337  010022                   JSR     R3,@#A2OCT              ;CHANGE TO OCTAL
2449  010200  000000                    12$:   0
2450  010202  000763                           BR      5$
2451  010204  010137  001106                   MOV     R1,@#$LPADR             ;SAVE SCOPE LOOP ADDRESS
2452  010210  000402                           BR      2$                      ;GO SET LOOP KEY
2453  010212  011637  001106            1$:   MOV     (SP),@#$LPADR           ;SAVE SCOPE LOOP ADDRESS
2454  010216  011637  001110            2$:   MOV     (SP),@#$LPERR           ;ERROR LOOP SAME AS THE TEST ADDRESS
2455  010222  012737  177777  006614          MOV     #-1,@#LOOPKEY           ;SET THE LOOP KEY
2456  010230  000402                           BR      4$
2457  010232  005037  006614            3$:   CLR     @#LOOPKEY               ;CLEAR THE LOOP KEY
2458  010236  013701  010244            4$:   MOV     SAVR1,R1                ;RESTORE R1
2459  010242  000002                           RTI                             ;BYE BYE
2460  010244  000000                    SAVR1: 0                               ;HOLD R1 HERE
```

```
2461
2462                                    ;;*****************************************************************
2463                                    ;THIS ROUTINE WILL SELECT THE NEXT DRIVE TO TEST
2464
2465  010246                            T.SELDRV:
2466  010246  032777  000200  170664            BIT     #SW07,@SWR              ;LOCK ON CURRENT DRIVE?
2467  010254  001041                           BNE     5$                      ;BR IF YES
2468  010256  010146                           MOV     R1,-(SP)                ;SAVE R1 ON THE STACK
2469  010260  013701  001236                   MOV     DRIVE,R1                ;GET THE DRIVE
2470  010264  111137  001250                   MOVB    (R1),@#CURDRV           ;SAVE CURRENT DRIVE
2471  010270  121127  000101                   CMPB    (R1),#'A                ;IS IT DRIVE "A"?
2472  010274  001406                           BEQ     1$                      ;BR IF YES
2473  010276  052714  000400                   BIS     #UNIT,@TACS             ;SELECT DRIVE "B"
2474  010302  012737  001262  001256           MOV     #RWDB,@#RWDFLG          ;SET REWIND POINTER TO "B"
2475  010310  000405                           BR      2$
2476  010312  042714  000400           1$:     BIC     #UNIT,@TACS             ;SELECT DRIVE "A"
2477  010316  012737  001260  001256           MOV     #RWDA,@#RWDFLG          ;SET REWIND POINTER TO "A"
2478  010324  005201                   2$:     INC     R1                      ;INCREMENT TO THE NEXT DRIVE
2479  010326  105711                           TSTB    (R1)                    ;OUT OF DRIVES TO TEST?
2480  010330  001002                           BNE     3$                      ;BR IF NO
2481  010332  012701  001240                   MOV     #DRVKEY,R1              ;RESET TO THE FIRST DRIVE
2482  010336  010137  001236           3$:     MOV     R1,DRIVE                ;SAVE FOR NEXT TIME
2483  010342  032777  001000  170570           BIT     #SW09,@SWR              ;IS SWITCH 9=1?
2484  010350  001002                           BNE     4$                      ;BR IF YES
2485  010352  105037  001103                   CLRB    @#SERFLG                ;CLEAR THE ERROR INDICATOR
2436  010356  012601                   4$:     MOV     (SP)+,R1                ;RESTORE R1
2487  010360  000002                   5$:     RTI
```

```
                        ;*************************************************************
                        ;THIS ROUTINE WILL COMPARE THE READ AND WRITE BUFFERS
010362                  .BLKCMP:
010362  011600                  MOV     (SP),R0                 ;GET THE PC OF THE CALL
010364  010037  001204          MOV     R0,@#SAVPC              ;SAVE THE PC OF THE CALL
010370  012737  015521  007304  MOV     #MBUFCK,@#MFUNC         ;POINTER FOR .ASCII MESSAGE
010376  012003                  MOV     (R0)+,R3                ;PICKUP THE READ LINK ADDRESS
010400  012301                  MOV     (R3)+,R1                ;PICKUP THE READ BUFFER ADDRESS
010402  011303                  MOV     (R3),R3                 ;PICKUP THE READ BUFFER SIZE
010404  013002                  MOV     @(R0)+,R2               ;PICKUP THE WRITE BUFFER ADDRESS
010406  005037  010446          CLR     2$+2                    ;ZERO THE OFFSET
010412  010016                  MOV     R0,(SP)                 ;SET THE RETURN
010414  005037  001206          CLR     @#BYTNUM                ;ZERO THE BYTE NUMBER
010420  032777  000010  170512  BIT     #SW03,@SWR              ;COMPARE DATA BUFFERS?
010426  001401                  BEQ     1$                      ;BR IF YES
010430  000002                  RTI                             ;NO---GO BACK
010432  005303          1$:     DEC     R3                      ;CHECK FOR THE TERMINATION
010434  002001                  BGE     4$                      ;BR IF MORE BYTES TO COMPARE
010436  000002                  RTI                             ;RETURN
010440  005237  001206          4$:     INC     @#BYTNUM        ;COUNT THIS BYTE
010444  122162  000000          2$:     CMPB    (R1)+,0(R2)     ;COMPARE READ AND WRITE BUFFER
010450  001006                  BNE     3$                      ;BR IF NOT EQUAL
010452  005237  010446          INC     2$+2                    ;INCREMENT WRITE BUFFER INDEX
010456  042737  177774  010446  BIC     #^C3,2$+2               ;DON'T LET IT GET BIGGER THAN 3
010464  000762                  BR      1$
010466  114137  001126          3$:     MOVB    -(R1),@#$BDDAT  ;GET THE RECEIVED BYTE
010472  063702  010446          ADD     2$+2,R2                 ;FORM ADDRESS
010476  111237  001124          MOVB    (R2),@#$GDDAT           ;GET THE EXPECTED BYTE
010502  104010                  ERROR   10                      ;RECEIVED AND EXPECTED DATA NOT EQUAL
                        ;*************************************************************
                        ;THIS ROUTINE WILL WRITE A FILE GAP ON THE TAPE
010504  011637  001204  .WFG:   MOV     (SP),@#SAVPC            ;SAVE THE PC OF THE CALL
010510  012737  015502  007304  MOV     #MXWFG,@#MFUNC          ;SET MESSAGE POINTER FOR "WFG"
010516  004037  012224  1$:     JSR     R0,@#DO.CMD             ;GO START A "WFG"
010522  000101                  XWFG!INT.EN!GO
010524  005037  177776          CLR     @#PS                    ;ALLOW INTERRUPTS
010530  004037  011716          JSR     R0,@#WAITFLAG           ;GO WAIT FOR A FLAG
010534  104003                  ERROR   3                       ;NO FLAGS OCCURRED
010536  104004                  ERROR   4                       ;NO INTERRUPT
010540  004037  012014          JSR     R0,@#FLAGS              ;NORMAL RETURN---FIND OUT WHAT  FLAG
010544  104002                  ERROR   2                       ;RETURN HERE IF "XFER REQ" AND "READY" =0
010546  104002                  ERROR   2                       ;RETURN HERE IF "XFER REQ" =1
010550  000002                  RTI                             ;"RETURN HERE IF "READY"=1 AND "ERROR" =0
010552  104002                  ERROR   2                       ;"READY"=1 AND "ERROR"=1
```

```
2532                                    ;;*****************************************************************
2533                                    ;THIS ROUTINE WILL WRITE A RECORD ON THE TAPE
2534  010554  011637  001204             .WRIT:  MOV     (SP),@#SAVPC            ;SAVE PC OF CALL
2535  010560  012737  015353  007304             MOV     #MXWRIT,@#MFUNC         ;POINTER FOR .ASCIZ OF "WRITE"
2536  010566  017602  000000                    MOV     @(SP),R2                ;GET THE WRITE LINK POINTER
2537  010572  062716  000002                    ADD     #2,(SP)                 ;ADJUST THE RETURN
2538  010576  012201                             MOV     (R2)+,R1                ;GET FIRST ADDRESS OF WRITE BUFFER
2539  010600  005037  012222                    CLR     @#CRC.WD                ;CLEAR THE CRC WORD
2540  010604  005037  010676                    CLR     6$+2                    ;ZERO THE INDEX
2541  010610  012202                             MOV     (R2)+,R2                ;GET NUMBER OF BYTE TO WRITE
2542  010612  003001                             BGT     2$                      ;CHECK FOR LEGAL NUMBER
2543  010614  104011                             ERROR   11                      ;BAD NUMBER OF BYTES
2544  010616  004037  012224             2$:     JSR     R0,@#DO.CMD             ;GO START A "WRITE"
2545  010622  000103                             XWRITE!INT.EN!GO
2546  010624  005037  177776             3$:     CLR     @#PS                    ;ALLOW INTERRUPTS
2547  010630  004037  011716                    JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
2548  010634  104003                             ERROR   3                       ;NO FLAGS OCCURRED
2549  010636  104004                             ERROR   4                       ;INTERRUPT FAILED
2550  010640  005037  001254                    CLR     @#WAITKEY               ;CLEAR THE WAIT FOR INTERRUPT KEY
2551  010644  012737  000340  177776             MOV     #340,@#PS               ;LOCK OUT THE WORLD
2552  010652  004037  012014                    JSR     R0,@#FLAGS              ;WAIT FLAG OCCURRED?
2553  010656  104002                             ERROR   2                       ;"READY"=0 & "XFER REQ" =0
2554  010660  000402                             BR      4$                      ;"XFER REQ"=1
2555  010662  104005                             ERROR   5                       ;"READY"=1 & "ERROR"=0
2556  010664  104002                             ERROR   2                       ;"READY"=1 & "ERROR"=1
2557  010666  005302             4$:     DEC     R2                      ;NEED TO WRITE A BYTE ON TAPE
2558  010670  002417                             BLT     5$                      ;BR IF NO
2559  010672  005000                             CLR     R0
2560  010674  156100  000000             6$:     BISB    0(R1),R0                ;PICKUP BYTE FOR TRANSFER
2561  010700  110015                             MOVB    R0,@TADB                ;TRANSFER A BYTE TO THE TA11
2562  010702  004737  012102                    JSR     PC,@#DO.CRC             ;CALCULATE CRC FOR THIS BYTE
2563  010706  005237  010676                    INC     6$+2                    ;INCREASE THE INDEX
2564  010712  042737  177774  010676             BIC     #^C3,6$+2               ;KEEP IT LESS THAN 4
2565  010720  012777  012340  170302             MOV     #SERV,@TAVEC            ;SETUP TO SERVICE THE INTERRUPT
2566  010726  000736                             BR      3$                      ;LOOP
2567  010730  013761  012222  000004   5$:     MOV     @#CRC.WD,4(R1)          ;SAVE THE CRC WORD
2568  010736  004037  012224                    JSR     R0,@#DO.CMD             ;START A "ILBS"
2569  010742  000122                             XWRITE!INT.EN!ILBS
2570  010744  005037  177776                    CLR     @#PS                    ;ALLOW INTERRUPTS
2571  010750  004037  011716                    JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
2572  010754  104003                             ERROR   3                       ;NO FLAG
2573  010756  104004                             ERROR   4                       ;NO INTERRUPT
2574  010760  004037  012014                    JSR     R0,@#FLAGS              ;WHAT FLAG IS SET?
2575  010764  104002                             ERROR   2                       ;NONE ("XFER REQ"=0 & "READY"=0)
2576  010766  104002                             ERROR   2                       ;"XFER REQ"
2577  010770  000002                             RTI                             ;"READY"
2578  010772  104002                             ERROR   2                       ;"READY" & "ERROR"
```

```
                                        ;*****************************************************************
                                        ;THIS ROUTINE WILL READ A RECORD FROM THE TAPE
011774 011637 001204            ; READ: MOV     (SP),@#SAVPC            ;SAVE THE PC OF THE CALL
011000 012737 015361 007304             MOV     #MXREAD,@#MFUNC         ;SET POINTER OF .ASCIZ MESSAGE
011006 017602 000000                    MOV     @(SP),R2                ;GET THE READ LINK POINTER
011012 062716 000002                    ADD     #2,(SP)                 ;ADJUST FOR RETURN
011016 010200                           MOV     R2,R0                   ;PICKUP AND SAVE THE "CRC" THAT
011020 016000 000034                    MOV     WLNK1-RLNK1(R0),R0      ;    WAS CALCULATED ON THE WRITE
011024 016037 000004 001210             MOV     4(R0),RCRC0             ;        DATA FOR THIS BLOCK
011032 012201                           MOV     (R2)+,R1                ;GET ADDRESS OF BUFFER
011034 012202                           MOV     (R2)+,R2                ;GET NUMBER OF BYTES
011036 003001                           BGT     2$                      ;MAKE SURE NUMBER IS LEGAL
011040 104011                           ERROR   11                      ;ILLEGAL NUMBER OF BYTES
011042 005037 012222            2$:     CLR     @#CRC.WD                ;CLEAR CRC WORD
011046 004037 012224                    JSR     R0,@#DO.CMD             ;GO START A "READ"
011052 000105                           XREAD!INT.EN!GO
011054 005037 177776            3$:     CLR     @#PS                    ;ALLOW INTERRUPTS
011060 105011                           CLRB    (R1)                    ;SET TERMINATOR IF LAST BYTE
011062 004037 011716                    JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
011066 104003                           ERROR   3                       ;NONE OCCURRED
011070 104004                           ERROR   4                       ;NO INTERRUPT
011072 005037 001254                    CLR     @#WAITKEY               ;CLEAR THE WAIT ON INTERRUPT KEY
011076 012737 000340 177776             MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
011104 004037 012014                    JSR     R0,@#FLAGS              ;WHAT FLAG OCCURRED
011110 104002                           ERROR   2                       ;?UNKNOWN FLAG
011112 000402                           BR      4$                      ;"XFER REQ"
011114 104005                           ERROR   5                       ;"READY"
011116 000413                           BR      8$                      ;"READY" WITH "ERROR"
011120 005302                   4$      DEC     R2                      ;READ A BYTE?
011122 002431                           BLT     5$                      ;BR IF NO
011124 005000                           CLR     R0
011126 151500                           BISB    @TADB,R0                ;READ ONE BYTE
011130 110021                           MOVB    R0,(R1)+                ;AND PUT IT IN THE BUFFER
011132 004737 012102                    JSR     PC,@#DO.CRC             ;CALCULATE CRC FOR THIS BYTE
011136 012777 012340 170064             MOV     #SERV,@TAVEC            ;SETUP TO SERVICE THE INTERRUPT
011144 000743                           BR      3$                      ;LOOP
011146 010237 001166            8$:     MOV     R2,@#$TMP0              ;GET NUMBER OF BYTES NOT READ
011152 001414                           BEQ     9$                      ;SKIP OUT IF 0
011154 013746 001172                    MOV     @#$ESCAPE,-(SP)         ;SAVE THE ESCAPE ADDRESS
011160 005037 001172                    CLR     @#$ESCAPE               ;NO ESCAPE DESIRED
011164 016646 000004                    MOV     4(SP),-(SP)             ;COPY THE STACK
011170 016646 000004                    MOV     4(SP),-(SP)
011174 104013                           ERROR   13                      ;BLOCK READ SHORT
011176 012637 001172                    MOV     (SP)+,@#$ESCAPE         ;RESTORE ESCAPE ADDRESS
011202 000002                           RTI                             ;LEAVE
011204 104002                   9$:     ERROR   2                       ;ERROR OCCURRED DURING READ
011206 013737 012222 001212     5$:     MOV     @#CRC.WD,@#RCRC1        ;SAVE THE CALCULATED CRC
011214 004037 012224                    JSR     R0,@#DO.CMD             ;DO AN ILBS
011220 000124                           XREAD!INT.EN!ILBS
011222 111537 001214                    MOVB    @TADB,@#RCRC2           ;SAVE 1ST BYTE OF CRC FROM TA11
011226 005037 177776                    CLR     @#PS                    ;ALLOW INTERRUPTS
011232 004037 011716                    JSR     R0,@#WAITFLAG           ;WAIT ON FLAG
011236 104003                           ERROR   3                       ;NONE OCCURRED
011240 104004                           ERROR   4                       ;INTERRUPT FAILED
011242 111537 001215                    MOVB    @TADB,RCRC2+1           ;SAVE 2ND BYTE OF CRC FROM TA11
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     #LEADER!WRTLOCK!FGAP!TIMERR!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  011304  011637  001204            T.BSFG: MOV     (SP),@#SAVPC            ;SAVE PC OF CALL
2649  011310  012737  015433  007304            MOV     #MXBSFG,@#MFUNC         ;SAVE POINTER OF .ASCIZ NAME
2650  011316  004037  012224            1$:     JSR     R0,@#DO.CMD             ;START A "BSFG"
2651  011322  000107                            XBSFG!INT.EN!GO
2652  011324  005037  177776                    CLR     @#PS                    ;A   W INTERRUPTS
2653  011330  004037  011716                    JSR     R0,@#WAITFLAG           ;   AIT ON FLAG
2654  011334  104003                            ERROR   3                       ;NO FLAG
2655  011336  104004                            ERROR   4                       ;FLAG DIDN'T INTERRUPT
2656  011340  004037  012014                    JSR     R0,@#FLAGS              ;FIND OUT WHAT FLAG OCCURRED
2657  011344  104002                            ERROR   2                       ;UNKNOWN
2658  011346  104002                            ERROR   2                       ;"XFER REQ"
2659  011350  000401                            BR      2$                      ;"READY"
2660  011352  104002                            ERROR   2                       ;"READY" & "ERROR"
2661  011354  032714  004000            2$:     BIT     #FGAP,@TACS             ;IN A FILE GAP?
2662  011360  001401                            BEQ     3$                      ;BR IN NO
2663  011362  000002                            RTI                             ;GO BACK
2664  011364  104006                    3$:     ERROR   6                       ;"BSFG" DIDN'T STOP IN A FILE  GAP
2665
2666
2667                                    ;;*****************************************************************
2668                                    ;THIS ROUTINE WILL BACK SPACE ONE BLOCK GAP
2669  011366  011637  001204            T.BSBG: MOV     (SP),@#SAVPC            ;SAVE THE PC OF THE CALL
2670  011372  012737  015457  007304            MOV     #MXBSBG,@#MFUNC         ;MESSAGE POINTER
2671  011400  004037  012224            1$:     JSR     R0,@#DO.CMD             ;START A "BSBG"
2672  011404  000111                            XBSBG!INT.EN!GO
2673  011406  005037  177776                    CLR     @#PS                    ;ALLOW INTERRUPTS
2674  011412  004037  011716                    JSR     R0,@#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     R0,@#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  011436  011637  001204            .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$:     JSR     R0,@#DO.CMD             ;START A "SFFG"
2687  011454  000113                            XSFFG!INT.EN!GO
2688  011456  005037  177776                    CLR     @#PS                    ;ALLOW INTERRUPTS
2689  011462  004037  011716                    JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
2690  011466  104003                            ERROR   3                       ;FLAG FAILED TO OCCUR
2691  011470  104004                            ERROR   4                       ;FLAG DIDN'T INTERRUPT
2692  011472  004037  012014                    JSR     R0,@#FLAGS              ;WHAT FLAG OCCURRED
2693  011476  104002                            ERROR   2                       ;UNKNOWN
2694  011500  104002                            ERROR   2                       ;"XFER REQ"
2695  011502  000401                            BR      2$                      ;"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$:     ERROR   6                       ;"SFFG" DIDN'T STOP IN A FILE  GAP
2701
2702
2703                                    ;;*****************************************************************
2704                                    ;THIS ROUTINE WILL SPACE FORWARD ONE BLOCK GAP
2705  011520  011637  001204            .SFBG:  MOV     (SP),@#SAVPC            ;SAVE PC OF CALL
2706  011524  012737  015411  007304            MOV     #MXSFBG,@#MFUNC         ;POINTER TO .ASCIZ MESSAGE
2707  011532  004037  012224            1$:     JSR     R0,@#DO.CMD             ;START A "SFBG"
2708  011536  000115                            XSFBG!INT.EN!GO
2709  011540  005037  177776                    CLR     @#PS                    ;ALLOW INTERRUPTS
2710  011544  004037  011716                    JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
2711  011550  104003                            ERROR   3                       ;NO FLAG
2712  011552  104004                            ERROR   4                       ;NO INTERRUPT
2713  011554  004037  012014                    JSR     R0,@#FLAGS              ;FIND OUT WHAT FLAG
2714  011560  104002                            ERROR   2                       ;UNKNOWN
2715  011562  104002                            ERROR   2                       ;"XFER REQ"
2716  011564  000002                            RTI                             ;"READY"
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"
```

```
2721                                    ;;*****************************************************************
2722                                    ;THIS ROUTINE WILL REWIND THE TAPE
2723  011600  011637  001204             T.RWND: MOV     (SP),@#SAVPC            ;SAVE PC OF CALL
2724  011604  012737  015344  007304             MOV     #MXRWND,@#MFUNC         ;POINTER TO .ASCIZ MESSAGE
2725  011612  004037  012014                     JSR     R0,@#FLAGS              ;CHECK THE FLAGS
2726  011616  104001                             ERROR   1                       ;NO FLAGS
2727  011620  104001                             ERROR   1                       ;"XFER REQ"
2728  011622  000411                             BR      1$                      ;NORMAL RETURN
2729  011624  005777  167426                     TST     @RWDFLG                 ;WAS LAST FUNCTION A REWIND?
2730  011630  001406                             BEQ     1$                      ;BR IF YES
2731  011632  032714  020000                     BIT     #LEADER,@TACS           ;ON CLEAR LEADER?
2732  011636  001403                             BEQ     1$                      ;BR IF NO
2733  011640  005077  167412                     CLR     @RWDFLG                 ;ALLOW THIS ERROR ONLY ONE TIME
2734  011644  104001                             ERROR   1                       ;ON CLEAR LEADER BUT SHOULDN'T BE
2735  011646  004037  012224             1$:     JSR     R0,@#DO.CMD             ;GO START A "REWIND"
2736  011652  000117                             XRWND!INT.EN!GO
2737  011654  005037  177776                     CLR     @#PS                    ;ALLOW INTERRUPTS
2738  011660  004037  011716                     JSR     R0,@#WAITFLAG           ;WAIT ON FLAG
2739  011664  104003                             ERROR   3                       ;NO FLAG OCCURRED
2740  011666  104004                             ERROR   4                       ;NO INTERRUPT
2741  011670  004037  012014                     JSR     R0,@#FLAGS              ;WHAT FLAG OCCURRED
2742  011674  104002                             ERROR   2                       ;UNKNOWN
2743  011676  104002                             ERROR   2                       ;"XFER REQ"
2744  011700  000401                             BR      2$                      ;"READY"
2745  011702  104002                             ERROR   2                       ;"READY" & "ERROR"
2746  011704  032714  020000             2$:     BIT     #LEADER,@TACS           ;ON "CLEAR LEADER"?
2747  011710  001401                             BEQ     3$                      ;BR IF NO
2748  011712  000002                             RTI                             ;YES---RETURN
2749  011714  104007             3$:     ERROR   7                       ;"REWIND" FAILED TO GO TO "CLEAR LEADER"
```

```
2750                                    ;*************************  ****************************************
2751                                    ;THIS ROUTINE IS USED TO WAIT ON A FLAG
2752                                    ;
2753                                    ;CALL:
2754                                    ;       JSR     R0,@#WAITFLAG
2755                                    ;       RETURN  FOR NO FLAGS
2756                                    ;       RETURN  FOR NO INTERRUPT
2757                                    ;       NORMAL  RETURN
2758                                    ;
2759  011716                            WAITFLAG:
2760  011716  005037  012006            CLR     LOCNT                   ;ZERO THE LOW COUNTER
2761  011722  013737  012012  012010    MOV     MAXCNT,HICNT            ;SET THE HIGH COUNTER TO MAX.COUNT
2762  011730  032777  000040  167202    BIT     #SW05,@SWR              ;RUNNING WITH INTERRUPTS?
2763  011736  001403                    BEQ     1$                      ;BR IF YES
2764  011740  012737  177777  001254    MOV     #-1,WAITKEY             ;SET THE INTERRUPT OCCURRED FLAG
2765  011746  032714  000240            1$:     BIT     #TR.REQ!READY,@TACS     ;TEST FOR FLAGS
2766  011752  001007                    BNE     2$                      ;BR IF A FLAG IS UP
2767  011754  005237  012006            INC     LOCNT                   ;COUNT THE LOW COUNTER
2768  011760  001372                    BNE     1$                      ;LOOP IF NOT ZERO
2769  011762  005337  012010            DEC     HICNT                   ;COUT THE HIGH COUNTER
2770  011766  100367                    BPL     1$                      ;LOOP IF NOT NEG.
2771  011770  000405                    BR      3$                      ;LEAVE
2772  011772  005720                    2$:     TST     (R0)+           ;INCREMENT THE RETURN ADDRESS
2773  011774  005737  001254            TST     WAITKEY                 ;LOOK AT THE WAIT ON INT. KEY
2774  012000  001401                    BEQ     3$                      ;BR IF NOT SET
2775  012002  005720                    TST     (R0)+                   ;INCREMENT THE RETURN ADDRESS
2776  012004  000200                    3$:     RTS     R0              ;RETURN TO USER
2777  012006  000000                    LOCNT:  0
2778  012010  000000                    HICNT:  0
2779  012012  000000                    MAXCNT: 0
```

```
2780
2781                                    ;;*****************************************************************
2782
2783                                    ;THIS ROUTINE IS USED TO DETERMINE WHAT FLAG IS SET
2784                                    ;
2785                                    ;CALL:
2786                                    ;       JSR     R0,@#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  16~100           BIT     #SW06,@#SWR             ;STALL?
2799  012040  001414                           BEQ     3$                      ;BR IF NO
2800  012042  010046                           MOV     R0,-(SP)        ;;PUSH R0 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     R0
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL CALCULATE THE CRC
                                ;CALL:
                                ;       JSR     PC,@#DO.CRC             ;R0=1 BYTE OF DATA
                                ;
012102                          DO.CRC:
012102  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
012104  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
012106  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
012110  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
012112  010446                         MOV     R4,-(SP)        ;;PUSH R4 ON STACK
012114  012737  000010  012176         MOV     #8.,3$                  ;MAKE EIGHT ITERATIONS
012122  013703  012222                 MOV     CRC.WD,R3               ;PICKUP THE CRC WORD
012126  005001                 1$:     CLR     R1
012130  010302                         MOV     R3,R2                   ;GET THE PARTIAL CRC WORD
012132  042702  177776                 BIC     #^CBIT00,R2             ;STRIP AWAY EVERYTHING BUT BIT00
012136  006000                         ROR     R0                      ;PULL OFF THIS BIT
012140  006101                         ROL     R1                      ;AND SETUP TO XOR IT
012142  010104                         MOV     R1,R4                   ;FORM THE XOR OF "R1" AND "R2"
012144  040204                         BIC     R2,R4
012146  040102                         BIC     R1,R2
012150  050402                         BIS     R4,R2                   ;RESULTS TO "R2"
012152  006002                         ROR     R2
012154  103006                         BCC     2$
012156  012701  040002                 MOV     #BIT14!BIT01,R1
012162  010104                         MOV     R1,R4                   ;FORM THE XOR OF "R1" AND "R3"
012164  040304                         BIC     R3,R4
012166  040103                         BIC     R1,R3
012170  050403                         BIS     R4,R3                   ;RESULTS TO "R3"
012172  006003                 2$:     ROR     R3
012174  005327                         DEC     (PC)+
012176  000000                 3$:     0
012200  003352                         BGT     1$                      ;BR IF MORE BITS TO DO
012202  010337  012222                 MOV     R3,CRC.WD
012206  012604                         MOV     (SP)+,R4        ;;POP STACK INTO R4
012210  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
012212  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
012214  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
012216  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
012220  000207                         RTS     PC
012222  000000                 CRC.WD: 0                               ;CRC WORD
```

```
                                ;*****************************************************************
                                ;THIS ROUTINE WILL LOAD THE LOW BYTE OF TACS WITH THE DESIRED FUNCTION
                                ;
                                ;CALL:
                                ;       JSR     R0,@#DO.CMD
                                ;       DESIRED FUNCTIN
                                ;       RETURN
                                ;
012224  010146                  DO.CMD: MOV     R1,-(SP)                ;SAVE R1
012226  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT ALL INERRUPTS
012234  012001                          MOV     (R0)+,R1                ;GET THE COMMAND
012236  042701  177640                  BIC     #^C<INT.EN!ILBS!FUNCTION!GO>,R1 ;STRIP AWAY JUNK
012242  032777  000040  166670          BIT     #SWOS,@SWR              ;WITH OR WITHOUT INTERRUPTS
012250  001405                          BEQ     4$                      ;BR IF WITH
012252  042701  000100                  BIC     #INT.EN,R1              ;CLEAR INTERRUPT ENABLE IF SET
012256  012777  012356  166744          MOV     #BADINT,@TAVEC          ;SETUP TO CATCH BAD INTERRUPT
012264  005037  001254          4$:     CLR     WAITKEY                 ;ZERO THE WAIT ON INTERRUPT KEY
012270  032701  000100                  BIT     #INT.EN,R1              ;RUNNING WITH INTERPUPTS?
012274  001403                          BEQ     3$                      ;BR IF NO
012276  012777  012340  166724          MOV     #SERV,@TAVEC            ;SETUP TO SERVICE THE INTERRUPT
012304  110114                  3$:     MOVB    R1,@TACS                ;LOAD THE COMMAND
012306  032701  000001                  BIT     #GO,R1                  ;IS GO BIT ON A "1"?
012312  001403                          BEQ     2$                      ;BR IF GO =0
012314  032714  000040          1$:     BIT     #READY,@TACS            ;WAIT FOR "READY" TO CLEAR
012320  001375                          BNE     1$
012322  005101                  2$:     COM     R1                      ;SET OR CLEAR REWIND FLAG
012324  042701  177761                  BIC     #^C<FUNCTION>,R1        ;EXTRACT FUNCTION BITS
012330  010177  166722                  MOV     R1,@RWDFLG              ;AND SAVE AS THE REWIND FLAG
012334  012601                          MOV     (SP)+,R1                ;GET R1 BACK
012336  000200                          RTS     R0                      ;RETURN TO USER

                                ;*****************************************************************
                                ;THIS ROUTINE WILL SERVICE ALL LEGAL INTERRUPTS
012340  012737  177777  001254  SERV:   MOV     #-1,@#WAITKEY           ;SET THE WAIT KEY
012346  012777  012356  166654          MOV     #BADINT,@TAVEC          ;SET TO CATCH ILLEGAL INTERRUPTS
012354  000002                          RTI

                                ;*****************************************************************
                                ;THIS ROUTINE WILL CATCH ILLEGAL INTERRUPTS FROM THE TA11
012356  011637  001204          BADINT: MOV     (SP),@#SAVPC            ;SAVE WHERE WE WERE AT
012362  104014                          ERROR   14                      ;ILLEGAL INTERRUPT OCCURRED
```

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



                                ;;*****************************************************************
                                ;       WRITE FILE GAPS FROM "BOT" TO "EOT"
                                        ;START AT 220
                                        ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND
                                        ;THE "WRITE DELAY MONO".
                                ;*****************************************************************
012364  012706  001100          WFGSUB: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
012370  013704  001220                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
012374  013705  001224                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
012400  000005                          RESET                           ;RESET THE WORLD
012402  012737  012364  001110          MOV     #WFGSUB,@#$LPERR        ;SETUP THE LOOP ON ERROR ADDRESS
012410  104422                          SELDRV                          ;SELECT THE DRIVE
012412  112714  000017                  MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
012416  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
012422  001775                          BEQ     100$
012424  104412                  1$:     WFG                             ;WRITE A FILE GAP
012426  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012432  001774                          BEQ     1$                      ;BR IF NO
012434  000000                          HALT                            ;STOP IF YES
012436  000752                          BR      WFGSUB                  ;LOOP ON CONT.
                                ;;*****************************************************************
                                ;       WRITE CONTINUOUS BLOCKS OF DATA
                                        ;START AT 224
                                        ;THE PROGRAM WILL HALT THREE(3) TIMES
                                        ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
                                        ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
                                        ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
                                        ;HALT 3 --- SWR<15:0> = OPTIONAL SWITCH SETTINGS
                                        ;THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"
                                        ;THIS ROUTINE SUPPORTS THE S W SWITCH REG FUNCTIONS
                                ;;*****************************************************************
012440  004737  013106          WRTSUB: JSR     PC,@#SETBUF             ;GET BLOCK SIZE AND PATTERN
012444  012706  001100          WLOOP:  MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
012450  013704  001220                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
012454  013705  001224                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
012460  000005                          RESET                           ;RESET THE WORLD
012462  012737  012444  001110          MOV     #WLOOP,@#$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
012470  104422                          SELDRV                          ;SELECT THE DRIVE
012472  112714  000017                  MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
012476  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
```

```
2954  012502  001775                    BEQ     100$
2955  012504  104413  017042      1$:   WRITE   WLNKX                   ;WRITE A BLOCK
2956  012510  032714  020000            BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2957  012514  001773                    BEQ     1$                      ;BR IF NO
2958  012516  000000                    HALT                            ;STOP IF "EOT"
2959  012520  000751                    BR      WLOOP                   ;LOOP IF CONT.
2960
2961
2962                          ;;*****************************************************************
2963                          ;       READ AND COMPARE CONTINUOUS BLOCKS OF DATA
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  012522  001110  MOV     #RDSUB,@#$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
2973  012546  104422                  SELDRV                          ;SELECT THE DRIVE
2974  012550  112714  000017          MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
2975  012554  032714  000040  100$:   BIT     #READY,@TACS            ;WAIT ON READY
2976  012560  001775                  BEQ     100$
2977  012562  104414  017006  1$:    READ    RLNKX                   ;READ A BLOCK
2978  012566  012706  001100          MOV     #STACK,SP               ;INIT. IN CASE OF ERROR
2979  012572  032714  020000          BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2980  012576  001351                  BNE     RDSUB                   ;BR IF YES---LOOP
2981  012600  104423                  RLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
2982  012602  017006                  RLNKX                           ;LINK TO THE READ BUFFER
2983  012604  017042                  WLNKX                           ;LINK TO THE WRITE BUFFER
2984  012606  000765                  BR      1$
2985
2986
2987                          ;;*****************************************************************
2988                          ;       WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
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".
```

```
                                        ;THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
                                ;;*******************************************************************
012610  004737  013106          WGPBLK: JSR     PC,@#SETBUF             ;GET BLOCK SIZE AND PATTERN
012614  012706  001100          WGBLOP: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
012620  013704  001220                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
012624  013705  001224                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
012630  000005                          RESET                           ;RESET THE WORLD
012632  012737  012614  001110          MOV     #WGBLOP,@#$LPERR        ;SETUP THE LOOP ON ERROR ADDRESS
012640  104422                          SELDRV                          ;SELECT THE DRIVE
012642  112714  000017                  MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
012646  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
012652  001775                          BEQ     100$
012654  104412                  1$:     WFG                             ;WRITE A FILE GAP
012656  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012662  001005                          BNE     2$                      ;BR IF YES
012664  104413  017042                  WRITE   WLNKX                   ;WRITE A BLOCK
012670  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012674  001767                          BEQ     1$                      ;BR IF NO
012676  000000                  2$:     HALT                            ;STOP AT "EOT"
012700  000745                          BR      WGBLOP                  ;START OVER ON CONT.


                                ;;*******************************************************************
                                ;       READ AND COMPARE A BLOCK OF DATA AND A FILE GAP
                                        ;START AT 240
                                        ;THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
                                        ;IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
                                        ;AND THE "TAPE BLANK MONO".
                                ;;*******************************************************************
012702  012706  001100          RGPBLK: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
012706  013704  001220                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
012712  013705  001224                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
012716  000005                          RESET                           ;RESET THE WORLD
012720  012737  012702  001110          MOV     #RGPBLK,@#$LPERR        ;SETUP THE LOOP ON ERROR ADDRESS
012726  104422                          SELDRV                          ;SELECT THE DRIVE
012730  112714  000017                  MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
012734  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
012740  001775                          BEQ     100$
012742  104414  017006          1$:     READ    RLNKX                   ;READ A BLOCK OF DATA
012746  012706  001100                  MOV     #STACK,SP               ;KEEP OUT OF THE WAY
012752  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012756  001351                          BNE     RGPBLK                  ;BR IF YES
012760  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
012762  017006                          RLNKX                           ;LINK TO THE READ BUFFER
012764  017042                          WLNKX                           ;LINK TO THE WRITE BUFFER
012766  104420                          SFBG                            ;GET INTO FILE GAP
012770  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012774  001342                          BNE     RGPBLK                  ;BR IF YES
012776  000761                          BR      1$                      ;LOOP


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

```
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     @#TACS,TACS             ;SETUP THE TA11 STATUS AND
3058  013010  013705  001224              MOV     @#TADB,TADB             ;DATA BUFFER REGISTERS
3059  013014  000005                      RESET                           ;RESET THE WORLD
3060  013016  012737  013000  001110      MOV     #SFFGSB,@#$LPERR        ;SETUP THE LOOP ON ERROR ADDRESS
3061  013024  104422                      SELDRV                          ;SELECT THE DRIVE
3062  013026  112714  000017              MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
3063  013032  032714  000040      100$:   BIT     #READY,@TACS            ;WAIT ON READY
3064  013036  001775                      BEQ     100$
3065  013040  104417              1$:     SFFG                            ;SPACE INTO A FILE GAP
3066  013042  032714  020000              BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
3067  013046  001774                      BEQ     1$                      ;BR IF NO
3068  013050  000000                      HALT                            ;STOP AT "EOT"
3069  013052  000752                      BR      SFFGSB                  ;LOOP ON CONT.
3070
3071
3072                                ;;**********************************************************************
3073                                ;       BACK SPACE FILE GAP
3074                                ;START AT 250
3075                                ;THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
3076                                ;;********************************** **********************************
3077  013054  000005              BSFGSB: RESET                           ;RESET THE WORLD
3078  013056  012737  013054  001110      MOV     #BSFGSB,@#$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      BSFGSB                  ;START OVER ON CONT.
3086
3087
3088                                ;;**********************************************************************
3089                                ;       SETUP READ AND WRITE LINKS FOR SUBROUTINES
3090  013106  005000              SETBUF: CLR     R0
3091  013110  000000                      HALT                            ;OPERATOR PUTS BYTE COUNT IN THE SWR
3092  013112  022737  000176  001140      CMP     #SWREG,SWR              ;USING S/W SWITCH REG?
3093  013120  001001                      BNE     20$                     ;NO- GET OUT
3094  013122  104405                      GTSWR                           ;GET VALUE
3095  013124                      20$:                                    ;CONTINUE
3096  013124  157700  166010              BISB    @SWR,R0         ;PICKUP THE BYTE COUNT
3097  013130  001006                      BNE     2$                      ;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     #WBUFX,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
                                ;*

013230  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
013234  100002                          BPL     1$              ;;BR IF YES
013236  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
013240  000'07                          BR      3$              ;;LEAVE
013242  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
013244  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
013250  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
013252  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
013254  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
013256  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
013260  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
013264  000002                          RTI                     ;;RETURN
013266  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
013272  001430                          BEQ     8$
013274  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
013300  001006                          BNE     5$
013302  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
013304  104401                          TYPE                    ;;TYPE A CR AND LF
013306  001201                          $CRLF
013310  105037  013444                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
013314  000755                          BR      2$              ;;GET NEXT CHARACTER
013316  004737  013400          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
013322  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
013326  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
013330  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
013334  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED
013340  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
013342  004737  013400                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
013346  105337  013444                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
013352  000770                          BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

013354  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
013360  004737  013400          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
013364  132737  000007  013444          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
013372  001372                          BNE     9$              ;;TAB STOP
013374  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
```

```
3177  013376  000724                          BR      2$              ;;GET NEXT CHARACTER
3178  013400  105777  165544          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
3179  013404  100375                          BPL     $TYPEC
3180  013406  116677  000002  165536          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3181  013414  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
3182  013422  001003                          BNE     1$              ;;BRANCH IF NO
3183  013424  105037  013444                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
3184  013430  000406                          BR      $TYPEX          ;;EXIT
3185  013432  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
3186  013440  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
3187  013442  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
3188  013444  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
3189  013446  000207          $TYPEX: RTS     PC
3190
3191                          .SBTTL  TTY INPUT ROUTINE
3192
3193                          ;;**************************************************************************
3194                          .ENABL  LSB
3195
3196                          ;;**************************************************************************
3197                          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3198                          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
3199                          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
3200                          ;*WHEN OPERATING IN TTY FLAG MODE.
3201  013450  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
3202  013456  001074                          BNE     15$             ;;BRANCH IF NO
3203  013460  105777  165460                  TSTB    @$TKS           ;;CHAR THERE?
3204  013464  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
3205  013466  117746  165454                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
3206  013472  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
3207  013476  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
3208  013502  001062                          BNE     15$             ;;NO, RETURN TO USER
3209  013504  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
3210  013512  001456                          BEQ     15$             ;;BRANCH IF YES
3211
3212  013514  104401  014175                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
3213  013520  104401  014202  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
3214  013524  013746  000176          MOV     $SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT
3215  013530  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3216  013532  104401  014213          TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
3217  013536  005046          19$:    CLR     -(SP)           ;;CLEAR COUNTER
3218  013540  005046                  CLR     -(SP)           ;;THE NEW SWR
3219  013542  105777  165376  7$:     TSTB    @$TKS           ;;CHAR THERE?
3220  013546  100375                  BPL     7$              ;;IF NOT TRY AGAIN
3221
3222  013550  117746  165372          MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
3223  013554  042716  177600          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3224
3225
3226
3227  013560  021627  000025  9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
3228  013564  001005                  BNE     10$             ;;BRANCH IF NOT
3229  013566  104401  014170                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
3230  013572  062706  000006  20$:     ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
3231  013576  000757                  BR      19$             ;;LET'S TRY IT AGAIN
3232
```

```
3233
3234  013600  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
3235  013604  001022                          BNE     16$             ;;BRANCH IF NO
3236  013606  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
3237  013612  001403                          BEQ     11$             ;;BRANCH IF YES
3238  013614  016677  000002  165316          MOV     2(SP),@SWR      ;;SAVE NEW SWR
3239  013622  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
3240  013626  104401  001201          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
3241  013632  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
3242  013640  001003                          BNE     15$             ;;BRANCH IF NOT
3243  013642  012777  000100  165274          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
3244  013650  000002          15$:    RTI                             ;;RETURN
3245  013652  004737  013400          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
3246  013656  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
3247  013662  002420                          BLT     18$             ;;BRANCH IF YES
3248  013664  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
3249  013670  003015                          BGT     18$             ;;BRANCH IF YES
3250  013672  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
3251  013676  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
3252  013702  001403                          BEQ     17$             ;;BRANCH IF YES
3253  013704  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
3254  013706  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
3255  013710  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
3256  013712  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
3257  013716  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
3258  013722  000707                          BR      7$              ;;GET THE NEXT ONE
3259  013724  104401  001200          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
3260  013730  000720                          BR      20$             ;;SIMULATE CONTROL-J
3261                                  .DSABL  LSB
3262
3263
3264                                  ;;*****************************************************************
3265                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3266                                  ;*CALL:
3267                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
3268                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
3269                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
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    @$TKS           ;;WAIT FOR
3275  013746  100375                          BPL     1$              ;;A CHARACTER
3276  013750  117766  165172  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
3277  013756  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
3278  013764  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
3279  013772  001013                          BNE     3$              ;;BRANCH IF NO
3280  013774  105777  165144          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
3281  014000  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
3282  014002  117746  165140                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
3283  014006  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3284  014012  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
3285  014016  001366                          BNE     2$              ;;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                  4$:     RTI                     ;;GO BACK TO USER
3293                                          ;;**********************************************************************
3294                                          ;*THIS ROUTINE WILL INPJT A STRING FROM THE TTY
3295                                          ;*CALL:
3296                                          ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
3297                                          ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3298                                          ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
3299
3300  014052  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
3301  014054  012703  014160          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
3302  014060  022703  014170          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
3303  014064  101405                          BLOS    4$              ;;BR IF YES
3304  014066  104407                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
3305  014070  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
3306  014072  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
3307  014076  001003                          BNE     3$              ;;SKIP IF NOT
3308  014100  104401  001200          4$:     TYPE    ,$QUES          ;;TYPE A '?'
3309  014104  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
3310  014106  111337  014156          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
3311  014112  104401  014156                  TYPE    ,9$
3312  014116  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
3313  014122  001356                          BNE     2$              ;;LOOP IF NOT RETURN
3314  014124  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
3315  014130  104401  001202                  TYPE    ,$LF            ;;TYPE A LINE FEED
3316  014134  012603                          MOV     (SP)+,R3        ;;RESTORE R3
3317  014136  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3318  014140  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
3319  014146  012766  014160  000004          MOV     #$TTYIN,4(SP)
3320  014154  000002                          RTI                     ;;RETURN
3321  014156     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
3322  014157     000                          .BYTE   0               ;;TERMINATOR
3323  014160  000010                  $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
3324  014170  052536  005015     000  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
3325  014175     136  006507  000012  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
3326  014202  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
3327  014210  020075     000
3328  014213     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
3329  014220  036440  000040
3330                                  .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
3331
3332                                          ;;**********************************************************************
3333                                          ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
3334                                          ;*CHANGE IT TO BINARY.
3335                                          ;*CALL:
3336                                          ;*      RDOCT                   ;;READ AN OCTAL NUMBER
3337                                          ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
3338                                          ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
3339
3340  014224  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
3341  014226  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
3342  014234  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3343  014236  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3344  014240  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
```

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

```
                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                ;;*****************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                ;*      .BYTE   M               ;;M=1 OR 0
                                ;*                                      ;;1=TYPE LEADING ZEROS
                                ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                ;*
                                ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                ;*$TYPOS OR $TYPOC
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPON                   ;;CALL FOR TYPEOUT
                                ;*
                                ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOC                   ;;CALL FOR TYPEOUT

014326  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
014332  116637  000001  014551          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
014340  112637  014553                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
014344  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
014350  000406                          BR      $TYPON
014352  112737  000001  014551  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
014360  112737  000006  014553          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
014366  112737  000005  014550  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
014374  010346                          MOV     R3,-(SP)        ;;SAVE R3
014376  010446                          MOV     R4,-(SP)        ;;SAVE R4
014400  010546                          MOV     R5,-(SP)        ;;SAVE R5
014402  113704  014553                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
014406  005404                          NEG     R4
014410  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
014414  110437  014552                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
014420  113704  014551                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
014424  016605  000012                  MOV     12(SP),R5       ;;PICK UP THE INPUT NUMBER
014430  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
014432  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
014434  000404                          BR      3$              ;;GO DO MSB
014436  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
014440  006105                          ROL     R5
014442  006105                          ROL     R5
014444  010503                          MOV     R5,R3
014446  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
014450  105337  014552                  DECB    $OMODE          ;;TYPE THIS DIGIT?
014454  100016                          BPL     7$              ;;BR IF NO
014456  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
014462  001002                          BNE     4$              ;;TEST FOR 0
014464  005704                          TST     R4              ;;SUPPRESS THIS 0?
014466  001403                          BEQ     5$              ;;BR IF YES
```

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

```
                                .SBTTL  TRAP DECODER

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

014554  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
014556  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
014562  005740                          TST     -(R0)           ;;BACKUP BY 2
014564  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
014566  006300                          ASL     R0              ;;POSITION FOR INDEXING
014570  016000  014610                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
014574  000200                          RTS     R0              ;;GO TO ROUTINE

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

014576  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
014600  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
014606  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
014610  014576                  $TRPAD: .WORD   $TRAP2
014612  013230                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
014614  014352                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
014616  014326                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
014620  014366                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)

014622  013520                          $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING

014624  013450                          $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
014626  013732                          $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
014630  014052                          $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
014632  014224                          $RDOCT  ;;CALL=RDOCT    TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
014634  010504                          T.WFG   ;;CALL=WFG      TRAP+12(104412) ROUTINE TO WRITE A FILE GAP
014636  010554                          T.WRIT  ;;CALL=WRITE    TRAP+13(104413) ROUTINE TO WRITE A BLOCK OF DATA
014640  010774                          T.READ  ;;CALL=READ     TRAP+14(104414) ROUTINE TO READ A BLOCK OF DATA
014642  011304                          T.BSFG  ;;CALL=BSFG     TRAP+15(104415) ROUTINE TO BACK SPACE A FILE GAP
014644  011366                          T.BSBG  ;;CALL=BSBG     TRAP+16(104416) ROUTINE TO BACK SPACE A BLOCK GAP
014646  011436                          T.SFFG  ;;CALL=SFFG     TRAP+17(104417) ROUTINE TO SPACE FWD A FILE GAP
014650  011520                          T.SFBG  ;;CALL=SFBG     TRAP+20(104420) ROUTINE TO SPACE FWD A BLOCK GAP
014652  011600                          T.RWND  ;;CALL=REWIND   TRAP+21(104421) ROUTINE TO REWIND THE TAPE
014654  010246                          T.SELDRV        ;;CALL=SELDRV   TRAP+22(104422) ROUTINE TO SELECT THE NEXT DRIVE
014656  010362                          T.BLKCMP        ;;CALL=BLKCMP   TRAP+23(104423) COMPARE READ AND WRITE BUFFERS
014660  010106                          T.SETLOOP       ;;CALL=SETLOOP  TRAP+24(104424) SETUP TO LOOP AS PER THE TTY
```

```
                                        .SBTTL  POWER DOWN AND UP ROUTINES
                                        ;;*****************************************************************
                                        ;POWER DOWN ROUTINE
014662  012737  015026  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
014670  012737  000340  000026         MOV     #340,@#PWRVEC+2  ;;PRIO:7
014676  010046                       MOV     R0,-(SP)         ;;PUSH R0 ON STACK
014700  010146                       MOV     R1,-(SP)         ;;PUSH R1 ON STACK
014702  010246                       MOV     R2,-(SP)         ;;PUSH R2 ON STACK
014704  010346                       MOV     R3,-(SP)         ;;PUSH R3 ON STACK
014706  010446                       MOV     R4,-(SP)         ;;PUSH R4 ON STACK
014710  010546                       MOV     R5,-(SP)         ;;PUSH R5 ON STACK
014712  017746  164222               MOV     @SWR,-(SP)       ;;PUSH @SWR ON STACK
014716  010637  015032               MOV     SP,$SAVR6        ;;SAVE SP
014722  012737  014734  000024       MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
014730  000000                       HALT
014732  000776                       BR      .-2              ;;HANG UP
                                        ;;*****************************************************************
                                        ;POWER UP ROUTINE
014734  012737  015026  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
014742  013706  015032               MOV     $SAVR6,SP        ;;GET SP
014746  005037  015032               CLR     $SAVR6           ;;WAIT LOOP FOR THE TTY
014752  005237  015032         1$:   INC     $SAVR6           ;;WAIT FOR THE INC
014756  001375                       BNE     1$               ;;OF  WORD
014760  012677  164154               MOV     (SP)+,@SWR       ;;POP STACK INTO @SWR
014764  012605                       MOV     (SP)+,R5         ;;POP STACK INTO R5
014766  012604                       MOV     (SP)+,R4         ;;POP STACK INTO R4
014770  012603                       MOV     (SP)+,R3         ;;POP STACK INTO R3
014772  012602                       MOV     (SP)+,R2         ;;POP STACK INTO R2
014774  012601                       MOV     (SP)+,R1         ;;POP STACK INTO R1
014776  012600                       MOV     (SP)+,R0         ;;POP STACK INTO R0
015000  012737  014662  000024       MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
015006  012737  000340  000026       MOV     #340,@#PWRVEC+2  ;;PRIO:7
015014  104401                       TYPE                     ;REPORT THE POWER FAILUPE
015016  015034         $PWRMG: .WORD  $POWER                   ;;POWER FAIL MESSAGE POINTER
015020  012716                       MOV     (PC)+,(SP)       ;;RESTART AT PWRST
015022  002300         $PWRAD: .WORD  PWRST                    ;;RESTART ADDRESS
015024  000002                 RTI
015026  000000         $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
015030  000776                 BR      .-2                     ;;  BEFORE THE POWER DOWN WAS COMPLETE
015032  000000         $SAVR6: 0                               ;;PUT THE SP HERE
015034  005015  047520  042527 $POWER: .ASCIZ  <15><12>"POWER"
015042  000122
                                        .EVEN
```

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

```
3564                                    ;;*****************************************************************
3565
3566                                    ;MESSAGES
3567
3568  015136  042200  044522  042526  MSGDRV: .ASCIZ  <CRLF>"DRIVE(S)? "
3569  015144  051450  037451  000040
3570  015152  005015  040524  051503  MSGASK: .ASCIZ  <15><12>/TACS?/
3571  015160  000077
3572  015162  042526  052103  051117  MSGVEC: .ASCIZ  /VECTOR?/
3573  015170  000077
3574  015172  051120  047511  044522  MSGPRI: .ASCIZ  /PRIORITY?/
3575  015200  054524  000077
3576  015204  052040  041501  036523  MTACS:  .ASCIZ  / TACS=/
3577  015212     000
3578  015213     040  040524  041104  MTADB:  .ASCIZ  / TADB=/
3579  015220  000075
3580  015222  053040  041505  047524  MTAVEC: .ASCIZ  / VECTOR=/
3581  015230  036522     000
3582  015233     040  051120  047511  MTAPRI: .ASCIZ  / PRIORITY=/
3583  015240  044522  054524  000075
3584  015246  005015  045517  000077  MSGOK:  .ASCIZ  <15><12>/OK?/
3585  015254  042531  100123     000  MYES:   .ASCIZ  /YES/<CRLF>
3586  015261     015  052012  051505  MTSTDRV:        .ASCIZ  <15><12>/TESTING DRIVE /
3587  015266  044524  043516  042040
3588  015274  044522  042526  000040
3589  015302  040440  042116  042040  MANDRV: .ASCIZ  / A:D 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  04    _1  026505  MXSFFG: .ASCIZ  /SPACE-FWD-FILE-GAP/
3601  015374  053506  026504  041506
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                  DH1:    .ASCII  <15><12>
3667  016104  042524  052123  020040          .ASCII  /TEST    ERROR/<15><12>
3668  016112  020040  051105  047522
3669  016120  006522     012
3670  016123     120  020103  020040          .ASCIZ  /PC      PC      TACS    TADB/<15><12>
3671  016130  020040  050040  020103
3672  016136  020040  020040  052040
3673  016144  041501  020123  020040
3674  016152  052040  042101  006502
3675  016160  000012
```

```
3676  016162  005015                    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  016300  020040  040504  040524
3691  016306  020040  020040  052516
3692  016314  041115  051105  005015
3693  016322     000
3694  016323     015     012                DH3:    .ASCII  <15><12>
3695  016325     124  051505  020124               .ASCII  /TEST   ERROR          WRITE   READ   TU60/<15><12>
3696  016332  020040  042440  051122
3697  016340  051117  020040  020040
3698  016346  020040  020040  020040
3699  016354  053440  044522  042524
3700  016362  020040  051040  040505
3701  016370  020104  020040  052040
3702  016376  033125  006460     012
3703  016403     120  020103  020040            .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                    .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
```

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

016756  017120  000200        RLNK1:  RBUF1,128.
016762  017322  000200        RLNK2:  RBUF2,128.
016766  017524  000010        RLNK3:  RBUF3,8.
016772  017726  000040        RLNK4:  RBUF4,32.
016776  020030  000020        RLNK5:  RBUF5,16.
017002  020332  000100        RLNK6:  RBUF6,64.
017006  017120  000000        RLNKX:  RBUF1,0
017012  017046  000200        WLNK1:  WBUF1,128.
017016  017054  000200        WLNK2:  WBUF2,128.
017022  017062  000010        WLNK3:  WBUF3,8.
017026  017070  000040        WLNK4:  WBUF4,32.
017032  017076  000020        WLNK5:  WBUF5,16.
017036  017104  000100        WLNK6:  WBUF6,64.
017042  017112  000000        WLNKX:  WBUFX,0

                                        ;BUFFERS

017046  000002                WBUF1:
017046     314                          .BYTE   ↑B11001100
017047     063                          .BYTE   ↑B00110011
017050     314                          .BYTE   ↑B11001100
017051     063                          .BYTE   ↑B00110011
017052  000000                WCRC1:  .WORD   0                       ;CRC STORAGE FOR WBUF1

017054  000002                WBUF2:
017054     377                          .BYTE   ↑B11111111
017055     000                          .BYTE   ↑B00000000
017056     377                          .BYTE   ↑B11111111
017057     000                          .BYTE   ↑B00000000
017060  000000                WCRC2:  .WORD   0                       ;CRC STORAGE FOR WBUF2

017062  000002                WBUF3:
017062     231                          .BYTE   ↑B10011001
017063     146                          .BYTE   ↑B01100110
017064     231                          .BYTE   ↑B10011001
017065     146                          .BYTE   ↑B01100110
017066  000000                WCRC3:  .WORD   0                       ;CRC STORAGE FOR WBUF3

017070  000002                WBUF4:
017070     360                          .BYTE   ↑B11110000
017071     017                          .BYTE   ↑B00001111
017072     360                          .BYTE   ↑B11110000
017073     017                          .BYTE   ↑B00001111
017074  000000                WCRC4:  .WORD   0                       ,CRC STORAGE FOR WBUF4

017076  000002                WBUF5:
017076     252                          .BYTE   ↑B10101010
017077     125                          .BYTE   ↑B01010101
017100     252                          .BYTE   ↑B10101010
017101     125                          .BYTE   ↑B01010101
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                  WBUFX:
 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
```

```
ASKADR  007532            1109    2329#
ASKDRV  007422            1106    2297#   2320
ASKKEY  001246             892#   1196#   1268    2219    2717
A20CT   010022            2404#   2437    2448
BADINT  012356            1200    1260    2869    2889    2895#
BEGINX  002036             817    1103#   1139    1141    1189    1191
BEGIN1  001450             814    1022#
BEGIN2  001502             815    1029#
BEGIN3  001510             816    1028    1031#
BEGIN4  001516            1033#
BGNCMN  001522            1025    1027    1030    1032    1034#
BIT0   = 000001            751#
BIT00  = 000001            741#    751     794    2116    2118    2828
BIT01  = 000002            740#    750     793    2118    2837
BIT02  = 000004            739#    749     792
BIT03  = 000010            738#    748     791
BIT04  = 000020            737#    747     790
BIT05  = 000040            736#    746     789
BIT06  = 000100            735#    745     788
BIT07  = 000200            734#    744     787    2116    2118
BIT08  = 000400            733#    743     786
BIT09  = 001000            732#    742     785    2167    2241
BIT1   = 000002            750#
BIT10  = 002000            731#    784    2225
BIT11  = 004000            730#    783    2174
BIT12  = 010000            729#    782
BIT13  = 020000            728#    781    2232
BIT14  = 040000            727#    780    2149    2837
BIT15  = 100000            726#    779    1238
BIT2   = 000004            749#
BIT3   = 000010            748#
BIT4   = 000020            747#
BIT5   = 000040            746#
BIT6   = 000100            745#
BIT7   = 000200            744#
BIT8   = 000400            743#
BIT9   = 001000            742#
BLKCMP= 104423            1295    1308    1321    1334    1339    1352    1357    1372    1385    1391    1406    1421    1436
                          1452    1467    1482    1486    1493    1500    1507    1513    1517    1530    1536    1543    1550
                          1558    1562    1576    1590    1603    1610    1623    1630    1646    1663    1682    1702    1721
                          1741    1760    1776    1793    1798    1818    1837    1855    1873    1898    1921    1926    1948
                          1954    1959    1964    1969    1997    2002    2033    2039    2046    2053    2981    3039    3496#
BPTVEC= 000014             758#
BSBG   = 104416           1306    1337    1383    1369    1418    1419    1448    1449    1479    1480    1489    1491    1496
                          1498    1503    1505    1510    1511    1528    1534    1539    1541    1546    1548    1553    1555
                          1556    1588    1608    1661    1680    1738    1739    1757    1774    1994    3491#
BSFG   = 104415           1319    1355    1370    1464    1574    1628    1718    1719    1758    1773    1791    1853    1870
                          1893    1894    1895    1924    1946    1951    1952    1957    1992    1993    2031    2036    2037
                          2042    2043    2044    2049    2050    2051    3081    3490#
BSFGSB  013054             824    3077#   3078    3085
BYTNUM  001206             977#   2500#   2507#   3550
CHKADR  002060            1105    1108    1127#
CHKDRV  002140            1165#
CKSWR  = 104406           2139    2206    2240    3483#
CR     = 000015            666#   3181    3191
CRCERR= 040000             780#
```

```
CRC.WD  012222  2539* 2567  2592* 2625  2825  2846* 2853*
CRLF  = 000200   667* 1091  3152  3191  3568  3585
CURDRV  001250   893* 1212* 1214  1221* 1224* 1226  1279  2470*
DDISP = 177570   673*  855  1062
DH1     016102   918   925   932   939   946   953   960  3666*
DH2     016162   967  3676*
DH201   016736  1003  3747*
DH202   016753  1008  3750*
DH3     016323   981  3694*
DH4     016461   988  3711*
DH5     016601   974   995  3726*
DISPLA  001142   855* 1062* 1070* 2188* 2224*
DISPRE  000174   811* 1070
DO.CMD  012224  2521  2544  2568  2593  2626  2650  2671  2686  2707  2735  2862*
DO.CRC  012102  2562  2612  2818*
DRIVE   001236   889* 1209* 2469  2482*
DRVKEY  001240   890*  891  1023* 1165  1178  1199  1217  1223  1261  2298* 2303  2317  2321
                2481
DRVPNT  001244   891* 1179* 1199* 1208  1227*
DSWR  = 177570   672*  854  1061
DT1     015044   920   927   934   941   948   955   962  3547*
DT2     015056   969  3550*
DT201   016652  1004  3735*
DT202   016660  1009  3737*
DT3     015074   983  3554*
DT4     015112   990  3558*
DT5     015126   976   997  3561*
EMTVEC= 000030   761* 1045* 1046*
EM1     015540   917  3622*
EM10    015773   966  3651*
EM11    016011   973  3654*
EM12    016035   980  3658*
EM13    016047   987  3660*
EM14    016064   994  3663*
EM2     015566   924  3626*
EM201   016664  1002  3739*
EM202   016713  1007  3743*
EM3     015615   931  3630*
EM4     015633   938  3633*
EM5     015654   945  3636*
EM6     015705   952  3641*
EM7     015737   959  3646*
ERROR = 100000   779*
ERRVEC= 000004   754* 1059  1060* 1071* 1127* 1133* 2154  2155* 2157* 2160*
EXAM    006464  1166  1170  1175  2103*
EXIT    007522  2316  2321*
FATAL   002702  1236  1255  1274*
FGAP  = 004000   783* 2638  2661  2697
FLAGS   012014  2527  2552  2574  2602  2634  2656  2677  2692  2713  2725  2741  2792*
FUNCTI= 000016   795* 2865  2880
FUNCO = 000002   793*  795
FUNC1 = 000004   792*  795
FUNC2 = 000010   791*  795
GNS   = ****** L  810  1090  1277  3476  3477  3478  3479  3481  3483  3484  3485  3486  3487
                3488  3489  3490  3491  3492  3493  3494  3495  3496  3497
GO    = 000001   794* 1249  2522  2545  2594  2651  2672  2687  2708  2736  2865  2875  2924
```

```
                    2952    2974    3006    3032    3062
GTSWR = 104405      1085    3094    3107    3118    3481#
HERE    002036      1076    1078    1089    1091#
HICNT   012010      1242    2761#   2769#   2777#
HT    = 000011       664#   3150    3191
ILBS  = 000020       790#   2211    2569    2627    2865
INT.EN= 000100       788#   1249    2522    2545    2569    2594    2627    2651    2672    2687    2708    2736    2865
                    2868    2871
IOTVEC= 000020       759#   1043#   1044#
ITEMS2  001430      1000#
LASTAD= 020534      3831#
LEADER= 020000       781#   2121    2638    2731    2746    2928    2955    2979    3010    3013    3037    3043    3065
                    3082
LF    = 000012       665#   3185    3191
LOCNT   012006      2760#   2767#   2777#
LOOP    007446      2304#   2318
LOOPKE  006614      1206#   2447#   2455#   2457#
LVFLAG  012100      2793    2796    2810    2812#
MANDRV  015302      1225    3589#
MANUAL  002260      1185    1193#
MAXCNT  012012      1238#   1242#   1243#   1244#   1245#   1271#   2761    2779#
MBUFCK  015521      2493    3619#
MFUNC   007304      2257#   2493#   2520#   2535#   2582#   2649#   2670#   2685#   2706#   2724#
MLOOP   015331      2442    3593#
MSGASK  015152      2330    3570#
MSGDRV  015136      2297    3568#
MSGID   002024      1079    1091#   1198
MSGOK   015246      2377    3584#
MSGPRI  015172      2353    3574#
MSGVEC  015162      2344    3572#
MTACS   015204      2365    3576#
MTADB   015213      2368    3578#
MTAPRI  015233      2374    3582#
MTAVEC  015222      2371    3580#
MTEST   015316      2429    3591#
MTSTDR  015261      1213    3586#
MXBSBG  015457      2670    3612#
MXBSFG  015433      2649    3608#
MXREAD  015361      2582    3599#
MXRWND  015344      2724    3595#
MXSFBG  015411      2706    3604#
MXSFFG  015366      2685    3600#
MXWFG   015502      2520    3616#
MXWRIT  015353      2535    3597#
MYES    015254      2386    3585#
NEXT    007502      2307    2311    2315#
NOTA    007460      2305    2308#
NOTB    007472      2309    2312#
NOTLGL  007514      2302    2313    2319#   2322
OFFLIN= 001000       785#   2112    2638
OK      002272      1194    1196#
PIRQ  = 177772       671#
PIRQVE= 000240       765#
PRO   = 000000       688#
PRI   = 000040       689#
PR2   = 000100       690#
```

```
PR3     = 000140   691#
PR4     = 000200   692#
PR5     = 000240   693#
PR6     = 000300   694#
PR7     = 000340   695#
PS      = 177776   668#   669   1202#  1250#  1254#  2523#  2546#  2551#  2570#  2595#  2601#  2629#  2652#
                  2673#  2688#  2709#  2737#  2863#
PSW     = 177776   669#
PWRST     002300  1198#  3535
PWRVEC= 000024   760#  1049#  1050#  3502#  3503#  3512#  3518#  3530#  3531#
RBUF1     017120  3757   3763   3824#
RBUF2     017322  3758   3825#
RBUF3     017524  3759   3826#
RBUF4     017726  3760   3827#
RBUF5     020130  3761   3828#
RBUF6     020332  3762   3829#
RCRC0     001210   878#  2587#  3554
RCRC1     001212   879#  2625#  3554
RCRC2     001214   880#  2628#  2633#  3554
RDCHR   = 104407  2142   2378   3304   3484#
RDLIN   = 104410  2299   2430   2443   3345   3485#
RDOCT   = 104411  2331   2345   2354   3486#
RDSUB     012522   820   2968#  2972   2980
READ    = 104414  1294   1307   1320   1333   1338   1351   1356   1371   1384   1390   1404   1405   1420
                  1435   1451   1466   1481   1485   1492   1499   1506   1512   1516   1529   1535   1542
                  1549   1557   1561   1575   1589   1602   1604   1622   1629   1645   1662   1681   1701
                  1720   1740   1759   1775   1792   1797   1817   1836   1854   1872   1897   1920   1925
                  1947   1953   1958   1963   1968   1996   2001   2032   2038   2045   2052   2977   3035
                  3489#
READY   = 000040   789#  2108   2213   2765   2792   2877   2925   2953   2975   3007   3033   3063
RESVEC= 000010   755#
REWIND= 104421  1240   1266   1291   1293   1304   1317   1330   1332   1348   1350   1366   1368   1381
                  1400   1403   1415   1430   1433   1445   1461   1476   1526   1571   1585   1599   1601
                  1619   1621   1639   1643   1655   1659   1672   1677   1691   1697   1711   1715   1730
                  1735   1750   1754   1769   1735   1789   1807   1814   1827   1833   1846   1849   1864
                  1867   1882   1889   1907   1917   1935   1943   1978   1988   2011   2024   3494#
RGPBLK    012702   822   3026#  3030   3038   3044
RLNKX     017006  2977   2982   3035   3040   3102#  3763#
RLNK1     016756  1294   1296   1307   1309   1320   1322   1333   1335   1338   1340   1351   1353   1356
                  1358   1371   1373   1384   1386   1404   1420   1422   1481   1483   1512   1514   1529
                  1531   1557   1559   1575   1577   1589   1591   1602   1604   1622   1624   1662   1664
                  1720   1722   1740   1742   1759   1761   1775   1777   1792   1794   1953   1955   1958
                  1960   2052   2054   2586   3757#
RLNK2     016762  1390   1392   1405   1407   1435   1437   1451   1453   1466   1468   1485   1487   1506
                  1508   1516   1518   1535   1537   1609   1611   1629   1631   1645   1647   1681   1683
                  1797   1799   1897   1899   3758#
RLNK3     016766  1492   1494   1561   1563   1817   1819   1836   1838   1854   1856   1872   1874   1963
                  1965   1996   1998   2045   2047   3759#
RLNK4     016772  1499   1501   1542   1544   1701   1703   1947   1949   1968   1970   3760#
RLNK5     016776  1549   1551   1920   1922   1925   1927   2001   2003   2038   2040   3761#
RLNK6     017002  2032   2034   3762#
RWDA      001260   895    896#  2477
RWDB      001262   897#  2474
RWDFLG    001256   895#  1237#  1264#  2474#  2477#  2729   2733#  2881#
SAVPC     001204   876#  2259#  2492#  2519#  2534#  2581#  2648#  2669#  2684#  2705#  2723#  2895#  3547
                  3550   3554   3558   3561
```

```
SAVR1   010244          2435*   2458    2460*
SELDRV= 104422          1235    1263    1290    1303    1316    1329    1347    1365    1380    1399    1414    1429    1444
                        1460    1475    1525    1570    1584    1598    1618    1638    1654    1671    1690    1710    1729
                        1749    1768    1784    1806    1825    1845    1863    1881    1906    1934    1977    2010    2923
                        2951    2973    3005    3031    3061    3080    3495*
SERV    012340          2565    2613    2873    2888*
SETBUF  013106          2945    2999    3090*
SETLOO= 104424          2145    3497*
SFBG    = 104420        1363    1434    1450    1465    1679    1699    1700    1717    1737    1756    1834    1850    1868
                        1871    1962    3042    3493*
SFFG    = 104417        1644    1660    1678    1698    1716    1736    1755    1790    1796    1815    1816    1835    1890
                        1891    1892    1896    1918    1919    1944    1945    1967    1989    1990    1991    1995    2000
                        2025    2026    2027    2028    2029    2030    3065    3492*
SFFGSB  013000           823    3056*   3060    3069
STACK   = 001100         659*   1041    2918    2946    2968    2978    3000    3026    3036    3056
STALL   001264           898*   1258*   1259*   2802    2803
START   002312          1197    1200*   2089
STKLMT= 177774           670*
SWR     001140           854*   1039    1061*   1063    1069*   1083    1215    2116    2118    2149    2167    2174    2225
                        2232    2237    2241    2466    2483    2501    2641    2762    2798    2866    3092    3096    3098
                        3105    3109    3116    3201    3238*   3510    3523*
SWREG   000176           812*   1069    1083    3092    3105    3116    3201    3214
SW0     = 000001         723*
SW00    = 000001         713*    723
SW01    = 000002         712*    722
SW02    = 000004         711*    721
SW03    = 000010         710*    720    2501
SW04    = 000020         709*    719    2641
SW05    = 000040         708*    718    2762    2866
SW06    = 000100         707*    717    2798
SW07    = 000200         706*    716    1215    2466
SW08    = 000400         705*    715
SW09    = 001000         704*    714    2483
SW1     = 000002         722*
SW10    = 002000         703*
SW11    = 004000         702*
SW12    = 010000         701*
SW13    = 020000         700*
SW14    = 040000         699*
SW15    = 100000         698*
SW2     = 000004         721*
SW3     = 000010         720*
SW4     = 000020         719*
SW5     = 000040         718*
SW6     = 000100         717*
SW7     = 000200         716*
SW8     = 000400         715*
SW9     = 001000         714*
TACSH   001222           884*   2339*   2340*
TACSL   001220           883*   1129    1203    2103    2336*   2339    2366    2919    2947    2969    3001    3027    3057
                        3735
TADBH   001226           886*   2342*   2343*
TADBL   001224           885*   1204    2338*   2342    2369    2920    2948    2970    3002    3028    3058
TAPRIO  001234           884*   1201    2364*   2375
TAVEC   001230           887*   1026    1200*   1201*   1248*   1260*   2350*   2352*   2372    2565*   2613*   2869*   2873*
                        2889*
```

```
TAI VE= 000014   756#
TIMERR= 002000   784#  2638
TKVEC = 000060   763#
TPVEC = 000064   764#
TRAPVE= 000034   762#  1047*  1048*
TRTVEC= 000014   757#
TR.REQ= 000200   787#  2765   2792
TST1    002762  1230   1269   1288#
TST10   003342  1379   1397#
TST11   003406  1398   1412#
TST12   003450  1413   1427#
TST13   003512  1428   1442#
TST14   003556  1443   1459#
TST15   003620  1459   1473#
TST16   004012  1474   1523#
TST17   004170  1524   1568#
TST2    003016  1289   1301#
TST20   004226  1569   1582#
TST21   004264  1583   1596#
TST22   004342  1597   1616#
TST23   004420  1617   1636#
TST24   004464  1637   1652#
TST25   004532  1653   1669#
TST26   004606  1670   1688#
TST27   004666  1689   1708#
TST3    003052  1302   1314#
TST30   004740  1709   1727#
TST31   005016  1728   1747#
TST32   005070  1748   1766#
TST33   005134  1767   1782#
TST34   005216  1783   1804#
TST35   005274  1805   1824#
TST36   005350  1825   1843#
TST37   005422  1844   1861#
TST4    003106  1315   1327#
TST40   005474  1862   1879#
TST41   005564  1880   1904#
TST42   005672  1905   1932#
TST43   006042  1933   1975#
TST44   006162  1976   2008#
TST5    003156  1328   1345#
TST6    003226  1346   1363#
TST7    003266  1364   1378#
TYPE  = 104401  1079   1198   1213   1214   1225   1226   1275   1279   2078   2079   2227   2235   2255
                2256   2258   2268   2272   2277   2285   2297   2319   2330   2344   2353   2365   2368
                2371   2374   2377   2384   2386   2429   2442   3155   3212   3213   3216   3229   3240
                3259   3308   3311   3315   3428   3476#  3532
TYPERR  007276  2234   2255#
TYPOC = 104402  2282   2367   2370   2373   2376   3215   3477#
TYPON = 104404  3479#
TYPOS = 104403  3478#
T.BLKC  010362  2490#  3496
T.ESBG  011366  2669#  3491
T.BSFG  011304  2648#  3490
T.READ  010774  2581#  3489
T.RWND  011600  2723#  3494
```

```
T.SELD  010246          2465#   3495
T.SETL  010106          2428#   2439    3497
T.SFBG  011520          2705#   3493
T.SFFG  011436          2684#   3492
T.WFG   010504          2519#   3487
T.WRIT  010554          2534#   3488
UNIT  = 000400           786#   2107    2473    2476
WAITFL  011716          2524    2547    2571    2597    2630    2653    2674    2689    2710    3738    2759#
WAITKE  001254           894#   2550*   2600*   2764*   2773    2870*   2888*
WBUFX   017112          3110    3770    3816#
WBUF1   017046          3764    3774#
WBUF2   017054          3765    3781#
WBUF3   017062          3766    3788#
WBUF4   017070          3767    3795#
WBUF5   017076          3768    3802#
WBUF6   017104          3769    3809#
WCRCX   017116          3821#
WCRC1   017052          3779#
WCRC2   017060          3786#
WCRC3   017066          3793#
WCRC4   017074          3800#
WCRC5   017102          3907#
WCRC6   017110          3814#
WFG   = 104412          1241    1267    1573    1587    1606    1626    1641    1657    1674    1693    1713    1732    1752
                        1771    1787    1809    1811    1813    1830    1832    1851    1884    1886    1888    1910    1913
                        1916    1939    1942    1981    1984    1987    2013    2015    2017    2019    2021    2023    2927
                        3009    3487#
WFGSUB  012364           818    2918#   2922    2931
WGBLOP  012614          3000#   3004    3016
WGPBLK  012610           321    2999#
WLNKX   017042          2955    2983    3012    3041    3103*   3770#
WLNK1   017012          1292    1297    1305    1310    1318    1323    1331    1336    1341    1349    1354    1359    1367
                        1374    1382    1387    1401    1416    1423    1431    1446    1462    1477    1484    1515    1527
                        1532    1560    1572     78     1536    1592    1600    1605    1620    1625    1640    1656    1665
                        1673    1692    1712     723    1731    1743    1751    1762    1770    1778    1786    1795    1808
                        1828    1847     186     1883    1908    1936    1956    1961    1979    2012    2055    2586    3764#
WLNK2   017016          1388    1393    1402    1408    1417    1432    1438    1447    1454    1463    1469    1470    1488
                        1504    1509    1519    1533    1538    1607    1612    1627    1632    1642    1648    1658    1675
                        1684    1694    1714    1733    1753    1772    1788    1800    1810    1829    1848    1866    1885
                        1900    1909    1937    1980    2014    3765#
WLNK3   017022          1490    1495    1554    1564    1676    1695    1734    1812    1820    1831    1839    1852    1857
                        1869    1875    1887    1911    1938    1966    1982    1999    2016    2049    3766#
WLNK4   017026          1497    1502    1540    1545    1696    1704    1912    1940    1950    1971    1983    2018    3767#
WLNK5   017032          1547    1552    1614    1923    1928    1941    1985    2004    2020    2041    3768#
WLNK6   017036          1915    1986    2022    2035    3769#
WLOOP   012444          2946#   2950    2959
WRITE = 104413          1292    1305    1318    1331    1349    1367    1382    1388    1401    1402    1416    1417    1431
                        1432    1446    1447    1462    1463    1477    1478    1490    1497    1504    1527    1533    1540
                        1547    1554    1572    1586    1600    1607    1620    1627    1640    1642    1656    1658    1673
                        1675    1676    1692    1694    1695    1696    1712    1714    1731    1733    1734    1751    1753
                        1770    1772    1786    1788    1808    1810    1812    1828    1829    1831    1847    1848    1852
                        1865    1866    1869    1883    1885    1887    1908    1909    1911    1912    1914    1915    1936
                        1937    1938    1940    1941    1979    1980    1982    1983    1985    1986    2012    2014    2016
                        2018    2020    2022    2955    3012    3488#
WRTLOC= 010000           782#   2114    2638
WRTSUB  012440           819    2945#
```

```
XBSBG = 000010            773#  2672
XBSFG = 000006            772#  2651
XREAD = 000004            771#  2594  2627
XRWND = 000016            776#  1239  1265  2706  2924  2952  2974  3006  3032  3062
XSFBG = 000014            775#  2708
XSFFG = 000012            774#  2687
XWFG  = 000000            769#  1249  2522
XWRITE= 000002            770#  2545  2569
$AUTOB  001134            851#  1087*  3209  3330
$BDADR  001122            846#
$BDDAT  001126            848#  2513*  3550
$BELL   001174            871#  2227  2253
$CHARC  013444           3157*  3167*  3174  3183*  3188#
$CKSWR  013450           3201#  3483
$CMTAG  001100            834#  1036  1037  1045  1051  1052  1053
$CM1  = 000002            866#   867#   868#
$CM2  = 000004            866#   867#   868#
$CM3  = 000002            864#   866
$CM4  = 000001            868#   869#
$CNTLG  014175           3212   3325#
$CNTLU  014170           3229   3324#
$CRLF   001201            873#  2235  2253  2255  2258  2277  3156  3191  3240  3324
$DOAGN  006442           2074   2081  2087#
$ENDAD  006432           1077   1142  1192  1282  2083#  2248
$ENDCT  006406           1051   2076#
$ENDMG  006451           2078   2091#
$ENULL  006446           2079   2090#
$EOP    006352           2009   2066#
$EOPCT  006400           1051*  2073#  2077
$ERFLG  001103            837#  2130  2163  2165  2171*  2191  2222*  2253  2485*
$ERMAX  001115            843#  1054*  2165  2187*  2191
$ERROR  007046           1045   2205#
$ERRPC  001116            844#  2229*  2230*  2231  2253  3547  3550  3554  3558  3561  3735  3737
$ERRTB  001270            913#  2266
$ERTTL  001112            841#  2228*  2253
$ESCAP  001172            870#  1053*  1236*  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*  1862*  1880*  1905*  1933*  1976*  2009*
                         2186*  2220*  2244  2246  2253  2617  2618*  2622*  3079*
$FILLC  001156            862#  3160  3191
$FILLS  001155            861#  3191
$GDADR  001120            845#
$GDDAT  001124            847#  2515*  3550
$GET42  006422           2080#
$GTSWR  013520           3213#  3481
$HD   = 000000             592
$HIOCT  014324           3362*  3367#
$ICNT   001104            838#  2178*  2179  2181*  2191
$ILLUP  015026           3502   3518  3537#
$INTAG  001135            852#  3241  3330
$ITEMB  001114            842#  2231*  2253  2261
$LF     001202            874#  2253  3191  3315  3324
$LPADR  001106            839#  1055*  1228*  2169*  2184*  2189  2191  2451*  2453*
$LPERR  001110            840#  1056*  2169  2185*  2191  2243  2454*  2922*  2950*  2972*  3004*  3030*  3060*
                         3078*
$MAIL = ****** L         1073   1083  2184  2237  3144
```

```
$MNEW   014213          3216    3328#
$MSWR   014202          3213    3326#
$MXCNT  001216           882#   1207#  1219#  2182   2191
$NULL   001154           860#   3162   3191
$NNTST= 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   2180   2168#
$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   231'   2384   3191   3259   3308   3324
$RDCHR  013732          3272#   3484
$RDDEC= ****** U        3487
$RDLIN  014052          3300#   3485
$RDOCT  014224          3340#   3486
$RDSZ = 000010          3293#
$REGAD  001160           864#
$REG0   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   104'   1 9    1051   1052   1053   1055   1077   1080   2068
                        2139    2206   2240   2248   3196   3_30
$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   1862   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
$SWRMK= 000000          2135
$TIMES  001170           869#   1052#  2069#  2172#  2179   2182#  2191
$TKB    001146           857#   3194   3205   3222   3276   3282
$TKS    001144           856#   2140   3194   3203   3219   3243#  3274   3280
$TMPO   001166           868#   2615#  3558
$TN   = 000045           582#    592   1285   1289#  1298   1302#  1311   1315#  1324   1328#  1342   1346#  1360
                        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   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#
$TPB    001152           859#   3180*  3191
$TPFLG  001157           863#   3138   3191
$TPS    001150           858#   3178   3191
```

```
$TRAP   014554            1047    3453#
$TRAP2  014576            3464#   3475
$TRP  = 000025            3468#   3477#   3478#   3479#   3480#   3481    3482#   3493    3484#   3485#   3486#   3487#   3488#
                          3489#   3490#   3491#   3492#   3493#   3494#   3495#   3496#   3497#   3498#
$TRPAD  014610            3458    3475#
$TSTNM  001102             836#   1205#   2068#   2130    2183#   2188    2191    2224    2253
$TTYIN  014160            3301    3302    3319    3323#
$TYPBN= ****** U          3480
$TYPDS= ****** U          3480
$TYPE   013230            3138#   3468    3476
$TYPEC  013400            3159    3166    3173    3178#   3179    3245
$TYPEX  013446            3184    3186    3189#
$TYPOC  014352            3398#   3477
$TYPON  014366            3397    3400#   3479
$TYPOS  014326            3393#   3478
$XTSTR  006630            2152#
$$GET4= 000000            2082#
$OFILL  014551            3394*   3398*   3408    3443#
$40CAT= ****** U          2149    2234
.     = 020534             806#    810#    833#    875    1040    1055    1056    2090    2093    2191    2253    2288#   3191
                          3194    3323#   3324    3330    3514    3538    3734#   3824#   3825#   3826#   3827#   3828#   3829#
                          3831

. ABS.  020534      000

 ERRORS DETECTED:  0

 DSKZ:CZTADD,DSKZ:CZTADD.SEQ=DSKZ:CZTADD.P11
 RUN-TIME:  20 14 1 SECONDS
 RUN-TIME RATIO: 213/36=5.8
 CORE USED:  23K  (45 PAGES)

 DOCUMENT PAGES:  97
```

80H