

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZTAA-C-D  
PRODUCT NAME: TA11 LOGIC TEST (PART 1)  
DATE REVISED: 21 JUNE 76  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: JIM LACEY

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1973,1976 BY DIGITAL EQUIPMEN CORPORATION

## CONTENTS

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

1. ABSTRACT

THIS PROGRAM CONTAINS A SERIES OF BASIC LOGIC TESTS THAT CHECK THE TA11 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

NONE

3. LOADING PROCEDURE

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

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE 5.1.

4.2 STARTING ADDRESSES

200	NORMAL STARTING ADDRESS
204	SELECT DRIVE(S) BEFORE STARTING TEST
210	SELECT DRIVE(S) AND ADDRESSES BEFORE STARTING TEST
214	SETUP FOR MANUAL LOOPING
220	WRITE FILE GAP FROM BOT TO EOT
224	WRITE CONTINUOUS BLOCKS OF DATA
230	READ CONTINUOUS BLOCKS OF DATA
234	WRITE FILE GAP AND A BLOCK OF DATA
240	READ BLOCK OF DATA AND INTO A FILE GAP
244	SPACE FWD FILE GAP FROM BOT TO EOT
250	BACK SPACE FILE GAPS
500	LOAD SWITCH REGISTER INTO THE TACS
600	WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
700	READ FROM BOT TO EOT

#### 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 PROGRAM WILL TYPE "SWR=XXXXXX NEW=" AFTER TYPING THE NAME OF THE PROGRAM.

##### 4.3.1 DRIVE SELECTION

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

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

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

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

##### 4.3.1.1 DRIVE SELECTION EXAMPLES

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

##### 4.3.2 ADDRESS SELECTION

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

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

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

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

#### 4.3.2.1 ADDRESS SELECTION EXAMPLES

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

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

#### 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<08>=1...LOOP ON TEST AS PER SW<07:00>  
SW<07>=1...LOCK ON CURRENT DRIVE (ONLY VALID  
FOR STARTING ADDRESSES 220 THRU 250).

## 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

### 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 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 LABEL OF THE STARTING ADDRESS OF THE SUBROUTINES.

5.2.4.1 TYPE (\$TYPE)

ROUTINE TO TYPE AN ASCIZ STRING ON THE TTY

THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.

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 WAITREADY (WAIT.ON.READY)

WAIT ON THE "TA11 READY" BIT TO SET

5.2.4.5 WAITXFER (WAIT.FOR.XFER.REQ)

WAIT ON THE "TA11 TRANSFER REQUEST" BIT TO SET

5.2.4.6 \$TYPOC

CHANGE A BINARY NUMBER TO OCTAL ASCII AND TYPE IT.

5.2.5 THE FOLLOWING SUBROUTINES ARE CALLED BY A JSR

5.2.5.1 \$TYPEC

ROUTINE TO TYPE A SINGLE ASCII CHARACTER

5.2.5.2 TYPERR

THIS ROUTINE WILL TYPE THE ERROR MESSAGES

5.2.5.3 SELDRV

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

5.2.5.4 SELADR

THIS ROUTINE WILL ASK THE OPERATOR FOR THE ADDRESSES OF  
THE "TACS", "TADB" AND VECTOR AND THE PRIORITY TO USE.

5.2.6 THE FOLLOW ROUTINES ARE USED TO MAKE ADJUSTMENTS TO  
THE TU60. BEFORE USING ANY OF THEM LOAD AND START 214.

5.2.6.1 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".

5.2.6.2 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 PROMPED  
FOR THE VALUE WITH "SWR=XXX NEW="



### 5.2.6.3 RDSUB

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

### 5.2.6.4 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 PROMPED  
FOR THE VALUE WITH "SWR=XXX NEW="

### 5.2.6.5 RGBLK

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

### 5.2.6.6 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).

### 5.2.6.7 BSFGSB

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

5.2.7 THE FOLLOWING SUBROUTINES ARE USED BY THE ADJUSTMENT  
ROUTINES

5.2.7.1 SETBUF

SETUP BLOCK SIZE AND PATTERN

5.2.7.2 WRTBLK

WRITES A BLOCK OF DATA

5.2.7.3 RDBLK

READS A BLOCK OF DATA

5.2.7.4 NXTDRV

CHANGE DRIVE

6. ERRORS

THERE ARE A NUMBER OF ERRORS THAT CAN OCCUR IN THIS  
PROGRAM. WHEN AN ERROR IS ENCOUNTERED THE CALL TO  
THE 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. AN ERROR MESSAGE
2. A DATA HEADER
3. A DATA STRING

REFER TO THE LISTING UNDER \$ERRTB FOR THE DIFFERENT  
ERRORS THAT CAN OCCUR.

7. RESTRICTIONS

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

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE FIRST PASS TAKES APPROXIMATELY 45 SECONDS ALL  
SUBSEQUENT PASSES TAKE APPROXIMATELY 100 SECONDS.

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,  
(2000 DECIMAL UNLESS OTHERWISE SPECIFIED WITHIN A TEST),  
ITERATIONS.

8.5 SPECIAL REGISTERS

REGISTER R3,R4 AND R5 ARE RESERVED FOR THE THE FOLLOWING  
PURPOSES:

R3 = DRIVE

R4 = TACS

R5 = TADB

9. PROGRAM DESCRIPTION

THIS PROGRAM IS A SEQUENCE OF SMALL INDEPENDENT TESTS THAT  
CHECK THE TA11 FOR PROPER OPERATION WITH TAPE MOTION KEPT  
AT A MINIMUM.

THE TESTS CAN BE GROUPED INTO THE FOLLOWING GENERAL GROUPS.

1. VERIFY THAT THE "CONTROL AND STATUS" (TACS) AND THE  
"DATA BUFFER" (TADB) REGISTERS RESPOND TO ADDRESSING.
2. TEST THE "TACS" FOR PROPER OPERATION
3. TEST "READY"
4. TEST "CLEAR LEADER ERROR"
5. TEST "TRANSFER REQUEST"
6. TEST THE "TADB" FOR PROPER OPERATION.

12	GENERAL INFORMATION
60	OPERATIONAL SWITCH SETTINGS
73	BASIC DEFINITIONS
183	TA11 DEFINITIONS
224	STARTING ADDRESSES
225	TRAP CATCHER
234	STARTING ADDRESS(ES)
246	TOGGLE IN ROUTINES
293	LOAD SWITCH REGISTER INTO TACS
301	WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
327	READ FROM BOT TO EOT
344	COMMON TAGS
404	ERROR POINTER TABLE
482	START OF TEST
507	INITIALIZE THE COMMON TAGS
545	TYPE PROGRAM NAME
552	GET VALUE FOR SOFTWARE SWITCH REGISTER
729	T1 TEST IF TACS EXIST
753	T2 TEST IF TADB EXIST
777	T3 TEST FOR LOW BYTE OF TACS
801	T4 TEST FOR LOW BYTE OF TADB
825	T5 TEST FOR HIGH BYTE OF TACS
849	T6 TEST FOR HIGH BYTE OF TADB
875	T7 TEST INITIAL CONDITION OF TACS BIT14 = 0
885	T10 TEST INITIAL CONDITION OF TACS BIT11 = 0
895	T11 TEST INITIAL CONDITION OF TACS BIT10 = 0
905	T12 TEST INITIAL CONDITION OF TACS BIT08 = 0
915	T13 TEST INITIAL CONDITION OF TACS BIT07 = 0
925	T14 TEST INITIAL CONDITION OF TACS BIT06 = 0
935	T15 TEST INITIAL CONDITION OF TACS BIT04 = 0
945	T16 TEST INITIAL CONDITION OF TACS BIT03 = 0
955	T17 TEST INITIAL CONDITION OF TACS BIT02 = 0
965	T20 TEST INITIAL CONDITION OF TACS BIT01 = 0
975	T21 TEST INITIAL CONDITION OF TACS BIT00 = 0
992	T22 TEST BIT08 OF TACS MAY BE "SET" AND "CLEARED"
1006	T23 TEST BIT08 OF TACS INITIALIZE TO ZERO
1021	T24 TEST BIT06 OF TACS MAY BE "SET" AND "CLEARED"
1035	T25 TEST BIT06 OF TACS INITIALIZE TO ZERO
1050	T26 TEST BIT04 OF TACS MAY BE "SET" AND "CLEARED"
1064	T27 TEST BIT04 OF TACS INITIALIZE TO ZERO
1079	T30 TEST BIT03 OF TACS MAY BE "SET" AND "CLEARED"
1093	T31 TEST BIT03 OF TACS INITIALIZE TO ZERO
1108	T32 TEST BIT02 OF TACS MAY BE "SET" AND "CLEARED"
1122	T33 TEST BIT02 OF TACS INITIALIZE TO ZERO
1137	T34 TEST BIT01 OF TACS MAY BE "SET" AND "CLEARED"
1151	T35 TEST BIT01 OF TACS INITIALIZE TO ZERO
1166	T36 COUNT PATTERN TO TACS<04:01>
1193	T37 TEST INITIAL CONDITION OF TACS BIT05 = 1
1204	T40 TEST INITIAL CONDITION OF TACS BIT09 = 0
1215	T41 TEST INITIAL CONDITION OF TACS BIT12 = 0
1227	T42 TEST INITIAL CONDITION OF TACS BIT13 = 1
1238	T43 TEST INITIAL CONDITION OF TACS BIT15 = 1
1257	T44 TEST TACS BIT15 IS READ ONLY
1278	T45 TEST TACS BIT14 IS READ ONLY
1299	T46 TEST TACS BIT13 IS READ ONLY
1320	T47 TEST TACS BIT12 IS READ ONLY

1341	T50	TEST TACS BIT11 IS READ ONLY
1362	T51	TEST TACS BIT10 IS READ ONLY
1383	T52	TEST TACS BIT09 IS READ ONLY
1404	T53	TEST TACS BIT07 IS READ ONLY
1425	T54	TEST TACS BIT05 IS READ ONLY
1449	T55	TEST HIGH BYTE SELECTION OF TACS
1461	T56	TEST LOW BYTE SELECTION OF TACS
1473	T57	TEST "OUT LOW" SIGNAL
1487	T60	TEST "OUT HIGH" SIGNAL
1501	T61	TEST "SELECT 00" ISN'T STUCK
1520	T62	TEST "ERROR" FOR "CLEAR LEADER"
1566	T63	TEST THAT "READY" SETS WHEN "REWIND" IS COMPLETED
1592	T64	TEST THAT "READY" SETS WHEN "WRITE FILE GAP" IS COMPLETED
1618	T65	TEST "READY" CLEARS WHEN STARTING A "REWIND" FUNCTION
1634	T66	TEST "READY" CLEARS WHEN STARTING A "WRITE FILE GAP" FUNCTION
1650	T67	TEST THAT "RESET" CANNOT ABORT A "REWIND"
1679	T70	TEST "RESET" WILL ABORT A "WRITE FILE GAP"
1703	T71	TEST "WFG" AND "REWIND" FOR NO ERRORS
1731	T72	ROUTINE TO DETERMINE TIME OF WAIT LOOPS
1753	T73	TEST THAT "CLEAR LEADER" GOES FALSE ON "REWIND"
1780	T74	TEST FOR "CLEAR LEADER ERROR" FOR "WRITE FILE GAP FUNCTION"
1810	T75	TEST "READY" CLEARS WHEN STARTING A "WRITE" FUNCTION
1826	T76	TEST "READY" CLEARS WHEN STARTING A "READ" FUNCTION
1842	T77	TEST "READY" CLEARS WHEN STARTING A "BACK SPACE FILE GAP" FUNCTION
1858	T100	TEST "READY" CLEARS WHEN STARTING A "BACK SPACE BLOCK GAP" FUNCTION
1874	T101	TEST "READY" CLEARS WHEN STARTING A "SPACE FORWARD FILE GAP" FUNCTION
1890	T102	TEST "READY" CLEARS WHEN STARTING A "SPACE FORWARD BLOCK GAP" FUNCTION
1906	T103	TEST FOR "CLEAR LEADER ERROR" FOR "WRITE FUNCTION"
1942	T104	TEST FOR "CLEAR LEADER ERROR" FOR "READ FUNCTION"
1972	T105	TEST FOR "CLEAR LEADER ERROR" FOR "BACK SPACE FILE GAP FUNCTION"
2002	T106	TEST FOR "CLEAR LEADER ERROR" FOR "BACK SPACE BLOCK GAP FUNCTION"
2032	T107	TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD FILE GAP FUNCTION"
2062	T110	TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD BLOCK GAP FUNCTION"
2092	T111	TEST "GO" BIT IS WRITE ONLY
2106		*****TRANSFER REQUEST*****
2113	T112	TEST TRANSFER REQUEST & ILBS FOR "WRITE"
2157	T113	TEST "TRANSFER REQ" AND "ILBS" FOR "READ"
2194		*****TADB*****
2195	T114	TEST THAT TADB (READ BUFFER) INITIALIZES TO 0'S
2205	T115	TEST TADB (WRITE BUFFER) FOR WRITE ONLY
2214	T116	FLOAT A "1" THROUGH TADB
2241	T117	FLOAT A "0" THROUGH TADB
2268	T120	COUNT PATTERN TO TADB
2295	T121	END OF TEST CODE
2305		END OF PASS ROUTINE
2341		SCOPE HANDLER ROUTINE
2405		ERROR HANDLER ROUTINE
2457		ERROR TIMEOUT ROUTINE
2493		ROUTINE TO WAIT ON THE READY BIT TO SET
2522		ROUTINE TO WAIT ON TRANSFER REQUEST
2551		ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2585		CHANGE TA11 ADDRESSES
2587		ROUTINE TO INPUT CSR, DBR, AND VECTOR ADDRESS AND PRIORITY
2651		***** MANUAL ADJUSTMENT ROUTINES *****
2661		WRITE FILE GAPS FROM "BOT" TO "EOT"
2686		WRITE CONTINUOUS BLOCKS OF DATA

2719	READ CONTINUOUS BLOCKS OF DATA
2742	WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
2780	READ A BLOCK OF DATA AND A FILE GAP
2808	SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
2835	BACK SPACE FILE GAP
2852	SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
2886	WRITE ROUTINE FOR THE MANUAL OPERATIONS
2904	READ ROUTINE FOR THE MANUAL OPERATIONS
2929	ROUTINE TO CHANGE DRIVES
2945	ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2975	TYPE ROUTINE
3045	READ AN OCTAL NUMBER FROM THE TTY
3083	TTY INPUT ROUTINE
3222	BINARY TO OCTAL (ASCII) AND TYPE
3299	TRAP DECODER
3322	TRAP TABLE
3343	POWER DOWN AND UP ROUTINES

```

1      .TITLE TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C
2      ;*COPYRIGHT (C) 1973,1976
3      ;*DIGITAL EQUIPMENT CORP.
4      ;*MAYNARD, MASS. 01754
5      ;*
6      ;*PROGRAM BY JIM LACEY
7      ;*
8      ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
9      ;*PACKAGE (MAINDEC-11-DZGAC-C1),MAR 24, 1976.
10     ;*
11     ;*****
12     ;*****
13     ;*****
14     .REMI

```

GENERAL INFORMATION ABOUT THE TA11/TU60 CASSETTE

ADDRESS MNEMONIC DESCRIPTION

```

21     777500 TACS   CONTROL AND STATUS REGISTER
22     777502 TADB   DATA BUFFER REGISTER
23     260    TAVEC  INTERRUPT VECTOR

```

TACS REGISTER DESCRIPTION

BIT	NAME	INIT STATE	READ AND/OR WRITE?
15	ERROR	?	READ ONLY
14	BLOCK CHECK ERROR	0	READ ONLY
13	CLEAR LEADER	?	READ ONLY
12	WRITE LOCK	?	READ ONLY
11	FILE GAP	0	READ ONLY
10	TIMING ERROR	0	READ ONLY
09	OFF LINE	?	READ ONLY
08	UNIT SELECT	0	READ/WRITE
07	TRANSFER REQUEST	0	READ ONLY
06	INTERRUPT ENABLE	0	READ/WRITE
05	READY	1	READ ONLY
04	IL55	0	READ/WRITE
03	FUNCTION BIT 02	0	READ/WRITE
02	FUNCTION BIT 01	0	READ/WRITE
01	FUNCTION BIT 00	0	READ/WRITE
	0=WRITE-FILE-GAP		
	1=READ		
	2=READ		
	3=BACK SPACE FILE GAP		
	4=BACK SPACE BLOCK GAP		
	5=SPACE FORWARD FILE GAP		
	6=SPACE FORWARD BLOCK GAP		
	7=REWIND		
00	GO BIT	0	WRITE ONLY!

GENERAL INFORMATION

```

57     ;*****
58     .SBTTL OPERATIONAL SWITCH SETTINGS
59     ;*
60     ;*      SWITCH      USE
61     ;*      -----      -----
62     ;*      15          HALT ON ERROR
63     ;*      14          LOOP ON TEST
64     ;*      13          INHIBIT ERROR TYPEOUTS
65     ;*      11          INHIBIT ITERATIONS
66     ;*      10          BELL ON ERROR
67     ;*      9           LOOP ON ERROR
68     ;*      8           LOOP ON TEST IN SWR<7:0>
69     ;*      7           LOCK ON CURRENT DRIVE (ONLY VALID WITH MANUAL LOOPING)
70     ;*****
71     .SBTTL BASIC DEFINITIONS
72
73     ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
74     001100 STACK= 1100
75     .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
76     .EQUIV IOT,SCOPE     ;;BASIC DEFINITION OF SCOPE CALL
77
78     ;*MISCELLANEOUS DEFINITIONS
79     000011 HT= 11        ;;CODE FOR HORIZONTAL TAB
80     000012 LF= 12        ;;CODE FOR LINE FEED
81     000015 CR= 15        ;;CODE FOR CARRIAGE RETURN
82     000200 CRLF= 200     ;;CODE FOR CARRIAGE RETURN-LINE FEED
83     177776 PS= 177776   ;;PROCESSOR STATUS WORD
84     .EQUIV PS,PSW
85     177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
86     177772 PIRQ= 177772  ;;PROGRAM INTERRUPT REQUEST REGISTER
87     177570 DSWR= 177570  ;;HARDWARE SWITCH REGISTER
88     177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
89
90     ;*GENERAL PURPOSE REGISTER DEFINITIONS
91     000000 R0= %0        ;;GENERAL REGISTER
92     000001 R1= %1        ;;GENERAL REGISTER
93     000002 R2= %2        ;;GENERAL REGISTER
94     000003 R3= %3        ;;GENERAL REGISTER
95     000004 R4= %4        ;;GENERAL REGISTER
96     000005 R5= %5        ;;GENERAL REGISTER
97     000006 R6= %6        ;;GENERAL REGISTER
98     000007 R7= %7        ;;GENERAL REGISTER
99     .EQUIV R6,SP        ;;STACK POINTER
100    .EQUIV R7,PC        ;;PROGRAM COUNTER
101
102     ;*PRIORITY LEVEL DEFINITIONS
103     000000 PR0= 0        ;;PRIORITY LEVEL 0
104     000040 PR1= 40       ;;PRIORITY LEVEL 1
105     000100 PR2= 100     ;;PRIORITY LEVEL 2
106     000140 PR3= 140     ;;PRIORITY LEVEL 3
107     000200 PR4= 200     ;;PRIORITY LEVEL 4
108     000240 PR5= 240     ;;PRIORITY LEVEL 5
109     000300 PR6= 300     ;;PRIORITY LEVEL 6
110     000340 PR7= 340     ;;PRIORITY LEVEL 7
111
112     ;*"SWITCH REGISTER" SWITCH DEFINITIONS

```

```

113      100000      SW15= 100000
114      040000      SW14= 400000
115      020000      SW13= 200000
116      010000      SW12= 100000
117      004000      SW11= 4000
118      002000      SW10= 2000
119      001000      SW09= 1000
120      000400      SW08= 400
121      000200      SW07= 200
122      000100      SW06= 100
123      000040      SW05= 40
124      000020      SW04= 20
125      000010      SW03= 10
126      000004      SW02= 4
127      000002      SW01= 2
128      000001      SW00= 1
129      ,EQUIV SW09,SW9
130      ,EQUIV SW08,SW8
131      ,EQUIV SW07,SW7
132      ,EQUIV SW06,SW6
133      ,EQUIV SW05,SW5
134      ,EQUIV SW04,SW4
135      ,EQUIV SW03,SW3
136      ,EQUIV SW02,SW2
137      ,EQUIV SW01,SW1
138      ,EQUIV SW00,SW0
139
140      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
141      100000      BIT15= 100000
142      040000      BIT14= 400000
143      020000      BIT13= 200000
144      010000      BIT12= 100000
145      004000      BIT11= 4000
146      002000      BIT10= 2000
147      001000      BIT09= 1000
148      000400      BIT08= 400
149      000200      BIT07= 200
150      000100      BIT06= 100
151      000040      BIT05= 40
152      000020      BIT04= 20
153      000010      BIT03= 10
154      000004      BIT02= 4
155      000002      BIT01= 2
156      000001      BIT00= 1
157      ,EQUIV BIT09,BIT9
158      ,EQUIV BIT08,BIT8
159      ,EQUIV BIT07,BIT7
160      ,EQUIV BIT06,BIT6
161      ,EQUIV BIT05,BIT5
162      ,EQUIV BIT04,BIT4
163      ,EQUIV BIT03,BIT3
164      ,EQUIV BIT02,BIT2
165      ,EQUIV BIT01,BIT1
166      ,EQUIV BIT00,BIT0
167
168      ;*BASIC "CPU" TRAP VECTOR ADDRESSES

```

```

169      000004      ERRVEC= 4      ;;TIME OUT AND OTHER ERRORS
170      000010      RESVEC= 10     ;;RESERVED AND ILLEGAL INSTRUCTIONS
171      000014      TRITVEC=14     ;;"T" BIT
172      000014      TRTVEC= 14     ;;TRACE TRAP
173      000014      BPTVEC= 14     ;;BREAKPOINT TRAP (BPT)
174      000020      IOTVEC= 20     ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
175      000024      PWRVEC= 24     ;;POWER FAIL
176      000030      EMTVEC= 30     ;;EMULATOR TRAP (EMT) **ERROR**
177      000034      TRAPVEC=34     ;;"TRAP" TRAP
178      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
179      000064      TPVEC= 64      ;;TTY PRINTER VECTOR
180      000240      PTRQVEC=240    ;;PROGRAM INTERRUPT REQUEST VECTOR

```



```

181                                     ;TA11 FUNCTIONS
182      000000                          WFG= 0                               ;WRITE FILE GAP FUNCTION
183      000002                          WRITE= 2                            ;WRITE FUNCTION
184      000004                          READ= 4                               ;READ FUNCTION
185      000006                          BSFG= 6                               ;BACK SPACE FILE GAP FUNCTION
186      000010                          BSBG= 10                              ;BACK SPACE BLOCK GAP FUNCTION
187      000012                          SFPG= 12                             ;SPACE FWD FILE GAP FUNCTION
188      000014                          SFBG= 14                              ;SPACE FWD BLOCK GAP FUNCTION
189      000016                          REWIND= 16                             ;REWIND FUNCTION
190                                     ;*****
191
192                                     ;TA11 BIT ASSIGNMENT
193      100000                          ERROR= BIT15
194      040000                          CRCERR= BIT14
195      020000                          LEADER= BIT13
196      010000                          WRTLOCK=BIT12
197      004000                          FGAP= BIT11
198      002000                          TIMERR= BIT10
199      001000                          OFFLINE=BIT09
200      000400                          UNIT= BIT08
201      000200                          TR_REQ= BIT07
202      000100                          INT_EN= BIT06
203      000040                          READY= BIT05
204      000020                          ILBS= BIT04
205      000010                          FUNC2= BIT03
206      000004                          FUNC1= BIT02
207      000002                          FUNC0= BIT01
208      000001                          GO= BIT00
209      000016                          FUNCTION= FUNC2+FUNC1+FUNC0
210                                     ;////////////////////////////////////
211                                     ;////////////////////////////////////
212                                     ;////////////////////////////////////
213
214                                     ;SPECIAL REGISTERS
215      000003                          DRIVE= %3                               ;R3 CONTAINS THE DRIVE UNDER TEST
216      000004                          TACS= %4                               ;R4 IS USED AS A POINTER TO THE TACS REGISTER
217      000005                          TADB= %5                               ;R5 IS USED AS A POINTER TO THE TADB REGISTER.
218
219                                     ;////////////////////////////////////
220                                     ;////////////////////////////////////
    
```

```

221                                     .SBTTL TRAP CATCHER
222
223      000000                          .=0
224                                     ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
225                                     ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
226                                     ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
227      000174                          .=174
228      000174      000000      DISPREG: .WORD 0                ;SOFTWARE DISPLAY REGISTER
229      000176      000000      SWREG: .WORD 0                 ;SOFTWARE SWITCH REGISTER
230
231      000200      000137      001356      .SBTTL STARTING ADDRESS(ES)
232      000204      000137      001410      JMP @*BEGIN1                ;JUMP TO STARTING ADDRESS OF PROGRAM
233      000210      000137      001416      JMP @*BEGIN2                ;SELECT DRIVE(S) BEFORE STARTING TEST
234      000214      000137      001424      JMP @*BEGIN3                ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
235      000220      000137      012622      JMP @*BEGIN4                ;SETUP FOR MANUAL LOOPING
236      000224      000137      012706      JMP @*WFGSUB                ;WRITE FILE GAP FROM BOT TO EOT
237      000230      000137      012774      JMP @*WRTSUB                ;WRITE CONTINUOUS BLOCKS OF DATA
238      000234      000137      013054      JMP @*RDSUB                 ;READ CONTINUOUS BLOCKS OF DATA
239      000240      000137      013156      JMP @*WCPBLK                ;WRITE FILE GAP AND A BLOCK OF DATA
240      000244      000137      013252      JMP @*RGPBLK                ;READ BLOCK OF DATA AND INTO A FILE GAP
241      000250      000137      013336      JMP @*SFFGSB                ;SPACE FWD FILE GAP FROM BOT TO EOT
                                                JMP @*BSFGSB                ;BACK SPACE FILE GAPS
    
```

```

242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294

```

```

;////////////////////////////////////
;////////////////////////////////////
;THE FOLLOWING ROUTINES CAN BE TOGGLED IN.
;////////////////////////////////////
;////////////////////////////////////

.REM !

THE FOLLOWING ROUTINES (LOOP1, LOOP2, & LOOP3) CAN BE TOGGLED
IN WHEN IT IS IMPOSSIBLE TO LOAD THE DIAGNOSTICS

NOTE: BEFORE USING THESE ROUTINES INSURE THAT R3,R4,& R5
ARE SETUP PROPERLY.

** NOTE: IF USING SOFTWARE SWITCH REGISTER
LOCATION SWR (=1140) MUST CONTAIN
ADDRESS "SWREG" (=176).
***** PUT VALUE INTO 176 *****
** REGISTERS 3,4,65 MUST BE SETUP **
** VIA MOVE INSTRUCTIONS **

R3= 0 IF USING DRIVE A
400 IF USING DRIVE B

R4= TA11 STATUS REG ADDRESS (TACS 177500)

R5= TA11 DATA BUFFER ADDRESS (TADB 177502)

LOOP1 WILL LOAD THE SWITCH REGISTER INTO THE TACS.

LOOP2 WILL WRITE THE CONTENTS OF THE SWITCH REGISTER
ALL THE WAY TO END-OF-TAPE(EOT).

LOOP3 WILL READ TO EOT. DATA WILL GO TO R0.

NOTE: LOOP2 AND LOOP3 WILL REWIND WHEN EOT IS REACHED AND
THEN START OVER.

!

;*****
;LOAD SWITCH REGISTER INTO THE TACS
;*****

.=500

LOOP1: MOV @SWR,@TACS ;LOAD TACS
BR LOOP1 ;LOOP

```

```

295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335

```

```

;*****
;WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
;*****

.=600

LOOP2: RESET ;CLEAR ALL FLAGS
MOV DRIVE,@TACS ;SELECT DRIVE
MOVB #REWIND!GO,@TACS ;GO TO BOT
1S: BIT #READY,@TACS ;WAIT TILL READY COMES UP
BEQ 1S
MOV #WRITE!GO,@TACS ;START A WRITE
2S: TSTB @TACS ;CHECK FOR TRANSFER REQUEST
BPL 3S ;BR IF NOT SET
MOV @SWR,@TADB ;SEND DATA TO TA11
BR 2S ;LOOP
3S: BIT #READY,@TACS ;DID READY SET?
BNE LOOP2 ;START OVER IF YES
BR 2S ;LOOP

;*****
;READ FROM BOT TO EOT
;*****

.=700

LOOP3: RESET ;CLEAR ALL FLAGS
MOV DRIVE,@TACS ;SELECT DRIVE
MOVB #REWIND!GO,@TACS ;START A REWIND
1S: BIT #READY,@TACS ;WAIT ON REWIND TO FINISH
BEQ 1S
MOV #READ!GO,@TACS ;START A READ
2S: TSTB @TACS ;CHECK TRANSFER REQ
BPL 3S ;BR IF NOT SET
MOV @TADB,R0 ;PICKUP THE DATA
BR 2S ;LOOP
3S: BIT #READY,@TACS ;CHECK READY
BNE LOOP3 ;START OVER
BR 2S ;LOOP

```

```

336                                ,SBTTL COMMON TAGS
337
338                                ;*****
339                                ;THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
340                                ;USED IN THE PROGRAM.
341
342                                ,=1100
343 001100 001100 $CMTAG: ,WORD 0 ;START OF COMMON TAGS
344 001100 000000 $PASS: ,WORD 0 ;CONTAINS PASS COUNT
345 001102 000 ,STNM: ,BYTE 0 ;CONTAINS THE TEST NUMBER
346 001103 000 $ERFLG: ,BYTE 0 ;CONTAINS ERROR FLAG
347 001104 000000 $ICNT: ,WORD 0 ;CONTAINS SUBTEST ITERATION COUNT
348 001106 000000 $LPADR: ,WORD 0 ;CONTAINS SCOPE LOOP ADDRESS
349 001110 000000 $LPERR: ,WORD 0 ;CONTAINS SCOPE RETURN FOR ERRORS
350 001112 000000 $ERTTL: ,WORD 0 ;CONTAINS TOTAL ERRORS DETECTED
351 001114 000 $ITEMB: ,BYTE 0 ;CONTAINS ITEM CONTROL BYTE
352 001115 001 $ERMAX: ,BYTE 1 ;CONTAINS MAX, ERRORS PER TEST
353 001116 000000 $ERRPC: ,WORD 0 ;CONTAINS PC OF LAST ERROR INSTRUCTION
354 001120 000000 $GDADR: ,WORD 0 ;CONTAINS ADDRESS OF "GOOD" DATA
355 001122 000000 $BDADR: ,WORD 0 ;CONTAINS ADDRESS OF "BAD" DATA
356 001124 000000 $GDDAT: ,WORD 0 ;CONTAINS "GOOD" DATA
357 001126 000000 $BDDAT: ,WORD 0 ;CONTAINS "BAD" DATA
358 001130 000000 ,WORD 0 ;RESERVED--NOT TO BE USED
359 001132 000000 ,WORD 0
360 001134 000 $AUTOB: ,BYTE 0 ;AUTOMATIC MODE INDICATOR
361 001135 000 $INTAG: ,BYTE 0 ;INTERRUPT MODE INDICATOR
362 001136 000000 ,WORD 0
363 001140 177570 SWR: ,WORD DSWR ;ADDRESS OF SWITCH REGISTER
364 001142 177570 DISPLAY: ,WORD DDISP ;ADDRESS OF DISPLAY REGISTER
365 001144 177560 $TKS: 177560 ;TTY KBD STATUS
366 001146 177562 $TKB: 177562 ;TTY KBD BUFFER
367 001150 177564 $TPS: 177564 ;TTY PRINTER STATUS REG, ADDRESS
368 001152 177566 $TPB: 177566 ;TTY PRINTER BUFFER REG, ADDRESS
369 001154 000 $NULL: ,BYTE 0 ;CONTAINS NULL CHARACTER FOR FILLS
370 001155 002 $FILLS: ,BYTE 2 ;CONTAINS # OF FILLER CHARACTERS REQUIRED
371 001156 012 $FILLC: ,BYTE 12 ;INSERT FILL CHARS. AFTER A "LINE FEED"
372 001157 000 $TPFLG: ,BYTE 0 ;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
373 001160 000000 $REGAD: ,WORD 0 ;CONTAINS THE ADDRESS FROM
374 ;WHICH ($REG0) WAS OBTAINED
375 001162 000000 $REG0: ,WORD 0 ;CONTAINS (($REGAD)+0)
376 001164 000000 $REG1: ,WORD 0 ;CONTAINS (($REGAD)+2)
377 001166 000000 $TIMES: 0 ;MAX. NUMBER OF ITERATIONS
378 001170 000000 $ESCAPE: 0 ;ESCAPE ON ERROR ADDRESS
379 001172 177607 $BELL: ,ASCIZ <207><377><377> ;CODE FOR BELL
380 001176 077 $QUES: ,ASCIZ /?/ ;QUESTION MARK
381 001177 015 $CRLF: ,ASCIZ <15> ;CARRIAGE RETURN
382 001200 000012 $LF: ,ASCIZ <12> ;LINE FEED
383 ;*****
384 001202 000000 $AVPC: ,WORD 0 ;STORAGE FOR THE PC
385 001204 000000 $AVPS: ,WORD 0 ;STORAGE FOR THE PS
386 001206 177500 TACSL: 177500 ;LOW BYTE ADDRESS OF TACS
387 001210 177501 TACSH: 177501 ;HIGH BYTE ADDRESS OF TACS
388 001212 177502 TADBL: 177502 ;LOW BYTE ADDRESS OF TADB
389 001214 177503 TADBH: 177503 ;HIGH BYTE ADDRESS OF TADB
390 001216 000260 TAVEC: 260,262 ;TA11 VECTOR ADDRESS
391 001222 000300 TAPRIO: 300 ;TA11 BR LEVEL 6
    
```

```

392 001224 000000 000000 DRVKEY: 0,0 ;DRIVE SELECT KEY:
393 001230 001224 DRVPT: DRVKEY
394 001232 000000 ASKEY: 0
395 001234 000000 CURDRV: 0 ;CURRENT DRIVE BEING TESTED
    
```

```

396          ,SBTTL  ERROR POINTER TABLE
397
398          ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
399          ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
400          ;*LOCATION SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
401          ;*NOTE1:      IF SITEMB IS 0 THE ONLY PERTINENT DATA IS (SERRPC).
402          ;*NOTE2:      EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
403
404          ;*      EM          ;;POINTS TO THE ERROR MESSAGE
405          ;*      DH          ;;POINTS TO THE DATA HEADER
406          ;*      DT          ;;POINTS TO THE DATA
407          ;*      DF          ;;POINTS TO THE DATA FORMAT
408
409
410 001236      SERRTB:
411
412          ;NOTE: ALL NUMBERS ARE TYPED AS 6-DIGIT OCTAL NUMBERS
413
414          ;ITEM 1
415          EM1          ;STATUS PROBLEM
416          DH1          ;PC TACS
417          DT1          ;SERRPC SREG0
418          0
419
420          ;ITEM 2
421          EM2          ;READY FAILED TO SET
422          DH2          ;PC TACS WAIT ADDRESS
423          DT2          ;SERRPC SREG0 SAVPC
424          0
425
426          ;ITEM 3
427          EM3          ;TRANSFER REQUEST FAILED TO SET
428          DH2          ;PC TACS WAIT ADDRESS
429          DT2          ;SERRPC SREG0 SAVPC
430          0
431
432          ;ITEM 4
433          EM4          ;THE WRONG FLAG SET
434          DH2          ;PC TACS WAIT ADDRESS
435          DT2          ;SERRPC SREG0 SAVPC
436          0
437
438          ;ITEM 5
439          EM5          ;COUNT PATTERN FAILED
440          DH5          ;PC TACS EXPECT RCV'D
441          DT5          ;SERRPC SREG0 $GDDAT $BDDAT
442          0
443
444          ;ITEM 6
445          EM6          ;DATA PROBLEM
446          DH6          ;PC TACS
447          DT6          ;SERRPC SREG1
448          0
449
450          ;ITEM 7
451          EM6          ;DATA PROBLEM
    
```

```

452 001320 016176      DH5 ;PC TACS EXPECT RCV'D
453 001322 016310      DT5 ;SERRPC SREG0 $GDDAT $BDDAT
454 001324 000000      0
455
456          ;ITEM 10
457 001326 016105      EM10 ;ADDRESS FAILED
458 001330 016251      DH10 ;PC ADDRESS
459 001332 016330      DT10 ;SERRPC $BDADR
460 001334 000000      0
461
462 001336          ITEMS2: ;ITEMS 201-202
463
464 001336 016350      EM201 ;TA11 FAILED TO RESPOND
465 001340 016422      DH201 ;PC TACS
466 001342 016336      DT201 ;SERRPC TACS
467 001344 000000      0 ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS
468
469 001346 016377      EM202 ;NO DRIVES AVAILABLE
470 001350 016437      DH202 ;PC
471 001352 016344      DT202 ;SERRPC
472 001354 000000      0
473
    
```

```

474 ;////////////////////////////////////
475 ;////////////////////////////////////
476 ;*****
477
478 ;BEGIN1 IS FOR NORMAL START
479 ;BEGIN2 IS FOR DRIVE SELECTION
480 ;BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
481 ;BEGIN4 IS FOR MANUAL OPERATION
482
483 ;*****
484
485 001356 005005 BEGIN1: CLR R5 ;NORMAL START
486 001360 012737 041101 001224 MOV #*AB,@DRVKEY
487 001366 122737 000005 000041 CMPB #5,@#41 ;CASSETTE DDP?
488 001374 001015 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
489 001376 022737 000260 001216 CMP #260,@TAVEC ;STANDARD VECTOR?
490 001404 001011 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
491 001406 000403 BR BEGIN3 ;GET DRIVES AND ADDRESSES
492 001410 012705 000001 BEGIN2: MOV #1,R5 ;ASK FOR DRIVES FLAG
493 001414 000405 BR BGNCMN ;BEGIN COMMON CODE
494 001416 012705 000002 BEGIN3: MOV #2,R5 ;ASK FOR DRIVES AND ADDRESSES
495 001422 000402 BR BGNCMN
496 001424 012705 000003 BEGIN4: MOV #3,R5
497 001430 BGNCMN:
498
499 .SBTTL INITIALIZE THE COMMON TAGS
500 ;:CLEAR THE COMMON TAGS (SCMTAG) AREA
501 001434 005026 MOV #SCMTAG,R6 ;FIRST LOCATION TO BE CLEARED
502 001436 022706 001140 CLR (R6)+ ;CLEAR MEMORY LOCATION
503 001442 001374 CMP #SWR,R6 ;:DONE?
504 001444 012706 001100 BNE -6 ;:LOOP BACK IF NO
505 MOV #STACK,SP ;:SETUP THE STACK POINTER
506 ;:INITIALIZE A FEW VECTORS
507 001450 012737 011232 000020 MOV #SCOPE,@#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
508 001456 012737 000340 000022 MOV #340,@#IOTVEC+2 ;:LEVEL 7
509 001464 012737 011504 000030 MOV #ERROR,@#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
510 001472 012737 000340 000032 MOV #340,@#EMTVEC+2 ;:LEVEL 7
511 001500 012737 015322 000034 MOV #TRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
512 001506 012737 000340 000036 MOV #340,@#TRAPVEC+2 ;:LEVEL 7
513 001514 012737 015406 000024 MOV #SPWRDN,@#PWRVEC ;:POWER FAILURE VECTOR
514 001522 012737 000340 000026 MOV #340,@#PWRVEC+2 ;:LEVEL 7
515 001530 016767 007424 007414 MOV #SENDT,SEOPCT ;:SETUP END-OF-PROGRAM COUNTER
516 001536 005067 177424 CLR STIMES ;:INITIALIZE NUMBER OF ITERATIONS
517 001542 005067 177422 CLR ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
518 001546 112767 000001 177341 MOVB #1,SERMAX ;:ALLOW ONE ERROR PER TEST
519 001554 012767 001554 177324 MOV #,,SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
520 001562 012767 001562 177320 MOV #,,SLPERR ;:SETUP THE ERROR LOOP ADDRESS
521 ;:SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
522 ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
523 001570 013746 000004 MOV #ERRVEC,-(SP) ;:SAVE ERROR VECTOR
524 001574 012737 001630 000004 MOV #64S,@ERRVEC ;:SET UP ERROR VECTOR
525 001602 012767 177570 177330 MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
526 001610 012767 177570 177324 MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
527 001616 022777 177777 177314 CMP #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
528 001624 001012 BNE 66S ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
529 001626 000403 BR 65S ;:AND THE HARDWARE SWR IS NOT = -1
;:BRANCH IF NO TIMEOUT

```

```

530 001630 012716 001636 64S: MOV #65S,(SP) ;:SET UP FOR TRAP RETURN
531 001634 000002 RTI
532 001636 012767 000176 177274 65S: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
533 001644 012767 000174 177270 MOV #DISPREG,DISPLAY
534 001652 012637 000004 66S: MOV (SP)+,@ERRVEC ;:RESTORE ERROR VECTOR
535
536 .SBTTL TYPE PROGRAM NAME
537 ;:TYPE THE NAME OF THE PROGRAM IF FIRST PASS
538 001656 005227 177777 INC #-1 ;:FIRST TIME?
539 001662 001036 BNE HERE ;:BRANCH IF NO
540 001664 022737 011200 000042 CMP #SENDAD,@#42 ;:ACT-11?
541 001672 001432 BEQ HERE ;:BRANCH IF YES
542 001674 104401 001732 TYPE ,MSGID ;:TYPE ASCIZ STRING
543 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
544 001700 005737 000042 TST #42 ;:ARE WE RUNNING UNDER XXDP/ACT?
545 001704 001006 BNE 67S ;:BRANCH IF YES
546 001706 026727 177226 000176 CMP SWR,#SWREG ;:SOFTWARE SWITCH REG SELECTED?
547 001714 001005 BNE 68S ;:BRANCH IF NO
548 001716 104405 GTSWR ;:GET SOFT-SWR SETTINGS
549 001720 000403 BR 68S
550 001722 112767 000001 177204 67S: MOVB #1,SAUTOB ;:SET AUTO-MODE INDICATOR
551 001730 68S:
552 001730 000413 BR HERE ;:GET OVER THE ASCIZ
553 ;:MSGID: .ASCIZ <CRLF>/MAINDEC-11-DZTAA-C/<CRLF>
554 001760 HERE:

```

```

555 ;*****
556 ;*****
557 ;
558 ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
559 ;
560 ; R5=3 MANUAL OPERATIONS
561 ; R5=2 ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
562 ; R5=1 ASK FOR DRIVE(S)
563 ; R5=0 DON'T ASK FOR ANYTHING
564 ;
565 ;*****
566 001760 010504 BEGINX: MOV R5,R4 ;COPY R5
567 001762 005305 DEC R5 ;ASK FOR DRIVES?
568 001764 002406 BLT CHKADR ;BR IF NO
569 001766 004737 012222 JSR PC,#ASKDRV ;GO GET DRIVES TO BE TESTED
570 001772 005305 DEC R5 ;ASK FOR ADDRESSES?
571 001774 002402 BLT CHKADR ;BR IF NO
572 001776 004737 012332 JSR PC,#ASKADR ;GO GET TA11 ADDRESSES
573 ;*****
574 ;*****
575 ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
576 ;
577 ;
578 ;I. TIMEOUT OCCURRED
579 ; A. TYPE ERROR MESSAGE
580 ; B. EXAMINE R4
581 ; 1. R4>0 GOTO BEGINX
582 ; 2. R4=0 EXAMINE (42)
583 ; A. (42)=0 GOTO BEGINX
584 ; B. (42)>0 GOTO SENDAD
585 ;
586 ;II. TIMEOUT DIDN'T OCCUR
587 ; A. CONTINUE
588 ;
589 ;*****
590 002002 012737 002020 000004 CHKADR: MOV #1S,#ERRVEC ;IN CASE OF TIMEOUTS
591 002010 005000 CLR R0 ;USE AS A SWITCH
592 002012 005777 177170 IST @TACSL ;SEE IF TA11 RESPONDS
593 002016 000402 BR 2S ;BR IF NO TIMEOUT
594 002020 005200 1S: INC R0 ;COME HERE ON TIMEOUT
595 002022 022626 CMP (SP)+,(SP)+ ;CLEANUP THE STACK
596 002024 012737 000006 000004 2S: MOV #ERRVEC+2,#ERRVEC ;RESTORE TIMEOUT VECTOR
597 002032 005700 TST R0 ;DID A TIMEOUT OCCUR?
598 002034 001412 BEQ 3S ;BR IF NO
599 002036 104201 ERROR 201 ;TA11 FAILED TO RESPOND
600 002040 012705 000002 MOV #2,P5 ;DRIVES & ADDRESSES
601 002044 005704 TST R4 ;OPERATOR INPUTS?
602 002046 001344 BNE BEGINX ;BR IF YES
603 002050 013700 000042 MOV #42,R0 ;GET MONITOR RETURN ADDRESS
604 002054 001741 BEQ BEGINX ;BR IF NO MONITOR
605 002056 000137 011200 JMP #SENDAD ;GO TO END
606 002062 3S:
    
```

```

607 ;*****
608 ;*****
609 ;
610 ;MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED
611 ;
612 ;I. DESIRED DRIVES CAN NOT BE TESTED
613 ; A. TYPE ERROR MESSAGE
614 ; B. EXAMINE R4
615 ; 1. R4>0 GOTO BEGINX
616 ; 2. R4=0 EXAMINE (42)
617 ; A. (42)=0 GOTO BEGINX
618 ; B. (42)>0 GOTO SENDAD
619 ;
620 ;II. BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
621 ; A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
622 ; B. CONTINUE IN PROGRAM
623 ;
624 ;III. DESIRED DRIVE(S) CAN BE TESTED
625 ; A. CONTINUE IN PROGRAM
626 ;
627 ;*****
628 002062 012700 001224 CHKDRV: MOV #DRVKEY,R0 ;PICKUP ADDRESS OF ASCII DRIVE KEY
629 002066 004737 013672 JSR PC,#EXAM ;GO EXAMINE FIRST DRIVE
630 002072 000410 BR 1S ;OK TO TEST---GO CHECK NEXT
631 002074 116010 000001 MOVB 1(R0),(R0) ;REPLACE 1ST WITH 2ND
632 002100 001412 BEQ 2S ;BR IF NO 2ND DRIVE SELECTED
633 002102 004737 013672 JSR PC,#EXAM ;GO EXAMINE DRIVE
634 002106 000407 BR 2S ;OK TO TEST
635 002110 005010 CLR (R0) ;CLEAR DRIVE CODES
636 002112 000405 BR 2S
637 002114 005200 1S: INC R0 ;POINT TO 2ND
638 002116 004737 013672 JSR PC,#EXAM ;GO EXAMINE DRIVE
639 002122 000401 BR 2S ;OK TO TEST
640 002124 105010 CLR (R0) ;CLEAR 2ND
641 002126 012700 001224 2S: MOV #DRVKEY,R0 ;RESET ADDRESS POINTERS
642 002132 010037 001230 MOV R0,#DRVPT
643 002136 121060 000001 CHPB (R0),1(R0) ;1ST = 2ND?
644 002142 001002 BNE 3S ;BR IF NO
645 002144 105060 000001 CLR (R0) ;YES---CLEAR 2ND
646 002150 005710 3S: TST (R0) ;ANY DRIVES?
647 002152 001401 BEQ 5S ;BR IF NO
648 002154 000412 BR MANUAL
649 002156 104202 5S: ERROR 202 ;NO DRIVES AVAILABLE
650 002160 012705 000002 MOV #2,R5 ;DRIVES & ADDRESS
651 002164 005704 TST R4 ;OPERATOR INPUTS?
652 002166 001274 BNE BEGINX ;BR IF YES
653 002170 013700 000042 MOV #42,R0 ;GET MONITOR RETURN ADDRESS
654 002174 001671 BEQ BEGINX ;NO MONITOR
655 002176 000137 011200 JMP #SENDAD ;GO TO END
656 002202 020427 000003 MANUAL: CMP R4,#3
657 002206 001002 BNE OK
658 002210 016704 175563 MOV -1,R4
659 002214 010437 001232 OK: MOV R4,#ASKKEY
660 002220 000405 BR START
661 002222 104401 001732 PWRST: TYPE ,MSGID ;POWER FAIL RESTART
662 002226 012737 001224 001230 MOV #DRVKEY,#DRVPT
    
```

```
663 002234 013777 001220 176754 START: MOV 0TAVEC+2,0TAVEC ;SETUP TA11 TRAP VECTOR
664 002242 005077 176752 CLR 0TAVEC+2
665 002246 012737 000340 177776 MOV 0340,0#PS ;LOCKOUT ALL I/O INT
666 002254 013704 001206 MOV 0TACSL,TACS ;SETUP TACS
667 002260 013705 001212 MOV 0TADBL,TADB ;SETUP TADB
668 002264 005737 001100 TST 0#PASS ;IF FIRST PASS SETUP FOR EXTRA LONG WAIT LOOPS
669 002270 001024 BNE 18 ;OTHERWISE USE OLD VALUES
670 002272 012737 077777 012116 MOV 0CBIT15,0#MAXCNT
671 002300 005737 000042 TST 0#42 ;UNDER MONITOR CONTROL?
672 002304 001016 BNE 18 ;SKIP TIMEOUT IF YES
673 002306 104401 002314 TYPE ,656 ;;TYPE ASCIZ STRING
674 002312 000413 BR 646 ;;GET OVER THE ASCIZ
675 ;;656: .ASCIZ <15><12>/DO A MANUAL REWIND/
676 646:
677 002342 005037 001102 18: CLR 0#6TSTNM ;ZERO THE TEST NUMBER
678 002346 005003 CLR 0TAVEC ;SET DRIVE TO "A"
679 002350 013701 001230 MOV 0#DRVPT,R1 ;GET DRIVE POINTER
680 002354 121127 000101 CMPB (R1),#*A ;IS IT DRIVE "A"?
681 002360 001402 BEQ 0TDRV ;BR IF YES
682 002362 012703 000400 MOV 0UNIT,DRIVE ;SET DRIVE TO "B"
683 002366
684 002366 104401 002374 TDRV: TYPF ,656 ;;TYPE ASCIZ STRING
685 002372 000411 BR 646 ;;GET OVER THE ASCIZ
686 ;;656: .ASCIZ <15><12>*TESTING DRIVE *
687 646:
688 002416 112167 176612 MOVB (R1)+,CURDRV ;SETUP TO TYPE CURRENT DRIVE
689 002422 104401 001234 TYPE ,CURDRV
690 002426 104401 001177 TYPE ,SCRLF ;TYPE A CR & LF
691 002432 105711 TSTB (R1) ;LAST DRIVE BEEN SELECTED
692 002434 001002 BNE 18 ;BR IF NO
693 002436 012701 001224 MOV 0#DRVKEY,R1 ;RESET DRIVE POINTER
694 002442 010137 001230 18: R1,0#DRVPT ;SAVE DRIVE POINTER FOR NEXT TIME
695 002446 005737 001232 TST 0#ASKKEY ;GO START TESTING IF NO MANUAL
696 002452 002007 BEQ 25 ; OPERATIONS REQUESTED
697 002454 005000 CLR R0
698 002456 000000 HALT ;GIVE CONTROL TO THE OPERATOR
699 002460 022767 000176 176452 CMP 0#SWREG,SWR ;SOFTWARE SWITCH REG.?
700 002466 001001 BNE 205 ;NO - LET HIM GO
701 002470 104405 GTSWR ;GET NEW SWR VALUE
702 205:
703 ;THIS CODE IS FOR ACT11 & DDP
704 002472 005737 000042 2S: TST 0#42 ;IS THERE A MONITOR?
705 002476 001406 BEQ 0TST1 ;GO START TESTING IF NO
706 002500 010314 MOV 0TACSL,TACS ;IF YES SELECT DRIVE
707 002502 112714 000017 MOV 0#REWIND,0TACS ;SEND TAPE TO BOT
708 002506 032714 000040 3S: BIT 0#READY,0TACS ;WAIT ON READY
709 002512 001775 BEQ 3S ;FALL THRU IF READY=1
```

```
710 ;////////////////////////////////////
711 ;////////////////////////////////////
712 ;THE FOLLOWING TESTS WILL CHECK THE TA11
713 ;STATUS (TACS) AND DATA BUFFER (TADB) REGISTERS
714 ;TO INSURE THAT THEY RESPOND TO ADDRESSING.
715 ;////////////////////////////////////
716
717 ;*****
718 ;TA11 REGISTER ADDRESS TEST
719 ;INSURE THAT TACS RESPONDS TO ADDRESSING
720 ;*****
721 ;*TEST 1 TEST IF TACS EXIST
722 ;*****
723 002514 000004 TST1: SCOPE
724 002516 012767 000001 176442 MOV 0#1,STIMES ;DO 1 ITERATION
725 002524 012706 001100 MOV 0#STACK,SP ;SETUP THE STACK POINTER
726 002530 012737 002542 000004 MOV 0#18,0#ERRVEC ;SETUP FOR BUS TIMEOUT
727 002536 005014 CLR 0TACS ;TACS ARE YOU THERE?
728 002540 000420 BR 25 ;YES -- NO TRAP
729 002542 012737 000006 000004 1S: MOV 0#ERRVEC+2,0#ERRVEC ;RESTORE TRAP CATCHER
730 002550 022626 CMP (SP)+,(SP)+ ;CLEAN UP THE STACK
731 002552 013746 000030 MOV 0#EMTVEC,-(SP) ;SETUP FOR THIS ERROR CALL
732 002556 012737 011516 000030 MOV 0#ERRORI,0#EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB
733 002564 010437 001122 MOV TACS,0#SBADDR ;GET THE FAILING ADDRESS
734 002570 104010 ERROR 10 ;ERROR -- TIME OUT OCCURRED
735 ;WHEN ADDRESSING THE TACS REGISTER
736 002572 012637 000030 MOV (SP)+,0#EMTVEC ;RESTORE THE ERROR CALLS VECTOR
737 002576 000137 011124 JMP 0#SEOP ;GO TO END OF PROGRAM
738 002602 012737 000006 000004 2S: MOV 0#ERRVEC+2,0#ERRVEC ;RESTORE TRAP CATCHER
739
740 ;*****
741 ;TA11 REGISTER ADDRESS TEST
742 ;INSURE THAT TADB RESPONDS TO ADDRESSING
743 ;*****
744 ;*TEST 2 TEST IF TADB EXIST
745 ;*****
746 002610 000004 TST2: SCOPE
747 002612 012767 000001 176346 MOV 0#1,STIMES ;DO 1 ITERATION
748 002620 012706 001100 MOV 0#STACK,SP ;SETUP THE STACK POINTER
749 002624 012737 002636 000004 MOV 0#18,0#ERRVEC ;SETUP FOR BUS TIMEOUT
750 002632 005015 CLR 0TADB ;TADB ARE YOU THERE?
751 002634 000420 BR 25 ;YES -- NO TRAP
752 002636 012737 000006 000004 1S: MOV 0#ERRVEC+2,0#ERRVEC ;RESTORE TRAP CATCHER
753 002644 022626 CMP (SP)+,(SP)+ ;CLEAN UP THE STACK
754 002646 013746 000030 MOV 0#EMTVEC,-(SP) ;SETUP FOR THIS ERROR CALL
755 002652 012737 011516 000030 MOV 0#ERRORI,0#EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB
756 002660 010537 001122 MOV TADB,0#SBADDR ;GET THE FAILING ADDRESS
757 002664 104010 ERROR 10 ;ERROR -- TIME OUT OCCURRED
758 ;WHEN ADDRESSING THE TADB REGISTER
759 002666 012637 000030 MOV (SP)+,0#EMTVEC ;RESTORE THE ERROR CALLS VECTOR
760 002672 000137 011124 JMP 0#SEOP ;GO TO END OF PROGRAM
761 002676 012737 000006 000004 2S: MOV 0#ERRVEC+2,0#ERRVEC ;RESTORE TRAP CATCHER
```

```
762  
763 ;*****  
764 ;TA11 REGISTER ADDRESS TEST  
765 ;INSURE THAT TACS RESPONDS TO ADDRESSING  
766 ;*****  
767 ;*TEST 3 TEST FOR LOW BYTE OF TACS  
768 ;*****  
769 002704 000004 TST3: SCOPE  
770 002706 012767 000001 176252 MOV #1,$TIMES ;DO 1 ITERATION  
771 002714 012706 001100 MOV $STACK,SP ;SETUP THE STACK POINTER  
772 002720 012737 002732 000004 MOV #1,$ERRVEC ;SETUP FOR BUS TIMEOUT  
773 002726 105014 CLRAB @TACS ;TACS ARE YOU THERE?  
774 002730 000420 BR 2$ ;YES -- NO TRAP  
775 002732 012737 000006 000004 1s: MOV $ERRVEC+2,@$ERRVEC ;RESTORE TRAP CATCHER  
776 002740 022626 CMP (SP)+,(SP)+ ;CLEAN UP THE STACK  
777 002742 013746 000030 MOV @EMTVEC,-(SP) ;SETUP FOR THIS ERROR CALL  
778 002746 012737 011516 000030 MOV $ERROR1,@EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB  
779 002754 010437 001122 MOV TACS,@$SBDADR ;GET THE FAILING ADDRESS  
780 002760 104010 ERROR 10 ;ERROR -- TIME OUT OCCURRED  
781 ;WHEN ADDRESSING THE TACS REGISTER  
782 002762 012637 000030 MOV (SP)+,@EMTVEC ;RESTORE THE ERROR CALLS VECTOR  
783 002766 000137 011124 JMP @SEOP ;GO TO END OF PROGRAM  
784 002772 012737 000006 000004 2s: MOV $ERRVEC+2,@$ERRVEC ;RESTORE TRAP CATCHER  
785  
786 ;*****  
787 ;TA11 REGISTER ADDRESS TEST  
788 ;INSURE THAT TADB RESPONDS TO ADDRESSING  
789 ;*****  
790 ;*TEST 4 TEST FOR LOW BYTE OF TADB  
791 ;*****  
792 003000 000004 TST4: SCOPE  
793 003002 012767 000001 176156 MOV #1,$TIMES ;DO 1 ITERATION  
794 003010 012706 001100 MOV $STACK,SP ;SETUP THE STACK POINTER  
795 003014 012737 003026 000004 MOV #1,$ERRVEC ;SETUP FOR BUS TIMEOUT  
796 003022 105015 CLRAB @TADB ;TADB ARE YOU THERE?  
797 003024 000420 BR 2$ ;YES -- NO TRAP  
798 003026 012737 000006 000004 1s: MOV $ERRVEC+2,@$ERRVEC ;RESTORE TRAP CATCHER  
799 003033 022626 CMP (SP)+,(SP)+ ;CLEAN UP THE STACK  
800 003036 013746 000030 MOV @EMTVEC,-(SP) ;SETUP FOR THIS ERROR CALL  
801 003042 012737 011516 000030 MOV $ERROR1,@EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB  
802 003050 105037 001122 MOV TADB,@$SBDADR ;GET THE FAILING ADDRESS  
803 003054 104010 ERROR 10 ;ERROR -- TIME OUT OCCURRED  
804 ;WHEN ADDRESSING THE TADB REGISTER  
805 003056 012637 000030 MOV (SP)+,@EMTVEC ;RESTORE THE ERROR CALLS VECTOR  
806 003062 000137 011124 JMP @SEOP ;GO TO END OF PROGRAM  
807 003066 012737 000006 000004 2s: MOV $ERRVEC+2,@$ERRVEC ;RESTORE TRAP CATCHER  
808  
809 ;*****  
810 ;TA11 REGISTER ADDRESS TEST  
811 ;INSURE THAT TACSH RESPONDS TO ADDRESSING  
812 ;*****  
813 ;*TEST 5 TEST FOR HIGH BYTE OF TACS  
814 ;*****  
815 003074 000004 TST5: SCOPE  
816 003076 012767 000001 176062 MOV #1,$TIMES ;DO 1 ITERATION  
817 003104 012706 001100 MOV $STACK,SP ;SETUP THE STACK POINTER
```

```
818 003110 012737 003124 000004 MOV #1,$ERRVEC ;SETUP FOR BUS TIMEOUT  
819 003116 105077 176066 CLRAB @TACSH ;TACSH ARE YOU THERE?  
820 003122 000421 BR 2$ ;YES -- NO TRAP  
821 003124 012737 000006 000004 1s: MOV $ERRVEC+2,@$ERRVEC ;RESTORE TRAP CATCHER  
822 003132 022626 CMP (SP)+,(SP)+ ;CLEAN UP THE STACK  
823 003134 013746 000030 MOV @EMTVEC,-(SP) ;SETUP FOR THIS ERROR CALL  
824 003140 012737 011516 000030 MOV $ERROR1,@EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB  
825 003146 016737 176036 001122 MOV TACSH,@$SBDADR ;GET THE FAILING ADDRESS  
826 003154 104010 ERROR 10 ;ERROR -- TIME OUT OCCURRED  
827 ;WHEN ADDRESSING THE TACSH REGISTER  
828 003156 012637 000030 MOV (SP)+,@EMTVEC ;RESTORE THE ERROR CALLS VECTOR  
829 003162 000137 011124 JMP @SEOP ;GO TO END OF PROGRAM  
830 003166 012737 000006 000004 2s: MOV $ERRVEC+2,@$ERRVEC ;RESTORE TRAP CATCHER  
831  
832 ;*****  
833 ;TA11 REGISTER ADDRESS TEST  
834 ;INSURE THAT TADBH RESPONDS TO ADDRESSING  
835 ;*****  
836 ;*TEST 6 TEST FOR HIGH BYTE OF TADB  
837 ;*****  
838 003174 000004 TST6: SCOPE  
839 003176 012767 000001 175762 MOV #1,$TIMES ;DO 1 ITERATION  
840 003204 012706 001100 MOV $STACK,SP ;SETUP THE STACK POINTER  
841 003210 012737 003224 000004 MOV #1,$ERRVEC ;SETUP FOR BUS TIMEOUT  
842 003216 105077 175772 CLRAB @TADBH ;TADBH ARE YOU THERE?  
843 003222 000421 BR 2$ ;YES -- NO TRAP  
844 003224 012737 000006 000004 1s: MOV $ERRVEC+2,@$ERRVEC ;RESTORE TRAP CATCHER  
845 003232 022626 CMP (SP)+,(SP)+ ;CLEAN UP THE STACK  
846 003234 013746 000030 MOV @EMTVEC,-(SP) ;SETUP FOR THIS ERROR CALL  
847 003240 012737 011516 000030 MOV $ERROR1,@EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB  
848 003246 016737 175742 001122 MOV TADBH,@$SBDADR ;GET THE FAILING ADDRESS  
849 003254 104010 ERROR 10 ;ERROR -- TIME OUT OCCURRED  
850 ;WHEN ADDRESSING THE TADBH REGISTER  
851 003256 012637 000030 MOV (SP)+,@EMTVEC ;RESTORE THE ERROR CALLS VECTOR  
852 003262 000137 011124 JMP @SEOP ;GO TO END OF PROGRAM  
853 003266 012737 000006 000004 2s: MOV $ERRVEC+2,@$ERRVEC ;RESTORE TRAP CATCHER  
854
```



```
855 ;////////////////////////////////////
856 ;////////////////////////////////////
857 ;THE FOLLOWING TESTS WILL CHECK THE INITIAL STATE OF ALL BITS
858 ;IN THE TACS THAT ARE NOT DRIVE DEPENDENT.
859 ;////////////////////////////////////
860 ;*****
861 ;*TEST 7 TEST INITIAL CONDITION OF TACS BIT14 = 0
862 ;*****
863 003274 000004 TST7: SCOPE
864 003276 012767 003306 175602 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
865 003304 000005 RESET
866 003306 032714 040000 1$: BIT #BIT14,@TACS ;IS BIT14 OF TACS = 0?
867 003312 001401 BEQ TST10 ;;BR IF YES
868 003314 104001 ERROR 1 ;ERROR -- BIT14 OF TACS DIDN'T INITIALIZE PROPER
869 ;*****
870 ;*TEST 10 TEST INITIAL CONDITION OF TACS BIT11 = 0
871 ;*****
872 003316 000004 TST10: SCOPE
873 003320 012767 003330 175560 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
874 003326 000005 RESET
875 003330 032714 040000 1$: BIT #BIT11,@TACS ;IS BIT11 OF TACS = 0?
876 003334 001401 BEQ TST11 ;;BR IF YES
877 003336 104001 ERROR 1 ;ERROR -- BIT11 OF TACS DIDN'T INITIALIZE PROPER
878 ;*****
879 ;*TEST 11 TEST INITIAL CONDITION OF TACS BIT10 = 0
880 ;*****
881 003340 000004 TST11: SCOPE
882 003342 012767 003352 175536 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
883 003350 000005 RESET
884 003352 032714 002000 1$: BIT #BIT10,@TACS ;IS BIT10 OF TACS = 0?
885 003356 001401 BEQ TST12 ;;BR IF YES
886 003360 104001 ERROR 1 ;ERROR -- BIT10 OF TACS DIDN'T INITIALIZE PROPER
887 ;*****
888 ;*TEST 12 TEST INITIAL CONDITION OF TACS BIT08 = 0
889 ;*****
890 003362 000004 TST12: SCOPE
891 003364 012767 003374 175514 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
892 003372 000005 RESET
893 003374 032714 000400 1$: BIT #BIT08,@TACS ;IS BIT08 OF TACS = 0?
894 003400 001401 BEQ TST13 ;;BR IF YES
895 003402 104001 ERROR 1 ;ERROR -- BIT08 OF TACS DIDN'T INITIALIZE PROPER
896 ;*****
897 ;*TEST 13 TEST INITIAL CONDITION OF TACS BIT07 = 0
898 ;*****
899 003404 000004 TST13: SCOPE
900 003406 012767 003416 175472 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
901 003414 000005 RESET
902 003416 032714 000200 1$: BIT #BIT07,@TACS ;IS BIT07 OF TACS = 0?
903 003422 001401 BEQ TST14 ;;BR IF YES
904 003424 104001 ERROR 1 ;ERROR -- BIT07 OF TACS DIDN'T INITIALIZE PROPER
905 ;*****
906 ;*TEST 14 TEST INITIAL CONDITION OF TACS BIT06 = 0
907 ;*****
908 003426 000004 TST14: SCOPE
909 003430 012767 003440 175450 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
910 003436 000005 RESET
```

```
911 003440 032714 000100 1$: BIT #BIT06,@TACS ;IS BIT06 OF TACS = 0?
912 003444 001401 BEQ TST15 ;;BR IF YES
913 003446 104001 ERROR 1 ;ERROR -- BIT06 OF TACS DIDN'T INITIALIZE PROPER
914 ;*****
915 ;*TEST 15 TEST INITIAL CONDITION OF TACS BIT04 = 0
916 ;*****
917 003450 000004 TST15: SCOPE
918 003452 012767 003462 175426 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
919 003460 000005 RESET
920 003462 032714 000020 1$: BIT #BIT04,@TACS ;IS BIT04 OF TACS = 0?
921 003466 001401 BEQ TST16 ;;BR IF YES
922 003470 104001 ERROR 1 ;ERROR -- BIT04 OF TACS DIDN'T INITIALIZE PROPER
923 ;*****
924 ;*TEST 16 TEST INITIAL CONDITION OF TACS BIT03 = 0
925 ;*****
926 003472 000004 TST16: SCOPE
927 003474 012767 003504 175404 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
928 003502 000005 RESET
929 003504 032714 000010 1$: BIT #BIT03,@TACS ;IS BIT03 OF TACS = 0?
930 003510 001401 BEQ TST17 ;;BR IF YES
931 003512 104001 ERROR 1 ;ERROR -- BIT03 OF TACS DIDN'T INITIALIZE PROPER
932 ;*****
933 ;*TEST 17 TEST INITIAL CONDITION OF TACS BIT02 = 0
934 ;*****
935 003514 000004 TST17: SCOPE
936 003516 012767 003526 175362 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
937 003524 000005 RESET
938 003526 032714 000004 1$: BIT #BIT02,@TACS ;IS BIT02 OF TACS = 0?
939 003532 001401 BEQ TST20 ;;BR IF YES
940 003534 104001 ERROR 1 ;ERROR -- BIT02 OF TACS DIDN'T INITIALIZE PROPER
941 ;*****
942 ;*TEST 20 TEST INITIAL CONDITION OF TACS BIT01 = 0
943 ;*****
944 003536 000004 TST20: SCOPE
945 003540 012767 003550 175340 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
946 003546 000005 RESET
947 003550 032714 000002 1$: BIT #BIT01,@TACS ;IS BIT01 OF TACS = 0?
948 003554 001401 BEQ TST21 ;;BR IF YES
949 003556 104001 ERROR 1 ;ERROR -- BIT01 OF TACS DIDN'T INITIALIZE PROPER
950 ;*****
951 ;*TEST 21 TEST INITIAL CONDITION OF TACS BIT00 = 0
952 ;*****
953 003560 000004 TST21: SCOPE
954 003562 012767 003572 175316 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
955 003570 000005 RESET
956 003572 032714 000001 1$: BIT #BIT00,@TACS ;IS BIT00 OF TACS = 0?
957 003576 001401 BEQ TST22 ;;BR IF YES
958 003600 104001 ERROR 1 ;ERROR -- BIT00 OF TACS DIDN'T INITIALIZE PROPER
959
```

```
960 ;////////////////////////////////////  
961 ;////////////////////////////////////  
962 ;THE FOLLOWING TESTS VALIDATES EACH READ/WRITE  
963 ;BIT OF THE TACS FOR SETTING, CLEARING,  
964 ;AND INITIALIZING  
965 ;////////////////////////////////////  
966 ;*****  
967 ;*TEST 22 TEST BIT08 OF TACS MAY BE "SET" AND "CLEARED"  
968 ;*****  
969 TST22: SCOPE  
970 003602 000004  
971 003604 052714 000400 BIS #BIT08,@TACS ;SET BIT08 OF TACS  
972 003610 032714 000400 BIT #BIT08,@TACS ;DID BIT08 SET?  
973 003616 001001 BNE 1$ ;BR IF YES  
974 ERROR 1 ;BIT08 FAILED TO SET  
975 003620 042714 000400 1$: BIC #BIT08,@TACS ;NOW CLEAR IT  
976 003624 032714 000400 BIT #BIT08,@TACS ;DID BIT08 CLEAR?  
977 003630 001401 BEQ TST23 ;;BR IF YES  
978 003632 104001 ERROR 1 ;ERROR -- BIT08 OF TACS FAILED TO CLEAR  
979 ;*****  
980 ;*TEST 23 TEST BIT08 OF TACS INITIALIZE TO ZERO  
981 ;*****  
982 TST23: SCOPE  
983 003634 000004  
984 003636 012767 000005 175322 MOV #5,$TIMES ;;DO 5 ITERATIONS  
985 003644 012714 000400 1$: MOV #BIT08,@TACS ;SET BIT08 OF TACS  
986 003650 032714 000400 BIT #BIT08,@TACS ;DID BIT08 SET?  
987 003654 001001 BNE 2$ ;BR IF YES  
988 ERROR 1 ;BIT08 FAILED TO SET  
989 2$: RESET ;CLEAR THE WORLD  
990 003662 032714 000400 BIT #BIT08,@TACS ;DID BIT08 CLEAR?  
991 003666 001401 BEQ TST24 ;;BR IF YES  
992 003670 104001 ERROR 1 ;ERROR -- RESET DIDN'T CLEAR BIT08  
993 ;*****  
994 ;*TEST 24 TEST BIT06 OF TACS MAY BE "SET" AND "CLEARED"  
995 ;*****  
996 TST24: SCOPE  
997 003672 000004  
998 003674 052714 000100 BIS #BIT06,@TACS ;SET BIT06 OF TACS  
999 003700 032714 000100 BIT #BIT06,@TACS ;DID BIT06 SET?  
1000 003706 001001 BNE 1$ ;BR IF YES  
1001 ERROR 1 ;BIT06 FAILED TO SET  
1002 1$: BIC #BIT06,@TACS ;NOW CLEAR IT  
1003 003710 032714 000100 BIT #BIT06,@TACS ;DID BIT06 CLEAR?  
1004 003720 001401 BEQ TST25 ;;BR IF YES  
1005 003722 104001 ERROR 1 ;ERROR -- BIT06 OF TACS FAILED TO CLEAR  
1006 ;*****  
1007 ;*TEST 25 TEST BIT06 OF TACS INITIALIZE TO ZERO  
1008 ;*****  
1009 TST25: SCOPE  
1010 003724 000004  
1011 003726 012767 000005 175232 MOV #5,$TIMES ;;DO 5 ITERATIONS  
1012 003734 012714 000100 1$: MOV #BIT06,@TACS ;SET BIT06 OF TACS  
1013 003740 032714 000100 BIT #BIT06,@TACS ;DID BIT06 SET?  
1014 003744 001001 BNE 2$ ;BR IF YES  
1015 003746 104001 ERROR 1 ;BIT06 FAILED TO SET
```

```
1016 003750 000005 2$: RESET ;CLEAR THE WORLD  
1017 003752 032714 000100 BIT #BIT06,@TACS ;DID BIT06 CLEAR?  
1018 003756 001401 BEQ TST26 ;;BR IF YES  
1019 003760 104001 ERROR 1 ;ERROR -- RESET DIDN'T CLEAR BIT06  
1020 ;*****  
1021 ;*TEST 26 TEST BIT04 OF TACS MAY BE "SET" AND "CLEARED"  
1022 ;*****  
1023 TST26: SCOPE  
1024 003762 000004  
1025 003764 052714 000020 BIS #BIT04,@TACS ;SET BIT04 OF TACS  
1026 003770 032714 000020 BIT #BIT04,@TACS ;DID BIT04 SET?  
1027 003774 001001 BNE 1$ ;BR IF YES  
1028 ERROR 1 ;BIT04 FAILED TO SET  
1029 1$: BIC #BIT04,@TACS ;NOW CLEAR IT  
1030 004000 042714 000020 BIT #BIT04,@TACS ;DID BIT04 CLEAR?  
1031 004004 032714 000020 BEQ TST27 ;;BR IF YES  
1032 004012 104001 ERROR 1 ;ERROR -- BIT04 OF TACS FAILED TO CLEAR  
1033 ;*****  
1034 ;*TEST 27 TEST BIT04 OF TACS INITIALIZE TO ZERO  
1035 ;*****  
1036 TST27: SCOPE  
1037 004014 000004  
1038 004016 012767 000005 175142 MOV #5,$TIMES ;;DO 5 ITERATIONS  
1039 004024 012714 000020 1$: MOV #BIT04,@TACS ;SET BIT04 OF TACS  
1040 004030 032714 000020 BIT #BIT04,@TACS ;DID BIT04 SET?  
1041 004034 001001 BNE 2$ ;BR IF YES  
1042 004036 104001 ERROR 1 ;BIT04 FAILED TO SET  
1043 2$: RESET ;CLEAR THE WORLD  
1044 004040 000005 BIT #BIT04,@TACS ;DID BIT04 CLEAR?  
1045 004042 032714 000020 BEQ TST30 ;;BR IF YES  
1046 004050 104001 ERROR 1 ;ERROR -- RESET DIDN'T CLEAR BIT04  
1047 ;*****  
1048 ;*TEST 30 TEST BIT03 OF TACS MAY BE "SET" AND "CLEARED"  
1049 ;*****  
1050 TST30: SCOPE  
1051 004052 000004  
1052 004054 052714 000010 BIS #BIT03,@TACS ;SET BIT03 OF TACS  
1053 004060 032714 000010 BIT #BIT03,@TACS ;DID BIT03 SET?  
1054 004064 001001 BNE 1$ ;BR IF YES  
1055 004066 104001 ERROR 1 ;BIT03 FAILED TO SET  
1056 1$: BIC #BIT03,@TACS ;NOW CLEAR IT  
1057 004070 042714 000010 BIT #BIT03,@TACS ;DID BIT03 CLEAR?  
1058 004100 001401 BEQ TST31 ;;BR IF YES  
1059 004102 104001 ERROR 1 ;ERROR -- BIT03 OF TACS FAILED TO CLEAR  
1060 ;*****  
1061 ;*TEST 31 TEST BIT03 OF TACS INITIALIZE TO ZERO  
1062 ;*****  
1063 TST31: SCOPE  
1064 004104 000004  
1065 004106 012767 000005 175052 MOV #5,$TIMES ;;DO 5 ITERATIONS  
1066 004114 012714 000010 1$: MOV #BIT03,@TACS ;SET BIT03 OF TACS  
1067 004120 032714 000010 BIT #BIT03,@TACS ;DID BIT03 SET?  
1068 004124 001001 BNE 2$ ;BR IF YES  
1069 004126 104001 ERROR 1 ;BIT03 FAILED TO SET  
1070 2$: RESET ;CLEAR THE WORLD  
1071 004130 000005 BIT #BIT03,@TACS ;DID BIT03 CLEAR?  
1072 004132 032714 000010
```

```

1072 004136 001401 BEQ TST32 ;;BR IF YES
1073 004140 104001 ERROR 1 ;ERROR -- RESET DIDN'T CLEAR BIT03
1074 ;*****
1075 ;*TEST 32 TEST BIT02 OF TACS MAY BE "SET" AND "CLEARED"
1076 ;*****
1077 004142 000004 TST32: SCOPE
1078 004144 052714 000004 BIS #BIT02,@TACS ;SET BIT02 OF TACS
1079 004150 032714 000004 BIT #BIT02,@TACS ;DID BIT02 SET?
1080 004154 001001 BNE 1$ ;BR IF YES
1081 004156 104001 ERROR 1 ;BIT02 FAILED TO SET
1082
1083 004160 042714 000004 1$: BIC #BIT02,@TACS ;NOW CLEAR IT
1084 004164 032714 000004 BIT #BIT02,@TACS ;DID BIT02 CLEAR?
1085 004170 001401 BEQ TST33 ;;BR IF YES
1086 004172 104001 ERROR 1 ;ERROR -- BIT02 OF TACS FAILED TO CLEAR
1087 ;*****
1088 ;*TEST 33 TEST BIT02 OF TACS INITIALIZE TO ZERO
1089 ;*****
1090 004174 000004 TST33: SCOPE
1091 004176 012767 000005 174762 MOV #5,STIMES ;;DO 5 ITERATIONS
1092 004204 012714 000004 1$: MOV #BIT02,@TACS ;SET BIT02 OF TACS
1093 004210 032714 000004 BIT #BIT02,@TACS ;DID BIT02 SET?
1094 004214 001001 BNE 2$ ;BR IF YES
1095 004216 104001 ERROR 1 ;BIT02 FAILED TO SET
1096
1097 004220 000005 2$: RESET ;CLEAR THE WORLD
1098 004222 032714 000004 BIT #BIT02,@TACS ;DID BIT02 CLEAR?
1099 004226 001401 BEQ TST34 ;;BR IF YES
1100 004230 104001 ERROR 1 ;ERROR -- RESET DIDN'T CLEAR BIT02
1101 ;*****
1102 ;*TEST 34 TEST BIT01 OF TACS MAY BE "SET" AND "CLEARED"
1103 ;*****
1104 004232 000004 TST34: SCOPE
1105 004234 052714 000002 BIS #BIT01,@TACS ;SET BIT01 OF TACS
1106 004240 032714 000002 BIT #BIT01,@TACS ;DID BIT01 SET?
1107 004244 001001 BNE 1$ ;BR IF YES
1108 004246 104001 ERROR 1 ;BIT01 FAILED TO SET
1109
1110 004250 042714 000002 1$: BIC #BIT01,@TACS ;NOW CLEAR IT
1111 004254 032714 000002 BIT #BIT01,@TACS ;DID BIT01 CLEAR?
1112 004260 001401 BEQ TST35 ;;BR IF YES
1113 004262 104001 ERROR 1 ;ERROR -- BIT01 OF TACS FAILED TO CLEAR
1114 ;*****
1115 ;*TEST 35 TEST BIT01 OF TACS INITIALIZE TO ZERO
1116 ;*****
1117 004264 000004 TST35: SCOPE
1118 004266 012767 000005 174672 MOV #5,STIMES ;;DO 5 ITERATIONS
1119 004274 012714 000002 1$: MOV #BIT01,@TACS ;SET BIT01 OF TACS
1120 004300 032714 000002 BIT #BIT01,@TACS ;DID BIT01 SET?
1121 004304 001001 BNE 2$ ;BR IF YES
1122 004306 104001 ERROR 1 ;BIT01 FAILED TO SET
1123
1124 004310 000005 2$: RESET ;CLEAR THE WORLD
1125 004312 032714 000002 BIT #BIT01,@TACS ;DID BIT01 CLEAR?
1126 004316 001401 BEQ TST36 ;;BR IF YES
1127 004320 104001 ERROR 1 ;ERROR -- RESET DIDN'T CLEAR BIT01
    
```

```

1128 ;*****
1129 ;*TEST 36 COUNT PATTERN TO TACS<04:01>
1130 ;*****
1131 004322 000004 TST36: SCOPE
1132 004324 012767 004370 174636 MOV #TST37,$ESCAPE ;;ESCAPE TO TEST 37 ON ERROR
1133 004332 005014 CLR @TACS ;SETUP TACS FOR COUNTING
1134 004334 005000 CLR R0 ;SETUP R0 FOR COUNTING
1135 004336 012702 000002 MOV #2,R2 ;SET THE INCREMENT
1136 004342 011401 1$: MOV @TACS,R1 ;GET TACS
1137 004344 042701 177741 BIC #C36,R1 ;CLEAR THE JUNK
1138 004350 020100 CMP R1,R0 ;COUNT OK?
1139 004352 001401 BEQ 2$ ;BR IF YES
1140 004354 104005 ERROR 5 ;ERROR--BAD COUNT
1141 004356 060200 2$: ADD R2,R0 ;UPDATE R0
1142 004360 060214 ADD R2,@TACS ;COUNT TACS
1143 004362 032700 000036 BIT #36,R0 ;READY?
1144 004366 001365 BNE 1$ ;NO--GO DO TEST COUNT PATTERN
1145
    
```

```
1146 ;////////////////////////////////////
1147 ;////////////////////////////////////
1148 ;THE FOLLOWING TESTS WILL CHECK THE INITIAL STATE OF ALL BITS
1149 ;IN THE TACS THAT ARE DRIVE AND FUNCTION DEPENDENT.
1150 ;NOTE: FOR ALL OF THESE TEST TO PASS
1151 ;THE DRIVE UNDERGOING TEST MUST BE
1152 ;ON CLEAR LEADER AND WRITE ENABLED.
1153 ;////////////////////////////////////
1154 ;*****
1155 ;*TEST 37 TEST INITIAL CONDITION OF TACS BIT05 = 1
1156 ;*****
1157 004370 000004 TST37: SCOPE ;SET SCOPE LOOP ADDRESS
1158 004372 012767 004404 174506 MOV #1s,$LPADR ;SET SCOPE LOOP ADDRESS
1159 004400 000005 RESET ;RESELECT DRIVE
1160 004402 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1161 004404 032714 000040 1s: BIT #BIT05,@TACS ;IS BIT05 OF TACS = 1?
1162 004410 001001 BNE TST40 ;BR IF YES
1163 004412 104001 ERROR 1 ;ERROR -- BIT05 OF TACS DIDN'T INITIALIZE PROPER
1164 ;*****
1165 ;*TEST 40 TEST INITIAL CONDITION OF TACS BIT09 = 0
1166 ;*****
1167 004414 000004 TST40: SCOPE ;SET SCOPE LOOP ADDRESS
1168 004416 012767 004430 174462 MOV #1s,$LPADR ;SET SCOPE LOOP ADDRESS
1169 004424 000005 RESET ;RESELECT DRIVE
1170 004426 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1171 004430 032714 001000 1s: BIT #BIT09,@TACS ;IS BIT09 OF TACS = 0?
1172 004434 001401 BEQ TST41 ;BR IF YES
1173 004436 104001 ERROR 1 ;ERROR -- BIT09 OF TACS DIDN'T INITIALIZE PROPER
1174 ;*****
1175 ;*TEST 41 TEST INITIAL CONDITION OF TACS BIT12 = 0
1176 ;*****
1177 004440 000004 TST41: SCOPE ;SET SCOPE LOOP ADDRESS
1178 004442 012767 004460 174436 MOV #1s,$LPADR ;SET SCOPE LOOP ADDRESS
1179 004450 000005 RESET ;RESELECT DRIVE
1180 004452 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1181 004454 052714 000002 BIS #WRITE,@TACS ;LOAD WRITE FUNCTION
1182 004460 032714 010000 1s: BIT #BIT12,@TACS ;IS BIT12 OF TACS = 0?
1183 004464 001401 BEQ TST42 ;BR IF YES
1184 004466 104001 ERROR 1 ;ERROR -- BIT12 OF TACS DIDN'T INITIALIZE PROPER
1185 ;*****
1186 ;*TEST 42 TEST INITIAL CONDITION OF TACS BIT13 = 1
1187 ;*****
1188 004470 000004 TST42: SCOPE ;SET SCOPE LOOP ADDRESS
1189 004472 012767 004504 174406 MOV #1s,$LPADR ;SET SCOPE LOOP ADDRESS
1190 004500 000005 RESET ;RESELECT DRIVE
1191 004502 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1192 004504 032714 020000 1s: BIT #BIT13,@TACS ;IS BIT13 OF TACS = 1?
1193 004510 001001 BNE TST43 ;BR IF YES
1194 004512 104001 ERROR 1 ;ERROR -- BIT13 OF TACS DIDN'T INITIALIZE PROPER
1195 ;*****
1196 ;*TEST 43 TEST INITIAL CONDITION OF TACS BIT15 = 1
1197 ;*****
1198 004514 000004 TST43: SCOPE ;SET SCOPE LOOP ADDRESS
1199 004516 012767 004530 174362 MOV #1s,$LPADR ;SET SCOPE LOOP ADDRESS
1200 004524 000005 RESET ;RESELECT DRIVE
1201 004526 010314 MOV DRIVE,@TACS ;SELECT DRIVE
```

```
1202 004530 032714 100000 1s: BIT #BIT15,@TACS ;IS BIT15 OF TACS = 1?
1203 004534 001001 BNE TST44 ;BR IF YES
1204 004536 104001 ERROR 1 ;ERROR -- BIT15 OF TACS DIDN'T INITIALIZE PROPER
```

```
1205 ;////////////////////////////////////
1206 ;////////////////////////////////////
1207 ;THE FOLLOWING TESTS CHECKS EACH READ ONLY
1208 ;BIT OF THE TACS FOR READ ONLY CAPABILITY
1209 ;////////////////////////////////////
1210 ;*****
1211 ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT15.
1212 ;THEN CHECK IT TO SEE IF IT CHANGED.
1213 ;*****
1214 ;*TEST 44 TEST TACS BIT15 IS READ ONLY
1215 ;*****
1216 004540 000004 TST44: SCOPE
1217 004542 012767 004552 174336 MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
1218 004550 000005 RESET ;;INITIALIZE ALL BITS
1219 004552 011400 1S: MOV @TACS,R0 ;GET TACS FOR COMPARING
1220 004554 010001 MOV R0,R1 ;PUT STATUS IN R1
1221 004556 005101 COM R1 ;COMPLEMENT R1
1222 004560 042700 077777 BIC #C<BIT15>,R0 ;GET RID OF UNDESIRE BITS
1223 004564 042701 077777 BIC #C<BIT15>,R1
1224 004570 010114 MOV R1,@TACS ;TRY TO CHANGE BIT15
1225 004572 011401 MOV @TACS,R1 ;READ IT BACK
1226 004574 042701 077777 BIC #C<BIT15>,R1 ;ONLY LOOK AT BIT15
1227 004600 020001 CMP R0,R1 ;DID IT CHANGE?
1228 004602 001401 BEQ TST45 ;;BR IF NO
1229 004604 104001 ERROR 1 ;ERROR -- BIT15 CHANGED
1230 ;*****
1231 ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT14.
1232 ;THEN CHECK IT TO SEE IF IT CHANGED.
1233 ;*****
1234 ;*TEST 45 TEST TACS BIT14 IS READ ONLY
1235 ;*****
1236 004606 000004 TST45: SCOPE
1237 004610 012767 004620 174270 MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
1238 004616 000005 RESET ;;INITIALIZE ALL BITS
1239 004620 011400 1S: MOV @TACS,R0 ;GET TACS FOR COMPARING
1240 004622 010001 MOV R0,R1 ;PUT STATUS IN R1
1241 004624 005101 COM R1 ;COMPLEMENT R1
1242 004626 042700 137777 BIC #C<BIT14>,R0 ;GET RID OF UNDESIRE BITS
1243 004632 042701 137777 BIC #C<BIT14>,R1
1244 004636 010114 MOV R1,@TACS ;TRY TO CHANGE BIT14
1245 004640 011401 MOV @TACS,R1 ;READ IT BACK
1246 004642 042701 137777 BIC #C<BIT14>,R1 ;ONLY LOOK AT BIT14
1247 004646 020001 CMP R0,R1 ;DID IT CHANGE?
1248 004650 001401 BEQ TST46 ;;BR IF NO
1249 004652 104001 ERROR 1 ;ERROR -- BIT14 CHANGED
1250 ;*****
1251 ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT13.
1252 ;THEN CHECK IT TO SEE IF IT CHANGED.
1253 ;*****
1254 ;*TEST 46 TEST TACS BIT13 IS READ ONLY
1255 ;*****
1256 004654 000004 TST46: SCOPE
1257 004656 012767 004666 174222 MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
1258 004664 000005 RESET ;;INITIALIZE ALL BITS
1259 004666 011400 1S: MOV @TACS,R0 ;GET TACS FOR COMPARING
1260 004670 010001 MOV R0,R1 ;PUT STATUS IN R1
```

```
1261 004672 005101 COM R1 ;COMPLEMENT R1
1262 004674 042700 BIC #C<BIT13>,R0 ;GET RID OF UNDESIRE BITS
1263 004700 042701 157777 BIC #C<BIT13>,R1
1264 004704 010114 MOV R1,@TACS ;TRY TO CHANGE BIT13
1265 004706 011401 MOV @TACS,R1 ;READ IT BACK
1266 004710 042701 157777 BIC #C<BIT13>,R1 ;ONLY LOOK AT BIT13
1267 004714 020001 CMP R0,R1 ;DID IT CHANGE?
1268 004716 001401 BEQ TST47 ;;BR IF NO
1269 004720 104001 ERROR 1 ;ERROR -- BIT13 CHANGED
1270 ;*****
1271 ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT12.
1272 ;THEN CHECK IT TO SEE IF IT CHANGED.
1273 ;*****
1274 ;*TEST 47 TEST TACS BIT12 IS READ ONLY
1275 ;*****
1276 004722 000004 TST47: SCOPE
1277 004724 012767 004734 174154 MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
1278 004732 000005 RESET ;;INITIALIZE ALL BITS
1279 004734 011400 1S: MOV @TACS,R0 ;GET TACS FOR COMPARING
1280 004736 010001 MOV R0,R1 ;PUT STATUS IN R1
1281 004740 005101 COM R1 ;COMPLEMENT R1
1282 004742 042700 167777 BIC #C<BIT12>,R0 ;GET RID OF UNDESIRE BITS
1283 004746 042701 167777 BIC #C<BIT12>,R1
1284 004752 010114 MOV R1,@TACS ;TRY TO CHANGE BIT12
1285 004754 011401 MOV @TACS,R1 ;READ IT BACK
1286 004756 042701 167777 BIC #C<BIT12>,R1 ;ONLY LOOK AT BIT12
1287 004762 020001 CMP R0,R1 ;DID IT CHANGE?
1288 004764 001401 BEQ TST50 ;;BR IF NO
1289 004766 104001 ERROR 1 ;ERROR -- BIT12 CHANGED
1290 ;*****
1291 ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT11.
1292 ;THEN CHECK IT TO SEE IF IT CHANGED.
1293 ;*****
1294 ;*TEST 50 TEST TACS BIT11 IS READ ONLY
1295 ;*****
1296 004770 000004 TST50: SCOPE
1297 004772 012767 005002 174106 MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
1298 005000 000005 RESET ;;INITIALIZE ALL BITS
1299 005002 011400 1S: MOV @TACS,R0 ;GET TACS FOR COMPARING
1300 005004 010001 MOV R0,R1 ;PUT STATUS IN R1
1301 005006 005101 COM R1 ;COMPLEMENT R1
1302 005010 042700 173777 BIC #C<BIT11>,R0 ;GET RID OF UNDESIRE BITS
1303 005014 042701 173777 BIC #C<BIT11>,R1
1304 005020 010114 MOV R1,@TACS ;TRY TO CHANGE BIT11
1305 005022 011401 MOV @TACS,R1 ;READ IT BACK
1306 005024 042701 173777 BIC #C<BIT11>,R1 ;ONLY LOOK AT BIT11
1307 005030 020001 CMP R0,R1 ;DID IT CHANGE?
1308 005032 001401 BEQ TST51 ;;BR IF NO
1309 005034 104001 ERROR 1 ;ERROR -- BIT11 CHANGED
1310 ;*****
1311 ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT10.
1312 ;THEN CHECK IT TO SEE IF IT CHANGED.
1313 ;*****
1314 ;*TEST 51 TEST TACS BIT10 IS READ ONLY
1315 ;*****
1316 005036 000004 TST51: SCOPE
```

```

1317 005040 012767 005050 174040 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1318 005046 000005 RESET ;;INITIALIZE ALL BITS
1319 005050 011400 1S: MOV @TACS,R0 ;GET TACS FOR COMPARING
1320 005052 010001 MOV R0,R1 ;PUT STATUS IN R1
1321 005054 005101 COM R1 ;COMPLEMENT R1
1322 005056 042700 175777 BIC #C<BIT10>,R0 ;GET RID OF UNDESIRED BITS
1323 005062 042701 175777 BIC #C<BIT10>,R1
1324 005066 010114 MOV R1,@TACS ;TRY TO CHANGE BIT10
1325 005070 011401 MOV @TACS,R1 ;READ IT BACK
1326 005072 042701 175777 BIC #C<BIT10>,R1 ;ONLY LOOK AT BIT10
1327 005076 020001 CMP R0,R1 ;DID IT CHANGE?
1328 005100 001401 BEQ TST52 ;;BR IF NO
1329 005102 104001 ERROR 1 ;ERROR -- BIT10 CHANGED
1330 ;*****
1331 ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT10.
1332 ;THEN CHECK IT TO SEE IF IT CHANGED.
1333 ;*****
1334 ;*TEST 52 TEST TACS BIT09 IS READ ONLY
1335 ;*****
1336 005104 000004 TST52: SCOPE
1337 005106 012767 005116 173772 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1338 005114 000005 RESET ;;INITIALIZE ALL BITS
1339 005116 011400 1S: MOV @TACS,R0 ;GET TACS FOR COMPARING
1340 005120 010001 MOV R0,R1 ;PUT STATUS IN R1
1341 005122 005101 COM R1 ;COMPLEMENT R1
1342 005124 042700 176777 BIC #C<BIT09>,R0 ;GET RID OF UNDESIRED BITS
1343 005130 042701 176777 BIC #C<BIT09>,R1
1344 005134 010114 MOV R1,@TACS ;TRY TO CHANGE BIT09
1345 005136 011401 MOV @TACS,R1 ;READ IT BACK
1346 005140 042701 176777 BIC #C<BIT09>,R1 ;ONLY LOOK AT BIT09
1347 005144 020001 CMP R0,R1 ;DID IT CHANGE?
1348 005146 001401 BEQ TST53 ;;BR IF NO
1349 005150 104001 ERROR 1 ;ERROR -- BIT09 CHANGED
1350 ;*****
1351 ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT09.
1352 ;THEN CHECK IT TO SEE IF IT CHANGED.
1353 ;*****
1354 ;*TEST 53 TEST TACS BIT07 IS READ ONLY
1355 ;*****
1356 005152 000004 TST53: SCOPE
1357 005154 012767 005164 173724 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1358 005162 000005 RESET ;;INITIALIZE ALL BITS
1359 005164 011400 1S: MOV @TACS,R0 ;GET TACS FOR COMPARING
1360 005166 010001 MOV R0,R1 ;PUT STATUS IN R1
1361 005170 005101 COM R1 ;COMPLEMENT R1
1362 005172 042700 177577 BIC #C<BIT07>,R0 ;GET RID OF UNDESIRED BITS
1363 005176 042701 177577 BIC #C<BIT07>,R1
1364 005202 010114 MOV R1,@TACS ;TRY TO CHANGE BIT07
1365 005204 011401 MOV @TACS,R1 ;READ IT BACK
1366 005206 042701 177577 BIC #C<BIT07>,R1 ;ONLY LOOK AT BIT07
1367 005212 020001 CMP R0,R1 ;DID IT CHANGE?
1368 005214 001401 BEQ TST54 ;;BR IF NO
1369 005216 104001 ERROR 1 ;ERROR -- BIT07 CHANGED
1370 ;*****
1371 ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT07.
1372 ;THEN CHECK IT TO SEE IF IT CHANGED.

```

```

1373 ;*****
1374 ;*TEST 54 TEST TACS BIT05 IS READ ONLY
1375 ;*****
1376 005220 000004 TST54: SCOPE
1377 005222 012767 005232 173656 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1378 005230 000005 RESET ;;INITIALIZE ALL BITS
1379 005232 011400 1S: MOV @TACS,R0 ;GET TACS FOR COMPARING
1380 005234 010001 MOV R0,R1 ;PUT STATUS IN R1
1381 005236 005101 COM R1 ;COMPLEMENT R1
1382 005240 042700 177737 BIC #C<BIT05>,R0 ;GET RID OF UNDESIRED BITS
1383 005244 042701 177737 BIC #C<BIT05>,R1
1384 005250 010114 MOV R1,@TACS ;TRY TO CHANGE BIT05
1385 005252 011401 MOV @TACS,R1 ;READ IT BACK
1386 005254 042701 177737 BIC #C<BIT05>,R1 ;ONLY LOOK AT BIT05
1387 005260 020001 CMP R0,R1 ;DID IT CHANGE?
1388 005262 001401 BEQ TST55 ;;BR IF NO
1389 005264 104001 ERROR 1 ;ERROR -- BIT05 CHANGED
1390 ;*****

```

```
1391 ;////////////////////////////////////
1392 ;////////////////////////////////////
1393 ;THE FOLLOWING TESTS WILL CHECK FOR DUAL AND/OR FAULTY SELECTION
1394 ;OF THE TA11 ADDRESSES
1395 ;////////////////////////////////////
1396 ;*****
1397 ;*TEST 55 TEST HIGH BYTE SELECTION OF TACS
1398 ;*****
1399 005266 000004 TST55: SCOPE
1400 005270 012767 005300 173610 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1401 005276 000005 RESET
1402 005300 012714 1s: MOV #BIT02,@TACS ;SET TACS BIT02
1403 005304 105077 173700 CLR #TACSH ;CLEAR THE HIGH BYTE
1404 005310 132714 000004 BIT #BIT02,@TACS ;IS BIT02 STILL SET?
1405 005314 001001 BNE TST56 ;;BR IF YES
1406 005316 104001 ERROR 1 ;CLRB TO HIGH BYTE OF TACS CLEARED LOW BYTE
1407
1408 ;*****
1409 ;*TEST 56 TEST LOW BYTE SELECTION OF TACS
1410 ;*****
1411 005320 000004 TST56: SCOPE
1412 005322 012767 005332 173556 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1413 005332 012714 000400 RESET
1414 005336 105014 1s: MOV #BIT08,@TACS ;SET BIT08 OF TACS
1415 005340 132777 000001 173642 CLR #TACSH ;CLEAR THE LOW BYTE OF TACS
1416 005346 001001 BIT #BIT08,@TACSH ;IS BIT08 OF TACS STILL SET?
1417 005350 104001 BNE TST57 ;;BR IF YES
1418 ERROR 1 ;CLRB TO LOW BYTE OF TACS CLEARED THE HIGH BYTE
1419
1420 ;*****
1421 ;*TEST 57 TEST "OUT LOW" SIGNAL
1422 ;*****
1423 005352 000004 TST57: SCOPE
1424 005354 012767 005412 173606 MOV #TST60,$ESCAPE ;;ESCAPE TO TEST 60 ON ERROR
1425 005362 005014 CLR #TACS ;ZERO THE TA11 STATUS
1426 005364 032714 000536 BIT #UNIT!INT,EN!ILBS!FUNCTION,@TACS
1427 005370 001401 BEQ 1s
1428 005372 104001 ERROR 1 ;TACS DIDN'T CLEAR PROPERLY
1429 005374 112777 173606 1s: MOV #INT,EN!ILBS!FUNCTION,@TACSH ;LOAD HIGH BYTE OF TACS
1430 005402 032714 000136 BIT #INT,EN!ILBS!FUNCTION,@TACS ;CHECK LOW BYTE
1431 005406 001401 BEQ TST60 ;;BR IF LOW BYTE IS OK
1432 005410 104001 ERROR 1 ;LOW BYTE IMPROPERLY SELECTED
1433
1434 ;*****
1435 ;*TEST 60 TEST "OUT HIGH" SIGNAL
1436 ;*****
1437 005412 000004 TST60: SCOPE
1438 005414 012767 005450 173546 MOV #TST61,$ESCAPE ;;ESCAPE TO TEST 61 ON ERROR
1439 005422 005014 CLR #TACS ;ZERO THE TA11 STATUS
1440 005424 032714 000536 BIT #UNIT!INT,EN!ILBS!FUNCTION,@TACS
1441 005430 001401 BEQ 1s
1442 005432 104001 ERROR 1 ;TACS DIDN'T CLEAR PROPERLY
1443 005434 112714 000400 1s: MOV #BIT08,@TACS ;LOAD LOW BYTE OF TACS
1444 005440 032714 000400 BIT #BIT08,@TACS ;DID BIT08 GET SET?
1445 005444 001401 BEQ TST61 ;;BR IF NO
1446 005446 104001 ERROR 1 ;HIGH BYTE IMPROPERLY SELECTED
1447
1448 ;*****
1449 ;*TEST 61 TEST "SELECT 00" ISN'T STUCK
1450 ;*****
```

```
1447 005450 000004 TST61: SCOPE
1448 005452 012767 005506 173510 MOV #TST62,$ESCAPE ;;ESCAPE TO TEST 62 ON ERROR
1449 005460 005014 CLR #TACS ;CLEAR TA11 STATUS
1450 005462 032714 000536 BIT #UNIT!INT,EN!ILBS!FUNCTION,@TACS
1451 005466 001401 BEQ 1s
1452 005470 104001 ERROR 1 ;TACS DIDN'T CLEAR PROPERLY
1453 005472 012715 177776 1s: MOV #C1,@TADB ;LOAD THE DATA BUFFER
1454 005476 032714 000536 BIT #UNIT!INT,EN!ILBS!FUNCTION,@TACS
1455 005502 001401 BEQ TST62 ;;BR IF TACS DIDN'T CHANGE
1456 005504 104001 ERROR 1 ;TACS CHANGED WHEN LOADING TADB
1457
1458 ;*****
1459 ;THE FOLLOWING TEST INSURES THAT THE STATE OF THE "ERROR" INDICATOR IS
1460 ;PROPER FOR ALL FUNCTIONS WHEN THE TAPE IS AT "CLEAR LEADER"
1461 ;*****
1462 ;*TEST 62 TEST "ERROR" FOR "CLEAR LEADER"
1463 ;*****
1464
1465 005506 000004 TST62: SCOPE
1466 005510 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1467 005512 032714 020000 BIT #LEADER,@TACS ;CHECK FOR CLEAR LEADER
1468 005516 001001 BNE 1s ;BR IF ON CLEAR LEADER
1469 005520 104001 ERROR 1 ;NOT ON CLEAR LEADER
1470 005522 112714 000000 1s: MOV #WFG,@TACS ;CHECK "ERROR" WITH "WFG"
1471 005526 005714 TST #TACS ;SAMPLE THE "ERROR" BIT
1472 005530 100401 BMI 2s ;BR IF "ERROR" = 1
1473 005532 104001 ERROR 1 ;"ERROR" NOT = 1
1474 005534 112714 000002 2s: MOV #WRITE,@TACS ;CHECK "ERROR" WITH "WRITE"
1475 005540 005714 TST #TACS ;SAMPLE THE "ERROR" BIT
1476 005542 100401 BMI 3s ;BR IF "ERROR" = 1
1477 005544 104001 ERROR 1 ;"ERROR" NOT = 1
1478 005546 112714 000004 3s: MOV #READ,@TACS ;CHECK "ERROR" WITH "READ"
1479 005552 005714 TST #TACS ;SAMPLE THE "ERROR" BIT
1480 005554 100401 BMI 4s ;BR IF "ERROR" = 1
1481 005556 104001 ERROR 1 ;"ERROR" NOT = 1
1482 005560 112714 000006 4s: MOV #BSFG,@TACS ;CHECK "ERROR" WITH "BSFG"
1483 005564 005714 TST #TACS ;SAMPLE THE "ERROR" BIT
1484 005566 100401 BMI 5s ;BR IF "ERROR" = 1
1485 005570 104001 ERROR 1 ;"ERROR" NOT = 1
1486 005572 112714 000010 5s: MOV #BSBG,@TACS ;CHECK "ERROR" WITH "BSBG"
1487 005576 005714 TST #TACS ;SAMPLE THE "ERROR" BIT
1488 005600 100401 BMI 6s ;BR IF "ERROR" = 1
1489 005602 104001 ERROR 1 ;"ERROR" NOT = 1
1490 005604 112714 000012 6s: MOV #SFFG,@TACS ;CHECK "ERROR" WITH "SFFG"
1491 005610 005714 TST #TACS ;SAMPLE THE "ERROR" BIT
1492 005612 100401 BMI 7s ;BR IF "ERROR" = 1
1493 005614 104001 ERROR 1 ;"ERROR" NOT = 1
1494 005616 112714 000014 7s: MOV #SFBG,@TACS ;CHECK "ERROR" WITH "SFBG"
1495 005622 005714 TST #TACS ;SAMPLE THE "ERROR" BIT
1496 005624 100401 BMI 8s ;BR IF "ERROR" = 1
1497 005626 104001 ERROR 1 ;"ERROR" NOT = 1
1498 005630 112714 000016 8s: MOV #REWIND,@TACS ;CHECK "ERROR" WITH "REWIND"
1499 005634 005714 TST #TACS ;SAMPLE THE "ERROR" BIT
1500 005636 100001 BPL TST63 ;;BR IF "ERROR" = 0
1501 005640 104001 ERROR 1 ;"ERROR" NOT = 0
```

```
1502 ;////////////////////////////////////
1503 ;////////////////////////////////////
1504 ;THE FOLLOWING TESTS WILL CHECK BASIC OPERATIONS OF THE
1505 ;"REWIND" AND "WRITE FILE GAP" FUNCTIONS.
1506 ;////////////////////////////////////
1507 ;*****
1508 ;*TEST 63 TEST THAT "READY" SETS WHEN "REWIND" IS COMPLETED
1509 ;*****
1510 005642 000004 TST63: SCOPE
1511 005644 012767 000005 173314 MOV #5,$TIMES ;DO 5 ITERATIONS
1512 005652 012767 005670 173226 MOV #3,$LPADR ;SET SCOPE LOOP ADDRESS
1513 005660 000005 RESET
1514 005662 012767 005742 173300 MOV #TST64,$ESCAPE ;ESCAPE TO TEST 64 ON ERROR
1515 005670 010314 36: MOV DRIVE,@TACS ;SELECT DRIVE
1516 005672 005001 CLR R1 ;WASTE SOME TIME
1517 005674 012700 MOV #100,,R0
1518 005700 112714 000017 MOVB #REWIND!GO,@TACS ;START A REWIND
1519 005704 032714 18: BIT #READY,@TACS ;SAMPLE READY
1520 005710 001010 BNE 2$ ;GET OUT IF READY IS SET
1521 005712 005201 INC R1
1522 005714 001373 BNE 1$
1523 005716 005300 DEC R0
1524 005720 001371 BNE 1$
1525 005722 032714 000040 BIT #READY,@TACS ;IS "READY" BIT SET?
1526 005726 001001 BNE 2$ ;BR IF YES
1527 005730 104001 ERROR 1 ;ERROR--"READY" BIT DIDN'T SET
1528 005732 032714 020000 26: BIT #LEADER,@TACS ;CHECK FOR CLEAR LEADER
1529 005736 001001 BNE TST64 ;BR IF CLEAR LEADER NE 0
1530 005740 104001 ERROR 1 ;NOT ON CLEAR LEADER
1531 ;AFTER DOING A REWIND
1532 ;*****
1533 ;*TEST 64 TEST THAT "READY" SETS WHEN "WRITE FILE GAP" IS COMPLETED
1534 ;*****
1535 005742 000004 TST64: SCOPE
1536 005744 012767 000005 173214 MOV #5,$TIMES ;DO 5 ITERATIONS
1537 005752 012767 005770 173126 MOV #3,$LPADR ;SET SCOPE LOOP ADDRESS
1538 005760 000005 RESET
1539 005762 012767 006042 173200 MOV #TST65,$ESCAPE ;ESCAPE TO TEST 65 ON ERROR
1540 005770 010314 36: MOV DRIVE,@TACS ;SELECT DRIVE
1541 005772 005001 CLR R1 ;WASTE SOME TIME
1542 005774 012700 000144 MOV #100,,R0
1543 006000 112714 000001 MOVB #WFG!GO,@TACS ;START A WFG
1544 006004 032714 18: BIT #READY,@TACS ;SAMPLE READY
1545 006010 001010 BNE 2$ ;GET OUT IF READY IS SET
1546 006012 005201 INC R1
1547 006014 001373 BNE 1$
1548 006016 005300 DEC R0
1549 006020 001371 BNE 1$
1550 006022 032714 000040 BIT #READY,@TACS ;IS "READY" BIT SET?
1551 006026 001001 BNE 2$ ;BR IF YES
1552 006030 104001 ERROR 1 ;ERROR--"READY" BIT DIDN'T SET
1553 006032 032714 020000 26: BIT #LEADER,@TACS ;CHECK FOR CLEAR LEADER
1554 006036 001401 BEQ TST65 ;BR IF CLEAR LEADER EQ 0
1555 006040 104001 ERROR 1 ;ON CLEAR LEADER
1556 ;AFTER DOING A WRITE FILE GAP
1557 ;*****
```

```
1558 ;*TEST 65 TEST "READY" CLEARS WHEN STARTING A "REWIND" FUNCTION
1559 ;*****
1560 006042 000004 TST65: SCOPE
1561 006044 012767 000005 173114 MOV #5,$TIMES ;DO 5 ITERATIONS
1562 006052 000005 RESET
1563 006054 012701 000001 MOV #1,R1 ;SET TIMER
1564 006060 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1565 006062 104412 WAITREADY ;WAIT FOR "READY"
1566 006064 112714 000017 MOVB #REWIND!GO,@TACS ;START A "REWIND"
1567 006070 032714 18: BIT #READY,@TACS ;CHECK "READY" BIT
1568 006074 001403 BEQ TST66 ;BR IF "READY" = 0
1569 006076 006301 ASL R1 ;IS TIME UP?
1570 006100 001373 BNE 1$ ;BR IF NO
1571 006102 104001 ERROR 1 ;"READY" FAILED TO CLEAR
1572 ;*****
1573 ;*TEST 66 TEST "READY" CLEARS WHEN STARTING A "WRITE FILE GAP" FUNCTION
1574 ;*****
1575 006104 000004 TST66: SCOPE
1576 006106 012767 000005 173052 MOV #5,$TIMES ;DO 5 ITERATIONS
1577 006114 000005 RESET
1578 006116 012701 000001 MOV #1,R1 ;SET TIMER
1579 006122 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1580 006124 104412 WAITREADY ;WAIT FOR "READY"
1581 006126 112714 000001 MOVB #WFG!GO,@TACS ;START A "WRITE FILE GAP"
1582 006132 032714 18: BIT #READY,@TACS ;CHECK "READY" BIT
1583 006136 001403 BEQ TST67 ;BR IF "READY" = 0
1584 006140 006301 ASL R1 ;IS TIME UP?
1585 006142 001373 BNE 1$ ;BR IF NO
1586 006144 104001 ERROR 1 ;"READY" FAILED TO CLEAR
1587 ;*****
1588 ;*TEST 67 TEST THAT "RESET" CANNOT ABORT A "REWIND"
1589 ;*****
1590 006146 000004 TST67: SCOPE
1591 006150 012767 MOV #1,$TIMES ;DO 1 ITERATION
1592 006156 012767 006234 173004 MOV #5,$ESCAPE ;ESCAPE TO 5$ ON ERROR
1593 006164 000005 RESET
1594 006166 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1595 006170 104412 WAITREADY ;GO WAIT FOR "READY" TO SET
1596 006172 112714 000017 MOVB #REWIND!GO,@TACS ;LOAD REWIND AND GO
1597 006176 005067 000010 CLR 3$
1598 006202 032714 26: BIT #READY,@TACS ;WAIT FOR READY TO CLEAR
1599 006206 001404 BEQ 4$
1600 006210 105227 INCB (PC)+
1601 006212 000000 36: 0
1602 006214 001372 BNE 2$
1603 006216 104001 ERROR 1 ;"READY" FAILED TO CLEAR
1604 006220 000005 46: RESET ;TRY TO STOP REWIND
1605 006222 050314 BIS DRIVE,@TACS ;SELECT DRIVE
1606 006224 032714 000040 BIT #READY,@TACS ;IS READY SET?
1607 006230 001401 BEQ 5$ ;BR IF NO
1608 006232 104001 ERROR 1 ;"READY" WAS SET
1609 006234 104412 56: WAITREADY ;WAIT ON READY TO SET
1610 ;*****
1611 ;THIS TEST WILL START A "WRITE FILE GAP" AND WAIT FOR "READY" TO CLEAR
1612 ;AS SOON AS "READY" CLEARS A "RESET" IS PERFORMED (WHICH SHOULD
1613 ;CLEAR THE FUNCTION AND SET READY) THEN "READY" IS EXAMINED
```



```

1614 ;TO INSURE THAT IT IS SET
1615 ;*****
1616 ;*TEST 70 TEST "RESET" WILL ABORT A "WRITE FILE GAP"
1617 ;*****
1618 006236 000004 TST70: SCOPE
1619 006240 012767 000005 172720 MOV #5,$TIMES ;;DO 5 ITERATIONS
1620 006246 012767 006264 172632 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1621 006254 012767 006332 172706 MOV #TST71,$ESCAPE ;;ESCAPE TO TEST 71 ON ERROR
1622 006262 000005 RESET
1623 006264 010314 18: MOV DRIVE,@TACS ;SELECT DRIVE
1624 006266 104412 WAITREADY ;WAIT ON READY
1625 006270 112714 000001 MOVB #WFG!GO,@TACS ;START A WRITE FILE GAP FUNCTION
1626 006274 005067 000010 CLR 3S
1627 006300 032714 28: BIT #READY,@TACS ;WAIT FOR READY TO CLEAR
1628 006304 001404 BEQ 4S ;BR IF "READY" = 0
1629 006306 105227 INCB (PC)+ ;WASTE SOME TIME
1630 006310 000000 38: 0
1631 006312 001372 BNE 2S
1632 006314 104001 ERROR 1 ;"READY" FAILED TO CLEAR
1633 006316 000005 48: RESET ;KILL WRITE FILE GAP
1634 006320 010314 MOV DRIVE,@TACS
1635 006322 032714 000040 BIT #READY,@TACS
1636 006326 001001 BNE TST71 ;;BR IF "RESET" SET "READY"
1637 006330 104001 ERROR 1 ;"READY" DIDN'T SET
1638 ;*****
1639 ;*TEST 71 TEST "WFG" AND "REWIND" FOR NO ERRORS
1640 ;*****
1641 006332 000004 TST71: SCOPE
1642 006334 012767 000005 172624 MOV #5,$TIMES ;;DO 5 ITERATIONS
1643 006342 012767 006362 172536 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1644 006350 012767 006424 172612 MOV #TST72,$ESCAPE ;;ESCAPE TO TEST 72 ON ERROR
1645 006356 000005 RESET
1646 006360 010314 MOV DRIVE,@TACS
1647 006362 112714 000017 18: MOVB #REWIND!GO,@TACS ;GO TO "CLEAR LEADER"
1648 006366 104412 WAITREADY ;GO WAIT FOR "READY" TO SET
1649 006370 005714 TST @TACS ;IS "ERROR" SET?
1650 006372 100001 BPL 2S ;BR IF NO
1651 006374 104001 ERROR 1 ;"ERROR" = 1 AFTER "REWIND"
1652 006376 032714 020000 28: BIT #LEADER,@TACS ;IS "CLEAR LEADER" = 1?
1653 006402 001001 BNE 3S ;BR IF YES
1654 006404 104001 ERROR 1 ;NOT ON "CLEAR LEADER" AFTER "REWIND"
1655 006406 112714 000001 38: MOVB #WFG!GO,@TACS ;GET OFF OF "CLEAR LEADER"
1656 006412 104412 WAITREADY ;GO WAIT FOR "READY" TO SET
1657 006414 032714 120000 BIT #ERROR!LEADER,@TACS ;SHOULD BE OFF OF "CLEAR LEADER"
1658 ;WITH NO "ERROR"
1659 006420 001401 BEQ ,+4 ;BR IF "WFG" MOVED TAPE WITH NO "ERROR"
1660 006422 104001 ERROR 1 ;"WFG" FAILED
1661 ;*****
1662 ;THIS ISN'T A REAL TEST BUT A SMALL ROUTINE TO DETERMINE THE MAX.
1663 ;TIME FOR THE WAIT LOOPS (WAIT FOR "READY" AND "TRANSFER REQUEST")
1664 ;*****
1665 ;*TEST 72 ROUTINE TO DETERMINE TIME OF WAIT LOOPS
1666 ;*****
1667 TST72: SCOPE
1668 006424 000004 MOV #1,$TIMES ;;DO 1 ITERATION
1669 006426 012767 000001 172532
    
```

```

1670 006434 005737 001100 TST @SPASS ;IS THIS THE FIRST PASS?
1671 006440 001021 BNE TST73 ;BR IF NO
1672 006442 000005 RESET
1673 006444 010314 MOV DRIVE,@TACS ;SELECT THE DRIVE
1674 006446 112714 000017 MOVB #REWIND!GO,@TACS ;START A REWIND
1675 006452 104412 WAITREADY ;WAIT FOR READY
1676 006454 112714 000001 MOVB #WFG!GO,@TACS ;WRITE A FILE GAP
1677 006460 104412 WAITREADY ;WAIT ON READY
1678 006462 163737 012072 012116 SUB @HGHNTIM,@MAXCNT ;GET THE TIME IT TOOK
1679 006470 005237 012116 INC @MAXCNT ;MAKE IT BIGGER
1680 006474 006137 012116 ROL @MAXCNT
1681 006500 006137 012116 ROL @MAXCNT
    
```

```
1682 ;////////////////////////////////////  
1683 ;////////////////////////////////////  
1684 ;THE FOLLOWING TESTS ARE USED TO CHECK VARIOUS OPERATIONS WHEN AT "CLEAR LEADER"  
1685 ;////////////////////////////////////  
1686 ;*****  
1687 ;*TEST 73 TEST THAT "CLEAR LEADER" GOES FALSE ON "REIND"  
1688 ;*****  
1689 TST73: SCOPE  
1690 006504 000004 MOV #5,$TIMES ;DO 5 ITERATIONS  
1691 006506 012767 000005 172452 MOV #6,$ESCAPE ;ESCAPE TO 6S ON ERROR  
1692 006522 000005 RESET  
1693 006524 010314 MOV DRIVE,@TACS ;SELECT DRIVE  
1694 006526 112714 000017 MOVB #REWINDGO,@TACS ;GO TO CLEAR LEADER  
1695 006532 104412 WAITREADY ;GO WAIT FOR "READY" TO SET  
1696 006534 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?  
1697 006540 001001 BNE 2$ ;BR IF YES  
1698 006542 104001 ERROR 1 ;ERROR--NOT AT CLEAR LEADER  
1699 006544 005214 2$: INC @TACS ;START ANOTHER REWIND  
1700 006546 005067 000010 CLR 4$  
1701 006552 032714 000040 3$: BIT #READY,@TACS ;WAIT ON "READY" TO CLEAR  
1702 006556 001404 BEQ 5$  
1703 006560 105227 INCB (PC)+  
1704 006562 000000 4$: 0  
1705 006564 001372 BNE 3$  
1706 006566 104001 ERROR 1 ;"READY" FAILED TO CLEAR  
1707 006570 032714 020000 5$: BIT #LEADER,@TACS ;DID "CLEAR LEADER" GO FALSE?  
1708 006574 001401 BEQ 6$ ;BR IF YES  
1709 006576 104001 ERROR 1 ;CLEAR LEADER DIDN'T CLEAR  
1710 006600 005067 172364 6$: CLR $ESCAPE  
1711 006604 104412 WAITREADY  
1712 ;*****  
1713 ;*TEST 74 TEST FOR "CLEAR LEADER ERROR" FOR "WRITE FILE GAP FUNCTION"  
1714 ;*****  
1715 TST74: SCOPE  
1716 006610 012767 000005 172350 MOV #5,$TIMES ;DO 5 ITERATIONS  
1717 006616 012767 006646 172262 MOV #1,$LPADR ;SET SCOPE LOOP ADDRESS  
1718 006624 012767 006722 172336 MOV #TST75,$ESCAPE ;ESCAPE TO TEST 75 ON ERROR  
1719 006632 000005 RESET  
1720 006634 010314 MOV DRIVE,@TACS ;SELECT DRIVE  
1721 006636 104412 WAITREADY ;MAKE SURE TA11 IS READY  
1722 006640 112714 000017 MOVB #REWIND+GO,@TACS ;LOAD REWIND AND GO  
1723 006644 104412 WAITREADY ;GO WAIT FOR "READY" TO SET  
1724 006646 012700 000001 1$: MOV #1,R0  
1725 006652 010314 MOV DRIVE,@TACS  
1726 006654 112714 000001 MOVB #WFGIGO,@TACS ;START A WRITE FILE GAP FUNCTION  
1727 006660 032714 000040 3$: BIT #READY,@TACS ;WAIT ON READY TO CLEAR  
1728 006664 001402 BEQ 4$  
1729 006666 006300 ASL R0  
1730 006670 100373 BPL 3$  
1731 006672 000005 4$: RESET  
1732 006674 010314 MOV DRIVE,@TACS ;SELECT DRIVE  
1733 006676 112714 000001 MOVB #WFGIGO,@TACS ;START A WRITE FILE GAP FUNCTION  
1734 006702 104412 WAITREADY ;GO WAIT FOR "READY" TO SET  
1735 006704 005714 TST @TACS ;DID ERROR BIT SET?  
1736 006706 100401 BMI 2$ ;BR IF YES  
1737 006710 104001 ERROR 1 ;"ERROR" BIT (BIT15) FAILED TO SET
```

```
1738 006712 032714 020000 2$: BIT #LEADER,@TACS ;HOW ABOUT CLEAR LEADER BIT  
1739 006716 001001 BNE TST75 ;BR IF SET  
1740 006720 104001 ERROR 1 ;"CLEAR LEADER" NOT SET  
1741 ;*****  
1742 ;*TEST 75 TEST "READY" CLEARS WHEN STARTING A "WRITE" FUNCTION  
1743 ;*****  
1744 TST75: SCOPE  
1745 006724 012767 000005 172234 MOV #5,$TIMES ;DO 5 ITERATIONS  
1746 006732 000005 RESET  
1747 006734 012701 000001 MOV #1,R1 ;SET TIMER  
1748 006740 010314 MOV DRIVE,@TACS ;SELECT DRIVE  
1749 006742 104412 WAITREADY ;WAIT FOR "READY"  
1750 006744 112714 000003 MOVB #WRITE!GO,@TACS ;START A "WRITE"  
1751 006750 032714 000040 1$: BIT #READY,@TACS ;CHECK "READY" BIT  
1752 006754 001403 BEQ TST76 ;BR IF "READY" = 0  
1753 006756 006301 ASL R1 ;IS TIME UP?  
1754 006760 001373 BNE 1$ ;BR IF NO  
1755 006762 104001 ERROR 1 ;"READY" FAILED TO CLEAR  
1756 ;*****  
1757 ;*TEST 76 TEST "READY" CLEARS WHEN STARTING A "READ" FUNCTION  
1758 ;*****  
1759 TST76: SCOPE  
1760 006764 000004 MOV #5,$TIMES ;DO 5 ITERATIONS  
1761 006766 012767 000005 172172 RESET  
1762 006774 000005 MOV #1,R1 ;SET TIMER  
1763 006776 012701 000001 MOV DRIVE,@TACS ;SELECT DRIVE  
1764 007002 010314 WAITREADY ;WAIT FOR "READY"  
1765 007006 112714 000005 MOVB #READ!GO,@TACS ;START A "READ"  
1766 007012 032714 000040 1$: BIT #READY,@TACS ;CHECK "READY" BIT  
1767 007016 001403 BEQ TST77 ;BR IF "READY" = 0  
1768 007020 006301 ASL R1 ;IS TIME UP?  
1769 007022 001373 BNE 1$ ;BR IF NO  
1770 007024 104001 ERROR 1 ;"READY" FAILED TO CLEAR  
1771 ;*****  
1772 ;*TEST 77 TEST "READY" CLEARS WHEN STARTING A "BACK SPACE FILE GAP" FUNCTION  
1773 ;*****  
1774 TST77: SCOPE  
1775 007026 000004 MOV #5,$TIMES ;DO 5 ITERATIONS  
1776 007030 012767 000005 172130 RESET  
1777 007036 000005 MOV #1,R1 ;SET TIMER  
1778 007040 012701 000001 MOV DRIVE,@TACS ;SELECT DRIVE  
1779 007044 010314 WAITREADY ;WAIT FOR "READY"  
1780 007046 104412 MOVB #BSFG!GO,@TACS ;START A "BACK SPACE FILE GAP"  
1781 007050 112714 000007 1$: BIT #READY,@TACS ;CHECK "READY" BIT  
1782 007054 032714 000040 BEQ TST100 ;BR IF "READY" = 0  
1783 007060 001403 ASL R1 ;IS TIME UP?  
1784 007062 006301 BNE 1$ ;BR IF NO  
1785 007064 001373 ERROR 1 ;"READY" FAILED TO CLEAR  
1786 ;*****  
1787 ;*TEST 100 TEST "READY" CLEARS WHEN STARTING A "BACK SPACE BLOCK GAP" FUNCTION  
1788 ;*****  
1789 TST100: SCOPE  
1790 007070 000004 MOV #5,$TIMES ;DO 5 ITERATIONS  
1791 007072 012767 000005 172066 RESET  
1792 007100 000005 MOV #1,R1 ;SET TIMER  
1793 007102 012701 000001 MOV DRIVE,@TACS ;SELECT DRIVE
```

```

1794 007110 104412          WAITREADY          ;WAIT FOR "READY"
1795 007112 112714          MOV      #BSBG!GO,@TACS ;START A "BACK SPACE BLOCK GAP"
1796 007116 032714 000011 1s:  BIT      #READY,@TACS ;CHECK "READY" BIT
1797 007122 001403          BEQ     TST101          ;;BR IF "READY" = 0
1798 007124 006301          ASL     R1              ;IS TIME UP?
1799 007126 001373          BNE     1$             ;BR IF NO
1800 007130 104001          ERROR   1              ;"READY" FAILED TO CLEAR
1801
1802 ;*****
1803 ;*TEST 101 TEST "READY" CLEARS WHEN STARTING A "SPACE FORWARD FILE GAP" FUNCTION
1804 ;*****
1804 007132 000004          TST101: SCOPE
1805 007134 012767 000005 172024 MOV      #5,$TIMES      ;;DO 5 ITERATIONS
1806 007142 000005          RESET
1807 007144 012701 000001          MOV      #1,R1         ;SET TIMER
1808 007150 010314          MOV      DRIVE,@TACS   ;SELECT DRIVE
1809 007152 104412          WAITREADY             ;WAIT FOR "READY"
1810 007154 112714 000013          MOV      #SFFG!GO,@TACS ;START A "SPACE FORWARD FILE GAP"
1811 007160 032714 000040 1s:  BIT      #READY,@TACS ;CHECK "READY" BIT
1812 007164 001403          BEQ     TST102          ;;BR IF "READY" = 0
1813 007166 006301          ASL     R1              ;IS TIME UP?
1814 007170 001373          BNE     1$             ;BR IF NO
1815 007172 104001          ERROR   1              ;"READY" FAILED TO CLEAR
1816
1817 ;*****
1818 ;*TEST 102 TEST "READY" CLEARS WHEN STARTING A "SPACE FORWARD BLOCK GAP" FUNCTION
1819 ;*****
1819 007174 000004          TST102: SCOPE
1820 007176 012767 000005 171762 MOV      #5,$TIMES      ;;DO 5 ITERATIONS
1821 007204 000005          RESET
1822 007206 012701 000001          MOV      #1,R1         ;SET TIMER
1823 007212 010314          MOV      DRIVE,@TACS   ;SELECT DRIVE
1824 007214 104412          WAITREADY             ;WAIT FOR "READY"
1825 007216 112714 000015          MOV      #SFBG!GO,@TACS ;START A "SPACE FORWARD BLOCK GAP"
1826 007222 032714 000040 1s:  BIT      #READY,@TACS ;CHECK "READY" BIT
1827 007226 001403          BEQ     TST103          ;;BR IF "READY" = 0
1828 007230 006301          ASL     R1              ;IS TIME UP?
1829 007232 001373          BNE     1$             ;BR IF NO
1830 007234 104001          ERROR   1              ;"READY" FAILED TO CLEAR
1831
1832 ;*****
1833 ;*TEST 103 TEST FOR "CLEAR LEADER ERROR" FOR "WRITE FUNCTION"
1834 ;*****
1834 007236 000004          TST103: SCOPE
1835 007240 012767 000005 171720 MOV      #5,$TIMES      ;;DO 5 ITERATIONS
1836 007246 012767 007276 171632 MOV      #1$,SLPADR    ;;SET SCOPE LOOP ADDRESS
1837 007254 012767 007370 171706 MOV      #TST104,$ESCAPE ;;ESCAPE TO TEST 104 ON ERROR
1838 007262 000005          RESET
1839 007264 010314          MOV      DRIVE,@TACS   ;SELECT DRIVE
1840 007266 104412          WAITREADY             ;MAKE SURE TA11 IS READY
1841 007270 112714 000017          MOV      #REWIND+GO,@TACS ;LOAD REWIND AND GO
1842 007274 104412          WAITREADY             ;GO WAIT FOR "READY" TO SET
1843 007276 012700 000001 1s:  MOV      #1,R0         ;SET TIMER
1844 007302 010314          MOV      DRIVE,@TACS   ;SELECT DRIVE
1845 007304 112714 000003          MOV      #WRITE!GO,@TACS ;START A WRITE FUNCTION
1846 007310 032714 000040 3s:  BIT      #READY,@TACS ;WAIT ON READY TO CLEAR
1847 007314 001402          BEQ     4$
1848 007316 006300          ASL     R0
1849 007320 100373          BPL     3$
    
```

```

1850 007322 000005          4s:  RESET
1851 007324 010314          MOV      DRIVE,@TACS   ;SELECT DRIVE
1852 007326 112714 000003          MOV      #WRITE!GO,@TACS ;START A WRITE FUNCTION
1853 007332 012700 000001          MOV      #1,R0         ;WAIT A WHILE FOR XFER REQ.
1854 007336 105714          5s:  TSTB   @TACS          ;XFER REQ = 1?
1855 007340 100402          BMI     6$             ;BR IF YES
1856 007342 005200          INC     R0              ;WAITED LONG ENOUGH?
1857 007344 001374          BNE     5$             ;BR IF NO
1858 007346 105715          6s:  TSTB   @TADB          ;CLEAR TRANSFER REQ.
1859 007350 104412          WAITREADY             ;GO WAIT FOR "READY" TO SET
1860 007352 005714          TST     @TACS          ;DID ERROR BIT SET?
1861 007354 100401          BMI     2$             ;BR IF YES
1862 007356 104001          ERROR   1              ;"ERROR" BIT (BIT15) FAILED TO SET
1863 007360 032714 020000 2s:  BIT      #LEADER,@TACS ;HOW ABOUT CLEAR LEADER BIT
1864 007364 001001          BNE     TST104          ;;BR IF SET
1865 007366 104001          ERROR   1              ;"CLEAR LEADER" NOT SET
1866
1867 ;*****
1868 ;*TEST 104 TEST FOR "CLEAR LEADER ERROR" FOR "READ FUNCTION"
1869 ;*****
1869 007370 000004          TST104: SCOPE
1870 007372 012767 000005 171566 MOV      #5,$TIMES      ;;DO 5 ITERATIONS
1871 007400 012767 007430 171500 MOV      #1$,SLPADR    ;;SET SCOPE LOOP ADDRESS
1872 007406 012767 007504 171554 MOV      #TST105,$ESCAPE ;;ESCAPE TO TEST 105 ON ERROR
1873 007414 000005          RESET
1874 007416 010314          MOV      DRIVE,@TACS   ;SELECT DRIVE
1875 007420 104412          WAITREADY             ;MAKE SURE TA11 IS READY
1876 007422 112714 000017          MOV      #REWIND+GO,@TACS ;LOAD REWIND AND GO
1877 007426 104412          WAITREADY             ;GO WAIT FOR "READY" TO SET
1878 007430 012700 000001 1s:  MOV      #1,R0         ;SET TIMER
1879 007434 010314          MOV      DRIVE,@TACS   ;SELECT DRIVE
1880 007436 112714 000005          MOV      #READ!GO,@TACS ;START A READ FUNCTION
1881 007442 032714 000040 3s:  BIT      #READY,@TACS ;WAIT ON READY TO CLEAR
1882 007446 001402          BEQ     4$
1883 007450 006300          ASL     R0
1884 007452 100373          BPL     3$
1885 007454 000005          4s:  RESET
1886 007456 010314          MOV      DRIVE,@TACS   ;SELECT DRIVE
1887 007460 112714 000005          MOV      #READ!GO,@TACS ;START A READ FUNCTION
1888 007464 104412          WAITREADY             ;GO WAIT FOR "READY" TO SET
1889 007466 005714          TST     @TACS          ;DID ERROR BIT SET?
1890 007470 100401          BMI     2$             ;BR IF YES
1891 007472 104001          ERROR   1              ;"ERROR" BIT (BIT15) FAILED TO SET
1892 007474 032714 020000 2s:  BIT      #LEADER,@TACS ;HOW ABOUT CLEAR LEADER BIT
1893 007500 001001          BNE     TST105          ;;BR IF SET
1894 007502 104001          ERROR   1              ;"CLEAR LEADER" NOT SET
1895
1896 ;*****
1897 ;*TEST 105 TEST FOR "CLEAR LEADER ERROR" FOR "BACK SPACE FILE GAP FUNCTION"
1898 ;*****
1898 007504 000004          TST105: SCOPE
1899 007506 012767 000005 171452 MOV      #5,$TIMES      ;;DO 5 ITERATIONS
1900 007514 012767 007544 171364 MOV      #1$,SLPADR    ;;SET SCOPE LOOP ADDRESS
1901 007522 012767 007620 171440 MOV      #TST106,$ESCAPE ;;ESCAPE TO TEST 106 ON ERROR
1902 007530 000005          RESET
1903 007532 010314          MOV      DRIVE,@TACS   ;SELECT DRIVE
1904 007534 104412          WAITREADY             ;MAKE SURE TA11 IS READY
1905 007536 112714 000017          MOV      #REWIND+GO,@TACS ;LOAD REWIND AND GO
    
```

```

1906 007542 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
1907 007544 012700 000001 16:  MOV #1,R0
1908 007550 010314          MOV DRIVE,@TACS
1909 007552 112714 000007  MOVB #BSFG!GO,@TACS ;START A BACK SPACE FILE GAP FUNCTION
1910 007556 032714 000040 3s:  BIT #READY,@TACS ;WAIT ON READY TO CLEAR
1911 007562 001402          BEQ 4$
1912 007564 006300          ASL R0
1913 007566 100373          BPL 3$
1914 007570 000005          4s:  RESET
1915 007572 010314          MOV DRIVE,@TACS ;SELECT DRIVE
1916 007574 112714 000007  MOVB #BSFG!GO,@TACS ;START A BACK SPACE FILE GAP FUNCTION
1917 007600 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
1918 007602 005714          TST @TACS ;DID ERROR BIT SET?
1919 007604 100401          BMI 2$ ;BR IF YES
1920 007606 104001          ERROR 1 ;"ERROR" BIT (BIT15) FAILED TO SET
1921 007610 032714 020000 2s:  BIT #LEADER,@TACS ;HOW ABOUT CLEAR LEADER BIT
1922 007614 001001          BNE TST106 ;;BR IF SET
1923 007616 104001          ERROR 1 ;"CLEAR LEADER" NOT SET
;*****
;*TEST 106 TEST FOR "CLEAR LEADER ERROR" FOR "BACK SPACE BLOCK GAP FUNCTION"
;*****
1926
1927 007620 000004 TST106: SCOPE
1928 007622 012767 000005 171336 MOV #5,$TIMES ;;DO 5 ITERATIONS
1929 007630 012767 007660 171250 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1930 007636 012767 007734 171324 MOV #TST107,$ESCAPE ;;ESCAPE TO TEST 107 ON ERROR
1931 007644 000005          RESET
1932 007646 010314          MOV DRIVE,@TACS ;SELECT DRIVE
1933 007650 104412          WAITREADY          ;MAKE SURE TA11 IS READY
1934 007652 112714 000017  MOVB #REWIND+GO,@TACS ;LOAD REWIND AND GO
1935 007656 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
1936 007660 012700 000001 1s:  MOV #1,R0
1937 007664 010314          MOV DRIVE,@TACS
1938 007666 112714 000011  MOVB #BSRG!GO,@TACS ;START A BACK SPACE BLOCK GAP FUNCTION
1939 007672 032714 000040 3s:  BIT #READY,@TACS ;WAIT ON READY TO CLEAR
1940 007676 001402          BEQ 4$
1941 007700 006300          ASL R0
1942 007702 100373          BPL 3$
1943 007704 000005          4s:  RESET
1944 007706 010314          MOV DRIVE,@TACS ;SELECT DRIVE
1945 007710 112714 000011  MOVB #BSBG!GO,@TACS ;START A BACK SPACE BLOCK GAP FUNCTION
1946 007714 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
1947 007716 005714          TST @TACS ;DID ERROR BIT SET?
1948 007720 100401          BMI 2$ ;BR IF YES
1949 007722 104001          ERROR 1 ;"ERROR" BIT (BIT15) FAILED TO SET
1950 007724 032714 020000 2s:  BIT #LEADER,@TACS ;HOW ABOUT CLEAR LEADER BIT
1951 007730 001001          BNE TST107 ;;BR IF SET
1952 007732 104001          ERROR 1 ;"CLEAR LEADER" NOT SET
;*****
;*TEST 107 TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD FILE GAP FUNCTION"
;*****
1955
1956 007734 000004 TST107: SCOPE
1957 007736 012767 000005 171222 MOV #5,$TIMES ;;DO 5 ITERATIONS
1958 007744 012767 007774 171134 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1959 007752 012767 010050 171210 MOV #TST110,$ESCAPE ;;ESCAPE TO TEST 110 ON ERROR
1960 007760 000005          RESET
1961 007762 010314          MOV DRIVE,@TACS ;SELECT DRIVE
    
```

```

1962 007764 104412          WAITREADY          ;MAKE SURE TA11 IS READY
1963 007766 112714 000017  MOVB #REWIND+GO,@TACS ;LOAD REWIND AND GO
1964 007772 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
1965 007774 012700 000001 1s:  MOV #1,R0
1966 010000 010314          MOV DRIVE,@TACS
1967 010002 112714 000013  MOVB #SFFG!GO,@TACS ;START A SPACE FWD FILE GAP FUNCTION
1968 010006 032714 000040 3s:  BIT #READY,@TACS ;WAIT ON READY TO CLEAR
1969 010012 001402          BEQ 4$
1970 010014 006300          ASL R0
1971 010016 100373          BPL 3$
1972 010020 000005          4s:  RESET
1973 010022 010314          MOV DRIVE,@TACS ;SELECT DRIVE
1974 010024 112714 000013  MOVB #SFFG!GO,@TACS ;START A SPACE FWD FILE GAP FUNCTION
1975 010030 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
1976 010032 005714          TST @TACS ;DID ERROR BIT SET?
1977 010034 100401          BMI 2$ ;BR IF YES
1978 010036 104001          ERROR 1 ;"ERROR" BIT (BIT15) FAILED TO SET
1979 010040 032714 020000 2s:  BIT #LEADER,@TACS ;HOW ABOUT CLEAR LEADER BIT
1980 010044 001001          BNE TST110 ;;BR IF SET
1981 010046 104001          ERROR 1 ;"CLEAR LEADER" NOT SET
;*****
;*TEST 110 TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD BLOCK GAP FUNCTION"
;*****
1984
1985 010050 000004 TST110: SCOPE
1986 010052 012767 000005 171106 MOV #5,$TIMES ;;DO 5 ITERATIONS
1987 010060 012767 010110 171020 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1988 010066 012767 010164 171074 MOV #TST111,$ESCAPE ;;ESCAPE TO TEST 111 ON ERROR
1989 010074 000005          RESET
1990 010076 010314          MOV DRIVE,@TACS ;SELECT DRIVE
1991 010100 104412          WAITREADY          ;MAKE SURE TA11 IS READY
1992 010102 112714 000017  MOVB #REWIND+GO,@TACS ;LOAD REWIND AND GO
1993 010106 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
1994 010110 012700 000001 1s:  MOV #1,R0
1995 010114 010314          MOV DRIVE,@TACS
1996 010116 112714 000015  MOVB #SFBG!GO,@TACS ;START A SPACE FWD BLOCK GAP FUNCTION
1997 010122 032714 000040 3s:  BIT #READY,@TACS ;WAIT ON READY TO CLEAR
1998 010126 001402          BEQ 4$
1999 010130 006300          ASL R0
2000 010132 100373          BPL 3$
2001 010134 000005          4s:  RESET
2002 010136 010314          MOV DRIVE,@TACS ;SELECT DRIVE
2003 010140 112714 000015  MOVB #SFBG!GO,@TACS ;START A SPACE FWD BLOCK GAP FUNCTION
2004 010144 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
2005 010146 005714          TST @TACS ;DID ERROR BIT SET?
2006 010150 100401          BMI 2$ ;BR IF YES
2007 010152 104001          ERROR 1 ;"ERROR" BIT (BIT15) FAILED TO SET
2008 010154 032714 020000 2s:  BIT #LEADER,@TACS ;HOW ABOUT CLEAR LEADER BIT
2009 010160 001001          BNE TST111 ;;BR IF SET
2010 010162 104001          ERROR 1 ;"CLEAR LEADER" NOT SET
;*****
;*TEST 111 TEST "GO" BIT IS WRITE ONLY
;*****
2013
2014 010164 000004 TST111: SCOPE
2015 010166 012767 000005 170772 MOV #5,$TIMES ;;DO 5 ITERATIONS
2016 010174 012767 010206 170704 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
2017 010202 000005          RESET
    
```

2018	010204	010314			MOV	DRIVE,@TACS		;LOAD THE DRIVE
2019	010206	052714	000001	1s:	BIS	#1,@TACS		;SET "GO" BIT
2020	010212	032714	000001		BIT	#1,@TACS		;TRY TO READ IT
2021	010216	001401			BEG	2s		;NO ERROR IF "GO" BIT=0
2022	010220	104001			ERROR	1		;GO BIT WASN'T 0
2023	010222	104412		2s:	WAITREADY			;WAIT ON "READY"

```

2024 ;////////////////////////////////////
2025 ;////////////////////////////////////
2026 ;THE FOLLOWING TESTS CHECKS FOR PROPER OPERATION OF "TRANSFER REQUEST"
2027 ;AND "ILBS" FOR BOTH WRITING AND READING
2028 ;////////////////////////////////////
2029
2030 ;*****
2031 ;*TEST 112 TEST TRANSFER REQUEST & ILBS FOR "WRITE"
2032 ;*****
2033 TST112: SCOPE
2034 MOV #10,,$TIMES ;DO 10. ITERATIONS
2035 MOV #8$,SLPADR ;SET SCOPE LOOP ADDRESS
2036 MOV #TST113,$ESCAPE ;ESCAPE TO TEST 113 ON ERROR
2037 RESET
2038 MOV DRIVE,@TACS ;SELECT DRIVE
2039 WAITREADY ;INSURE DRIVE IS READY
2040 MOV#B #REWIND+GO,@TACS ;GO TO BOT
2041 WAITREADY ;GO WAIT FOR "READY" TO SET
2042 MOV#B #WFG+GO,@TACS ;GET OFF OF CLEAR LEADER
2043 WAITREADY ;GO WAIT FOR "READY" TO SET
2044 RESET
2045 MOV DRIVE,@TACS
2046 MOV#B #WRITE,@TACS ;LOAD "WRITE" FUNCTION
2047 MOV #1,R0
2048 1s: ASL R0 ;KILL A LITTLE TIME
2049 BNE 1s
2050 TSTB @TACS
2051 BPL 2s ;INSURE XFER REQ=0
2052 ERROR 1 ;XFER REQ. SET BEFORE "GO"
2053 2s: INC @TACS ;SET "GO"
2054 MOV #1,R0
2055 3s: ASL R0 ;KILL A LITTLE TIME
2056 BNE 3s
2057 TSTB @TACS
2058 BMI 4s ;CHECK THAT XFER EQ=1
2059 ERROR 1 ;XFER REQ FAILED TO SET
2060 CLR#B @TADB ;KNOCK DOWN XFER REQ.
2061 TSTB @TACS
2062 BPL 5s ;CHECK THAT "XFER REQ" IS CLEAR
2063 ERROR 1 ;XFER REQ FAILED TO CLEAR
2064 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
2065 BIS #ILBS,@TACS ;SET ILBS--WRITE CRC
2066 TSTB @TACS ;DID "XFER REQ" CLEAR?
2067 BPL 6s ;BR IF YES
2068 ERROR 1 ;"ILBS" DIDN'T CLEAR "XFER REQ"
2069 6s: WAITREADY ;GO WAIT FOR "READY" TO SET
2070 TST @TACS ;ANY ERROR?
2071 BPL TST113 ;BR IF NO
2072 ERROR 1 ;AN ERROR OCCURRED
2073 ;*****
2074 ;*TEST 113 TEST "TRANSFER REQ" AND "ILBS" FOR "READ"
2075 ;*****
2076 TST113: SCOPE
2077 MOV #10,,$TIMES ;DO 10. ITERATIONS
2078 MOV #4$,SLPADR ;SET SCOPE LOOP ADDRESS
2079 MOV #TST114,$ESCAPE ;ESCAPE TO TEST 114 ON ERROR

```

```

2080 010420 000005 RESET
2081 010422 010314 MOV DRIVE,@TACS ;SELECT DRIVE
2082 010424 104412 WAITREADY ;CHECK READY
2083 010426 052714 000017 BIS #REWIND+GO,@TACS
2084 010432 104412 WAITREADY ;GO WAIT FOR "READY" TO SET
2085 010434 112714 000001 MOVB #WFG!GO,@TACS ;GET OFF OF "CLEAR LEADER"
2086 010440 104412 WAITREADY ;GO WAIT ON "READY"
2087 010442 112714 000003 MOVB #WRITE!GO,@TACS ;START A WRITE
2088 010446 104413 WAITXFER
2089 010450 112715 000377 MOVB #377,@TADB ;WRITE ON TAPE
2090 010454 104413 WAITXFER
2091 010456 112715 000377 MOVB #377,@TADB ;WRITE ON TAPE
2092 010462 104413 WAITXFER
2093 010464 052714 000020 BIS #ILBS,@TACS
2094 010470 104412 WAITREADY
2095 010472 112714 000017 4s: MOVB #REWIND!GO,@TACS
2096 010476 104412 WAITREADY
2097 010500 112714 000005 MOVB #READ!GO,@TACS ;START A READ
2098 010504 104413 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
2099 010506 005715 TST @TADB ;CLEAR XFER REQ
2100 010510 105714 TSTB @TACS ;DID XFER REQ CLEAR?
2101 010512 100001 BPL 26 ;BR IF YES
2102 010514 104001 ERROR 1 ;XFER REQ FAILED TO CLEAR
2103 010516 104413 2s: WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
2104 010520 052714 000020 BIS #ILBS,@TACS ;START ILBS--CLEARS XFER REQ
2105 010524 105714 TSTB @TACS ;DID XFER REQ CLEAR?
2106 010526 100001 BPL 36 ;BR IF YES
2107 010530 104001 ERROR 1 ;XFER REQ FAILED TO CLEAR
2108 010532 104412 3s: WAITREADY ;GO WAIT FOR "READY" TO SET
2109
2110
2111
2112 010534 000004 TST114: SCOPE
2113 010536 012767 010546 170342 MOV #18,$LPADR ;;SET SCOPE LOOP ADDRESS
2114 010544 000005 RESET
2115 010546 011500 1s: MOV @TADB,R0 ;PUT TADB IN R0
2116 010550 001401 BEQ TST115 ;;CHECK FOR ALL ZEROS
2117 010552 104006 ERROR 6 ;TADB NOT ALL ZEROS
2118
2119
2120
2121 010554 000004 TST115: SCOPE
2122 010556 012715 177777 MOV #-1,@TADB ;LOAD TADB WITH 1'S
2123 010562 011500 MOV @TADB,R0 ;PUT TADB IN R0
2124 010564 001401 BEQ TST116 ;;BR IF IT DIDN'T CHANGE
2125 010566 104006 ERROR 6 ;ERROR
    
```

```

2126
2127
2128
2129 010570 000004 TST116: SCOPE
2130 010572 012767 000012 170366 MOV #10, $TIMES ;;DO 10. ITERATIONS
2131 010600 012767 010634 170300 MOV #18,$LPADR ;;SET SCOPE LOOP ADDRESS
2132 010606 012767 010666 170354 MOV #48,$ESCAPE ;;ESCAPE TO 4s ON ERROR
2133 010614 000005 RESET
2134 010616 010314 MOV DRIVE,@TACS
2135 010620 112714 000017 MOVB #REWIND!GO,@TACS ;POSITION THE TAPE
2136 010624 104412 WAITREADY ;GO WAIT FOR "READY" TO SET
2137 010626 112714 000001 MOVB #WFG!GO,@TACS ;GET OFF OF "CLEAR LEADER"
2138 010632 104412 WAITREADY ;GO WAIT FOR "READY" TO SET
2139 010634 012700 000001 1s: MOV #1,R0 ;INITIALIZE THE PATTERN
2140 010640 112714 000003 MOVB #WRITE!GO,@TACS ;START A WRITE FUNCTION
2141 010644 104413 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
2142 010646 110015 MOVB R0,@TADB ;LOAD TADB WRITE BUFFER
2143 010650 104413 WAITXFER ;WAIT ON "TRANSFER REQUEST"
2144 010652 011501 MOV @TADB,R1 ;READ THE DATA BUFFER
2145 010654 120001 CMPB R0,R1 ;CHECK THE DATA
2146 010656 001401 BEQ 3s ;BR IF DATA IS GOOD
2147 010660 104007 ERFOR 7 ;ERROR--TADB NOT EQUAL R0
2148 010662 106300 ASLB R0 ;NEXT PATTERN
2149 010664 103370 BCC 2s ;BR IF MORE TO DO
2150 010666 152714 4s: BISB #ILBS,@TACS ;WRITE CRC
2151 010672 104412 WAITREADY ;GO WAIT FOR "READY" TO SET
    
```

```

2152 ;*****
2153 ;*TEST 117    FLOAT A "0" THROUGH TADB
2154 ;*****
2155 010674 000004 TST117: SCOPE
2156 010676 012767      MOV    #10, $TIMES    ;;DO 10. ITERATIONS
2157 010704 012767 000012 170262      MOV    #15, $LPADR    ;;SET SCOPE LOOP ADDRESS
2158 010712 012767 010772 170174      MOV    #4$, $ESCAPE  ;;ESCAPE TO 4$ ON ERROR
2159 010720 000005      RESET
2160 010722 010314      MOV    DRIVE, @TACS
2161 010724 112714 000017      MOV    #REWIND!GO, @TACS    ;POSITION THE TAPE
2162 010730 104412      WAITREADY                ;GO WAIT FOR "READY" TO SET
2163 010732 112714 000001      MOV    #WFG!GO, @TACS    ;GET OFF OF "CLEAR LEADER"
2164 010736 104412      WAITREADY                ;GO WAIT FOR "READY" TO SET
2165 010740 012700 177776 1s:      MOV    #-2, R0          ;INITIALIZE THE PATTERN
2166 010744 112714 000003      MOV    #WRITE!GO, @TACS  ;START A WRITE FUNCTION
2167 010750 104413      WAITXFER                ;GO WAIT ON "TRANSFER REQUEST" TO SET
2168 010752 110015 2s:      MOV    R0, @TADB        ;LOAD INTO WRITE BUFFER
2169 010754 104413      WAITXFER                ;WAIT ON "TRANSFER REQUEST"
2170 010756 011501      MOV    @TADB, R1        ;READ THE DATA BUFFER
2171 010760 120001      CMPB   R0, R1          ;CHECK THE DATA
2172 010762 001401      BEQ    3$              ;BR IF DATA IS GOOD
2173 010764 104007      ERROR  7              ;ERROR--TADB NOT EQUAL R0
2174 010766 106300 3s:      ASLB   R0              ;NEXT PATTERN
2175 010770 103770      BCS   2$              ;BR IF MORE TO DO
2176 010772 152714 000020 4s:      BISB   #ILBS, @TACS    ;WRITE CRC
2177 010776 104412      WAITREADY                ;GO WAIT FOR "READY" TO SET
    
```

```

2178 ;*****
2179 ;*TEST 120    COUNT PATTERN TO TADB
2180 ;*****
2181 011000 000004 TST120: SCOPE
2182 011002 012767 000012 170156      MOV    #10, $TIMES    ;;DO 10. ITERATIONS
2183 011010 012767 011044 170070      MOV    #1$, $LPADR    ;;SET SCOPE LOOP ADDRESS
2184 011016 012767 011074 170144      MOV    #4$, $ESCAPE  ;;ESCAPE TO 4$ ON ERROR
2185 011024 000005      RESET
2186 011026 010314      MOV    DRIVE, @TACS
2187 011030 112714 000017      MOV    #REWIND!GO, @TACS    ;POSITION THE TAPE
2188 011034 104412      WAITREADY                ;GO WAIT FOR "READY" TO SET
2189 011036 112714 000001      MOV    #WFG!GO, @TACS    ;GET OFF OF "CLEAR LEADER"
2190 011042 104412      WAITREADY                ;GO WAIT FOR "READY" TO SET
2191 011044 005000 1s:      CLR    R0              ;INITIALIZE THE COUNT PATTERN
2192 011046 112714 000003      MOV    #WRITE!GO, @TACS  ;START A WRITE FUNCTION
2193 011052 104413      WAITXFER                ;GO WAIT ON "TRANSFER REQUEST" TO SET
2194 011054 110015 2s:      MOV    R0, @TADB        ;LOAD TADB WRITE BUFFER
2195 011056 104413      WAITXFER                ;WAIT ON "TRANSFER REQUEST"
2196 011060 011501      MOV    @TADB, R1        ;READ THE DATA BUFFER
2197 011062 120001      CMPB   R0, R1          ;CHECK THE DATA
2198 011064 001401      BEQ    3$              ;BR IF DATA IS GOOD
2199 011066 104005      ERROR  5              ;ERROR--TADB NOT EQUAL R0
2200 011070 105200 3s:      INCB   R0              ;UPDATE THE PATTERN
2201 011072 001370      BNE   2$              ;LOOP IF NON-ZERO
2202 011074 152714 000020 4s:      BISB   #ILBS, @TACS    ;WRITE CRC
2203 011100 104412      WAITREADY                ;GO WAIT FOR "READY" TO SET
    
```

```

2204 ;*****
2205 ;*TEST 121 END OF TEST CODE
2206 ;*****
2207 011102 000004 TST121: SCOPE
2208 011104 012767 000001 170054 MOV #1,STIMES ;;DO 1 ITERATION
2209 011112 000005 RESET
2210 011114 010314 MOV DRIVE,@TACS ;SELECT DRIVE
2211 011116 112714 000017 MOVB #REWIND1GO,@TACS
2212 011122 104412 WAITREADY
2213 .SBTTL END OF PASS ROUTINE
2214
2215 ;*****
2216 ;*INCREMENT THE PASS NUMBER (SPASS)
2217 ;*TYPE "END PASS"
2218 ;*IF THERES A MONITOR GO TO IT
2219 ;*IF THERE ISN'T JUMP TO START
2220 ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
2221 ;*SENDMG CAN BE CHANGED TO 7.
2222
2223 011124 SEOP:
2224 011124 000004 SCOPE
2225 011126 005067 167750 CLR STSTNM ;;ZERO THE TEST NUMBER
2226 011132 005067 170030 CLR STIMES ;;ZERO THE NUMBER OF ITERATIONS
2227 011136 005267 167736 INC SPASS ;;INCREMENT THE PASS NUMBER
2228 011142 042767 100000 167730 BIC #100000,SPASS ;;DON'T ALLOW A NEG. NUMBER
2229 011150 005327 (PC)+ ;;LOOP?
2230 011152 000001 SEOPCT: .WORD 1
2231 011154 003015 BGT $DOAGN ;;YES
2232 011156 012737 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
2233 011160 000001 SENDCT: .WORD 1
2234 011162 011152 SEOPCT
2235 011164 104401 011217 TYPE ,SENDMG ;;TYPE "END PASS"
2236 011170 013700 000042 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
2237 011174 001405 BEQ $DOAGN ;;BRANCH IF NO MONITOR
2238 011176 000005 RESET ;;CLEAR THE WORLD
2239 011200 004710 SENDAD: JSH PC,(R0) ;;GO TO MONITOR
2240 011202 000240 NOP ;;SAVE ROOM
2241 011204 000240 NOP ;;FOR
2242 011206 000240 NOP ;;ACT11
2243 011210 $DOAGN: JMP @(PC)+ ;;RETURN
2244 011210 000137 SFTNAD: .WORD START
2245 011212 002234 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
2246 011214 377 377 000 $ENDMG: .ASCIZ <15><12>/END PASS/
2247 011217 015 042412 042116
2248 011224 050040 051501 000123
    
```

```

2249 .SBTTL SCOPE HANDLER ROUTINE
2250
2251 ;*****
2252 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
2253 ;*AND LOAD THE TEST NUMBER(STSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
2254 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
2255 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2256 ;*SW14=1 LOOP ON TEST
2257 ;*SW11=1 INHIBIT ITERATIONS
2258 ;*SW09=1 LOOP ON ERROR
2259 ;*SW08=1 LOOP ON TEST IN SWR<7:0>
2260 ;*CALL
2261 ;* SCOPE ;;SCOPE=IOT
2262
2263 011232 6SCOPE:
2264 011232 104406 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
2265 011234 032777 040000 167676 BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
2266 011242 001111 BNE SOVER ;;YES IF SW14=1
2267 ;####START OF CODE FOR THE XOR TESTER####
2268 011244 000416 EXTSTR: BR 6S ;;IF PUNNING ON THE "XOR" TESTER CHANGE
2269 ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
2270 011246 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
2271 011252 012737 011272 000004 MOV #S,@ERRVEC ;;SET FOR TIMEOUT
2272 011260 005737 177060 TST @#177060 ;;TIME OUT ON XOR?
2273 011264 012637 000004 (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
2274 011270 000463 BR SSVLAD ;;GO TO THE NEXT TEST
2275 011272 022626 5S: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
2276 011274 012637 000004 MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
2277 011300 000423 BR 7S ;;LOOP ON THE PRESENT TEST
2278 011302 6S:;####END OF CODE FOR THE XOR TESTER####
2279 011302 032777 000400 167630 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
2280 011310 001404 BEQ 2S ;;BR IF NO
2281 011312 127767 167622 167562 CMPB @SWR,STSTNM ;;ON THE RIGHT TEST? SWR<7:0>
2282 011320 001462 BEQ SOVER ;;BR IF YES
2283 011322 105767 167555 2S: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
2284 011326 001421 BEQ 3S ;;BR IF NO
2285 011330 126767 167561 167545 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
2286 011336 101015 BHI 3S ;;BR IF NO
2287 011340 032777 001000 167572 BIT #BIT09,@SWR ;;LOOP ON ERROR?
2288 011346 001404 BEQ 4S ;;BR IF NO
2289 011350 016767 167534 167530 7S: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
2290 011356 000443 BR SOVER
2291 011360 105067 167517 4S: CLR $ERFLG ;;ZERO THE ERROR FLAG
2292 011364 005067 167576 CLR STIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
2293 011370 000415 BR 1S ;;ESCAPE TO THE NEXT TEST
2294 011372 032777 004000 167540 3S: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
2295 011400 001011 1S: BNE 1S ;;BR IF YES
2296 011402 005767 167472 TST SPASS ;;IF FIRST PASS OF PROGRAM
2297 011406 001406 BEQ 1S ;; INHIBIT ITERATIONS
2298 011410 005267 167470 INC $ICNT ;;INCREMENT ITERATION COUNT
2299 011414 026767 167546 167462 CMP STIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
2300 011422 002021 BGE SOVER ;;BR IF MORE ITERATION REQUIRED
2301 011424 012767 000001 167452 1S: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
2302 011432 016767 000044 167526 MOV $MXCNT,STIMES ;;SET NUMBER OF ITERATIONS TO DO
2303 011440 105267 167436 $SVLAD: INCB STSTNM ;;COUNT TEST NUMBERS
2304 011444 011667 167436 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
    
```



```

2305 011450 011667 167434      MOV      (SP), $LPERR      ;;SAVE ERROR LOOP ADDRESS
2306 011454 005067 167510      CLR      $ESCAPE        ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
2307 011460 112767 000001 167427  MOV      #1, $ERMAX      ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2308 011466 016777 167410 167446  SOVER:  MOV      $STNM, @DISPLAY  ;;DISPLAY TEST NUMBER
2309 011474 016716 167406      MOV      $LPADR, (SP)   ;;FUDGE RETURN ADDRESS
2310 011500 000002                RTI                    ;;FIXES PS
2311 011502 003720                $MXCNT: 2000,         ;;MAX. NUMBER OF ITERATIONS
  
```

```

2312                .SBTTL  EPORR HANDLER ROUTINE
2313
2314                ;;*****
2315                ;;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2316                ;;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2317                ;;AND GO TO TYPERR ON ERROR
2318                ;;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2319                ;;*SW15=1  HALT ON ERROR
2320                ;;*SW13=1  INHIBIT ERROR TYPEOUTS
2321                ;;*SW10=1  BELL ON ERROR
2322                ;;*SW09=1  LOOP ON ERROR
2323                ;;CALL
2324                ;;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2325
2326                $ERROR:
2327 011504 104406      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
2328 011506 011437      MOV          ;;SAVE THE STATUS REG.
2329 011512 011537 001162  MOV      @TACS, @$REG0      ;;SAVE THE DATA BUFFER
2330 011516 010037 001124  ERROR1: MOV      @TABD, @$REG1      ;;R0 WILL CONTAIN THE GOOD DATA
2331 011522 010137 001126  MOV      R0, @$GDDAT      ;;R1 WILL CONTAIN THE BAD DATA
2332 011526 105267 167351  7S:      MOV      R1, @$BDDAT
2333 011532 001775      SERFLG      ;;SET THE ERROR FLAG
2334 011534 016777 167342 167400  BEQ      7S          ;;DON'T LET THE FLAG GO TO ZERO
2335 011542 032777 002000 167370  MOV      $STNM, @DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
2336 011550 001402      BIT      $BIT10, @SWR    ;;BELL ON ERROR?
2337 011552 104401 001172      BEQ      1S          ;;NO - SKIP
2338 011556 005267 167330 1S:      INC          ;;RING BELL
2339 011562 011667 167330      SERTL      ;;COUNT THE NUMBER OF ERRORS
2340 011566 162767 000002 167322  MOV      (SP), $ERRPC    ;;GET ADDRESS OF ERROR INSTRUCTION
2341 011574 117767 167316 167312  SUB      #2, $ERRPC
2342 011602 032777 020000 167330  MOV      @SERRPC, $ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
2343 011610 001004      BIT      $BIT13, @SWR    ;;SKIP TYPEOUT IF SET
2344 011612 004767 000006      BNE      20S          ;;SKIP TYPEOUTS
2345 011616 104401 001177      JSR      PC, TYPERR     ;;GO TO USER ERROR ROUTINE
2346 011622                TYPE      $CRLF
2347 011622 005777 167312 20S:
2348 011626 100002 2S:      TST      @SWR          ;;HALT ON ERROR
2349 011630 000000      BPL          ;;SKIP IF CONTINUE
2350 011632 104406      HALT      ;;HALT ON ERROR!
2351 011634 032777 001000 167276 3S:      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
2352 011642 001402      BIT      $BIT09, @SWR    ;;LOOP ON ERROR SWITCH SET?
2353 011644 016716 167240      BEQ      4S          ;;BR IF NO
2354 011650 005767 167314      $LPERR, (SP)      ;;FUDGE RETURN FOR LOOPING
2355 011654 001402      $ESCAPE          ;;CHECK FOR AN ESCAPE ADDRESS
2356 011656 016716 167306      BEQ      5S          ;;BR IF NONE
2357 011662      MOV      $ESCAPE, (SP)  ;;FUDGE RETURN ADDRESS FOR ESCAPE
2358 011662 022737 011200 000042 5S:      CMP      $$ENDAD, @#42   ;;ACT-11 AUTO-ACCEPT?
2359 011670 001001      BNE          ;;BRANCH IF NO
2360 011672 000000      HALT      ;;YES
2361 011674      6S:
2362 011674 000002      RTI          ;;RETURN
2363
  
```

```

2364 ;*****
2365 ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
2366
2367 011676 104401 001177 TYPERR: TYPE , $CRLF ;TYPE A CARRIAGE RETURN & LINE FEED
2368 011702 010046 MOV R0,-(SP) ;SAVE R0
2369 011704 113700 MOV #SITE MB,R0 ;PICKUP THE ITEM INDEX
2370 011710 005300 DEC R0 ;ADJUST THE INDEX
2371 011712 006300 ASL R0 ;SO IT WILL WORK FOR
2372 011714 006300 ASL R0 ;THE ERROR TABLE
2373 011716 006300 ASL R0
2374 011720 062700 001236 ADD #ERRTB,R0 ;FORM THE TABLE POINTER
2375 011724 012067 000002 MOV (R0)+,1S ;PICKUP "ERROR MESSAGE" POINTER
2376 011730 104401 TYPE ;TYPE "ERROR MESSAGE"
2377 011732 000000 1S: 0 ;"ERROR MESSAGE POINTER" GOES HERE
2378 011734 104401 001177 TYPE , $CRLF
2379 011740 012067 000004 MOV (R0)+,2S ;PICKUP "DATA HEADER" POINTER
2380 011744 001404 BEQ 3S ;IF "0" DON'T TYPE
2381 011746 104401 TYPE ;TYPE "DATA HEADER"
2382 011750 000000 2S: 0 ;"DATA HEADER" POINTER GOES HERE
2383 011752 104401 001177 TYPE , $CRLF
2384 011756 012000 3S: MOV (R0)+,R0 ;PICKUP "DATA POINTER"
2385 011760 001004 BNE 5S ;IF THERE IS DATA TO TYPE GO DO IT
2386 011762 012600 4S: MOV (SP)+,R0 ;RESTORE R0
2387 011764 104401 001177 TYPE , $CRLF ;TYPE A CARRIAGE RETURN&LINE FEED
2388 011770 000207 RTS PC ;RETURN
2389 011772 5S: MOV 0(R0)+,-(SP) ;SAVE 0(R0)+ FOR TYPEOUT
2390 011772 013046 ;TYPE DATA
2391 ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2392 011774 104402 TYP0C
2393 011776 005710 TST (R0) ;TERMINATOR?
2394 012000 001770 BEQ 4S ;BR IF YES
2395 012002 104401 012010 TYPE ,6S ;TYPE 2 SPACES
2396 012006 000771 BR 5S ;LOOP
2397 012010 020040 000 6S: .ASCIZ / / ;ASCII STRING OF 2 SPACES
2398 012014 .EVEN

```

```

2399 ;*****
2400 ;ROUTINE TO WAIT ON THE READY BIT
2401
2402 012014 WAIT,ON,READY: CLR WAIT2 ;SETUP MAX. TIME TO WAIT ON "READY"
2403 012014 005067 000044 MOV MAXCNT,HGHTIM
2404 012020 016767 000072 000044 MOV (SP)+,@#SAVPC ;GET THE PC OF THE WAITREADY INSTRUCTION
2405 012026 012637 001202 SUB #2,@#SAVPC
2406 012032 162737 000002 001202 MOV (SP)+,@#SAVPS ;SAVE THE PS
2407 012040 012637 001704 WAIT1: BIT #READY,@TACS ;READY=1?
2408 012044 032714 000040 BNE WAIT3 ;GO ON IF YES
2409 012050 001013 TSTB @TACS ;CHECK TRANSFER REQUEST
2410 012052 105714 BPL WAIT4
2411 012054 100002 ERROR 4 ;"TRANSFER REQUEST" SET WHILE WAITING ON "READY"
2412 012056 104004 BR WAIT3
2413 012060 000407 WAIT4: INC (PC)+ ;COUNT FAST COUNTER
2414 012062 005227 WAIT2: 0
2415 012064 000000 BNE WAIT1 ;GO CHECK "READY" AGAIN
2416 012066 001366 DEC (PC)+ ;COUNT LOOP COUNTER
2417 012070 005327 HGHTIM: 0
2418 012072 000000 BGT WAIT1 ;GO LOOP AGAIN
2419 012074 003363 ERROR 2 ;"READY" FAILED TO SET
2420 012076 104002 WAIT3: MOV @#SAVPS,-(SP) ;GET THE STATUS BACK
2421 012100 013746 001204 MOV @#SAVPC,-(SP) ;GET THE PC
2422 012104 013746 001202 ADD #2,(SP)
2423 012110 062716 000002 RTI
2424 012114 000002 MAXCNT: 0
2425 012116 000000
2426
2427 ;*****
2428 ;ROUTINE TO WAIT ON TRANSFER REQUEST
2429
2430 012120 WAIT,FOR,XFER,REQ: CLR 2S ;SETUP WASTE TIME LOOP
2431 012120 005067 000044 MOV @#MAXCNT,3S
2432 012124 013767 012116 000044 MOV (SP)+,@#SAVPC ;GET THE PC OF THE WAITXFER INSTRUCTION
2433 012132 012637 001202 SUB #2,@#SAVPC
2434 012136 162737 000002 001202 MOV (SP)+,@#SAVPS ;SAVE THE PS
2435 012144 012637 001204 1S: TSTB @TACS ;CHECK XFER REQ
2436 012150 105714 BMI 4S ;EXIT IF SET
2437 012152 100414 BIT #READY,@TACS ;LOOK AT READY
2438 012154 032714 000040 BEQ 5S ;BR IF "READY" ISN'T SET
2439 012160 001402 ERROR 4 ;"READY" SET WHILE WAITING FOR "XFER REQ"
2440 012162 104004 BR 4S
2441 012164 000407 5S: INC (PC)+ ;COUNT
2442 012166 005227 2S: 0
2443 012170 000000 BNE 1S ;BR IF MORE TO DO
2444 012172 001366 DEC (PC)+
2445 012174 005327 3S: 0
2446 012176 000000 BGT 1S
2447 012200 003363 ERROR 3 ;"TRANSFER REQUEST" FAILED TO SET
2448 012202 104003 4S: MOV @#SAVPS,-(SP) ;GET THE STATUS BACK
2449 012204 013746 001204 MOV @#SAVPC,-(SP) ;GET THE PC
2450 012210 013746 001202 ADD #2,(SP)
2451 012214 062716 000002 RTI
2452 012220 000002 ;GO BACK

```

```
2453 ;*****
2454
2455 .SBTTL ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2456
2457 ;CALL
2458 ; JSR PC,##ASKDRV
2459 ; RETURN ;NOTE: R0 AND R1 ARE DESTROYED
2460
2461 012222 104401 015570 ASKDRV: TYPE ,MSGDRV ;<CRLF>"DRIVE(S)? "
2462 012226 005067 166772 CLR DRVKEY
2463 012232 104410 RDLIN ;GO GET A DRIVE
2464 012234 012600 MOV (SP)+,R0 ;SETUP TO CHECK FOR VALID DRIVE(S)
2465 012236 105710 TSTB @R0 ;WAS A DRIVE SELECTED?
2466 012240 001425 BEQ NOTLGL ;BR IF NO
2467 012242 012701 001224 MOV @DRVKEY,R1
2468 012246 122710 000101 LOOP: CMPB #*A,@R0 ;WAS DRIVE "A" SELECTED?
2469 012252 001002 BNE NOTA ;BR IF NO
2470 012254 112021 MOVB (R0)+,(R1)+ ;SET KEY FOR DRIVE "A"
2471 012256 000411 BR NEXT
2472 012260 122710 000102 NOTA: CMPB #*B,@R0 ;WAS DRIVE "B" SELECTED?
2473 012264 001002 BNE NOTB ;BR IF NO
2474 012266 112021 MOVB (R0)+,(R1)+ ;SET KEY FOR DRIVE "B"
2475 012270 000404 BR NEXT
2476 012272 122710 000054 NOTB: CMPB #54,@R0 ;WAS A COMMA TYPED?
2477 012276 001006 BNE NOTLGL ;BR IF NO
2478 012300 105720 TSTB (R0)+ ;DUMP THE COMMA
2479 012302 105710 NEXT: TSTB @R0 ;TERMINATOR?
2480 012304 001406 BEQ EXIT ;BR IF YES
2481 012306 022701 001226 CMP @DRVKEY+2,R1 ;TWO DRIVES SELECTED?
2482 012312 101355 BHI LOOP ;BR IF NO
2483 012314 104401 001176 NOTLGL: TYPE ,SQUES ;ILLEGAL INPUT DETECTED
2484 012320 000740 BR ASKRV ;GO TRY AGAIN
2485 012322 005767 166676 EXIT: TST DRVKEY ;ANY DRIVE SELECTED?
2486 012326 001772 BEQ NOTLGL ;BR IF NO
2487 012330 000207 RTS PC
2488
```

```
2489 ;*****
2490 .SBTTL ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY
2491 ;CALL
2492 ;JSR PC,##ASKADR
2493
2494 012332 010046 ASKADR: MOV R0,-(SP) ;SAVE R0
2495 012334 104401 015603 1S: TYPE ,MSGASK ;"TACS?"
2496 012340 104411 RDOCT ;GET VALUE
2497 012342 012600 MOV (SP)+,R0 ;PICK UP THE OCTAL NUMBER
2498 012344 001423 BEQ 3S ;IF "0" USE OLD VALUES
2499 012346 020027 160000 CMP R0,#160000 ;MAKE SURE IT IS A BUS ADDRESS
2500 012352 103770 BLO 1S
2501 012354 010037 001206 MOV R0,@TACSL ;SAVE TOE TACS
2502 012360 062700 000002 ADD #2,R0 ;STEP TO TADB ADDRESS
2503 012364 010037 001212 MOV R0,@TADBL ;AND SAVE IT
2504 012370 013737 001206 001210 MOV @TACSL,@TACSH ;SETUP TACS UPPER
2505 012376 005237 001210 INC @TACSH ;BYTE POINTER
2506
2507 012402 013737 001212 001214 MOV @TADBL,@TADBH ;SETUP TADB UPPER
2508 012410 005237 001214 INC @TADBH ;BYTE POINTER
2509 012414 104401 015613 3S: TYPE ,MSGVEC ;"VECTOR?"
2510 012420 104411 RDOCT
2511 012422 012600 MOV (SP)+,R0
2512 012424 001411 BEQ 5S
2513 012426 020027 001000 CMP R0,#1000 ;MAKE SUPE ADDRESS IS IN VECTOR AREA
2514 012432 103370 BHS 3S
2515 012434 010037 001216 MOV R0,@TAVEC ;SAVE AS VECTOR ADDRESS
2516 012440 062700 000002 ADD #2,R0
2517 012444 010037 001220 MOV R0,@TAVEC+2
2518 012450 104401 015623 5S: TYPE ,MSGPRI ;ASK FOR PRIORITY
2519 012454 104411 RDOCT
2520 012456 012600 MOV (SP)+,R0
2521 012460 001413 BEQ 6S ;IF "0" USE OLD VALUE
2522 012462 020027 000007 CMP R0,#7 ;MAKE SURE ITS VALID
2523 012466 101370 BHI 5S
2524 012470 000300 SWAB R0 ;PUT INTO HIGH BYTE
2525 012472 006200 ASR R0 ;AND SHIFT
2526 012474 006200 ASR R0 ;INTO PROPER
2527 012476 006200 ASR R0 ;POSITION
2528 012500 042700 177437 BIC #*C<340>,R0 ;SAVE ONLY PRIORITY BITS
2529 012504 010037 001222 MOV R0,@TAPRIO ;STORE IT AWAY
2530 012510 104401 015635 6S: TYPE ,MTACS ;TACS="
2531 012514 016746 166466 MOV TACSL,-(SP) ;SAVE TACSL FOR TYPEOUT
2532 012520 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2533 012522 104401 015643 TYPE ,MTADB ;"TADB="
2534 012526 016746 166460 MOV TADBL,-(SP) ;SAVE TADBL FOR TYPEOUT
2535 012532 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2536 012534 104401 015652 TYPE ,MTAVEC ;"VECTOR="
2537 012540 016746 166452 MOV TAVEC,-(SP) ;SAVE TAVEC FOR TYPEOUT
2538 012544 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2539 012546 104401 015663 TYPE ,MTAPRI ;"PRIORITY="
2540 012552 016746 166444 MOV TAPRIO,-(SP) ;SAVE TAPRIO FOR TYPEOUT
2541 012556 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2542 012560 104401 015676 TYPE ,MSGOK ;"OK?"
2543 012564 104407 RDCHR ;GO READ ONE CHARACTER
2544 012566 012600 MOV (SP)+,R0 ;GET IT
```

```

2545 012570 022700 000015      CMP      #15,R0      ;IS IT "CR"?
2546 012574 001406              BEQ      7S          ;BRANCH IF YES
2547 012576 022700 000131      CMP      #'Y,R0     ;IS IT "Y"?
2548 012602 001403              BEQ      7S          ;IT WAS
2549 012604 104401 001176      TYPE    ,SQUES      ;TYPE "?"
2550 012610 000651              BR       1S          ;AND LET HIM CORRECT THEM
2551 012612 104401 015704      7S:    TYPE    ,MYES   ;TYPE OUT "YES"
2552 012616 012600              MOV      (SP)+,R0   ;RESTORE R0
2553 012620 000207              RTS       PC         ;AND RETURN
    
```

```

2554 ;////////////////////////////////////
2555 ;////////////////////////////////////
2556 ;THE FOLLOWING ROUTINES CAN BE USED TO MAKE ADJUSTMENTS TO THE TU60
2557 ;NOTE: *** BEFORE USING ANY OF THE ROUTINES LOAD AND START AT 214 ***
2558 ;////////////////////////////////////
2559
2560
2561
2562 ;*****
2563 ; WRITE FILE GAPS FROM "BOT" TO "EOT"
2564 ; START AT 220
2565 ; THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND
2566 ; THE "WRITE DELAY MONO".
2567 ;*****
2568 012622 012706 001100 WFGSUB: MOV  #STACK,SP      ;KEEP THE STACK OUT OF THE WAY
2569 012626 013704 001206      MOV  #TACSL,TACS        ;SETUP THE TA11 STATUS AND
2570 012632 013705 001212      MOV  #TADBL,TADB        ;DATA BUFFER REGISTERS
2571 012636 000005              RESET                    ;RESET THE WORLD
2572 012640 012737 012622 001110      MOV  #WFGSUB,##$LPERR    ;SETUP THE LOOP ON ERROR ADDRESS
2573 012646 004737 013626      JSR  PC,##NXTDRV        ;GO SETUP FOR NEXT DRIVE
2574 012652 010314              MOV  DRIVE,@TACS        ;SELECT DRIVE
2575 012654 112714 000017      MOV  #REWIND!GO,@TACS   ;SEND TAPE TO "BOT"
2576 012660 032714 000040 100S:  BIT  #READY,@TACS      ;WAIT ON READY
2577 012664 001775              BEQ  100S
2578 012666 112714 000001 1S:    MOV  #WFG!GO,@TACS      ;WRITE A FILE GAP
2579 012672 104412              WAITREADY              ;WAIT ON READY
2580 012674 032714 020000      BIT  #LEADER,@TACS      ;AT "CLEAR LEADER"?
2581 012700 001772              BEQ  1S                ;BR IF NO
2582 012702 000000              HALT                    ;STOP IF YES
2583 012704 000746              BR   WFGSUB            ;LOOP ON CONT.
2584
2585
2586 ;*****
2587 ; WRITE CONTINUOUS BLOCKS OF DATA
2588 ; START AT 224
2589 ; THE PROGRAM WILL HALT THREE(3) TIMES
2590 ; AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
2591 ; **** IF USING SOFTWARE SWITCH REGISTER
2592 ; PRESS CONTINUE AND
2593 ; PROGRAM WILL TYPE "SWR=XXXXXX NEW="
2594 ; AND ALLOW A NEW VALUE TO BE INPUT
2595 ; HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
2596 ; HALT 2 --- SWR<7:0> = PATTERN DESIRED
2597 ; HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
2598 ; THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"
2599 ;*****
2600 012706 004737 013370 WRTSUB: JSR  PC,##SETBUF    ;GET BLOCK SIZE AND PATTERN
2601 012712 012706 001100 WLOOP:  MOV  #STACK,SP      ;KEEP THE STACK OUT OF THE WAY
2602 012716 013704 001206      MOV  #TACSL,TACS        ;SETUP THE TA11 STATUS AND
2603 012722 013705 001212      MOV  #TADBL,TADB        ;DATA BUFFER REGISTERS
2604 012726 000005              RESET                    ;RESET THE WORLD
2605 012730 012737 012712 001110      MOV  #WLOOP,##$LPERR    ;SETUP THE LOOP ON ERROR ADDRESS
2606 012736 004737 013626      JSR  PC,##NXTDRV        ;GO SETUP FOR NEXT DRIVE
2607 012742 010314              MOV  DRIVE,@TACS        ;SELECT DRIVE
2608 012744 112714 000017      MOV  #REWIND!GO,@TACS   ;SEND TAPE TO "BOT"
2609 012750 032714 000040 100S:  BIT  #READY,@TACS      ;WAIT ON READY
    
```

```
2610 012754 001775 BEQ 100$ ;
2611 012756 004737 013504 1$: JSR PC,@#WRTBLK ;WRITE A BLOCK
2612 012762 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2613 012766 001773 BEQ 1$ ;BR IF NO
2614 012770 000000 HALT ;STOP IF "EOT"
2615 012772 000747 BR WLOOP ;LOOP IF CONT.
2616
2617
2618 ;*****
2619 ; READ CONTINUOUS BLOCKS OF DATA
2620 ;START AT 230
2621 ;THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
2622 ;AND THE "THRESHOLD POT".
2623 ;*****
2624 012774 012706 001100 RDSUB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2625 013000 013704 001206 MOV #TACSL,TACS ;SETUP THE TA11 STATUS AND
2626 013004 013705 001212 MOV #TADBL,TADB ;DATA BUFFER REGISTERS
2627 013010 000005 RESET ;RESET THE WORLD
2628 013012 012737 012774 001110 MOV #RDSUB,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2629 013020 004737 013626 JSR PC,@#NXTDRV ;GO SETUP FOR NEXT DRIVE
2630 013024 010314 MOV DRIVE,@TACS ;SELECT DRIVE
2631 013026 112714 000017 MOV# #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
2632 013032 032714 000040 100$: BIT #READY,@TACS ;WAIT ON READY
2633 013036 001775 BEQ 100$
2634 013040 004737 013546 1$: JSR PC,@#RDBLK ;READ A BLOCK
2635 013044 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2636 013050 001351 BNE RDSUB ;BR IF YES---LOOP
2637 013052 000772 BR 1$
2638
2639
2640 ;*****
2641 ; WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
2642 ;START AT 234
2643 ;THE PROGRAM WILL HALT THREE(3) TIMES
2644 ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
2645 ;**** IF USING SOFTWARE SWITCH REGISTER
2646 ; PRESS CONTINUE AND
2647 ; PROGRAM WILL TYPE "SWR=XXXXXX NEW="
2648 ; AND ALLOW A NEW VALUE TO BE INPUT
2649 ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
2650 ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
2651 ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
2652 ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
2653 ;AND THE "GAP TIME MONO".
2654 ;*****
2655 013054 004737 013370 WGPBLK: JSR PC,@#SETBUF ;GET BLOCK SIZE AND PATTERN
2656 013060 012706 001100 WGBL0P: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2657 013064 013704 001206 MOV #TACSL,TACS ;SETUP THE TA11 STATUS AND
2658 013070 013705 001212 MOV #TADBL,TADB ;DATA BUFFER REGISTERS
2659 013074 000005 RESET ;RESET THE WORLD
2660 013076 012737 013060 001110 MOV #WGBL0P,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2661 013104 004737 013626 JSR PC,@#NXTDRV ;GO SETUP FOR NEXT DRIVE
2662 013110 010314 MOV DRIVE,@TACS ;SELECT DRIVE
2663 013112 112714 000017 MOV# #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
2664 013116 032714 000040 100$: BIT #READY,@TACS ;WAIT ON READY
2665 013122 001775 BEQ 100$
```

```
2666 013124 112714 000001 1$: MOV# #WFG!GO,@TACS ;WRITE A FILE GAP
2667 013130 104412 WAITREADY ;WAIT ON READY
2668 013132 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2669 013136 001005 BNE 2$ ;BR IF YES
2670 013140 004737 013504 JSR PC,@#WRTBLK ;WRITE A BLOCK
2671 013144 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2672 013150 001765 BEQ 1$ ;BR IF NO
2673 013152 000000 2$: HALT ;STOP AT "EOT"
2674 013154 000741 BR WGBL0P ;START OVER ON CONT.
2675
2676
2677 ;*****
2678 ; READ A BLOCK OF DATA AND A FILE GAP
2679 ;START AT 240
2680 ;THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
2681 ;IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
2682 ;AND THE "TAPE BLANK MONO".
2683 ;*****
2684 013156 012706 001100 RGPBLK: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2685 013162 013704 001206 MOV #TACSL,TACS ;SETUP THE TA11 STATUS AND
2686 013166 013705 001212 MOV #TADBL,TADB ;DATA BUFFER REGISTERS
2687 013172 000005 RESET ;RESET THE WORLD
2688 013174 012737 013156 001110 MOV #RGPBLK,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2689 013202 004737 013626 JSR PC,@#NXTDRV ;GO SETUP FOR NEXT DRIVE
2690 013206 010314 MOV DRIVE,@TACS ;SELECT DRIVE
2691 013210 112714 000017 MOV# #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
2692 013214 032714 000040 100$: BIT #READY,@TACS ;WAIT ON READY
2693 013220 001775 BEQ 100$
2694 013222 004737 013546 1$: JSR PC,@#RDBLK ;READ A BLOCK OF DATA
2695 013226 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2696 013232 001351 BNE RGPBLK ;BR IF YES
2697 013234 112714 000015 MOV# #SFBG!GO,@TACS ;GET INTO A FILE GAP
2698 013240 104412 WAITREADY ;
2699 013242 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2700 013246 001343 BNE RGPBLK ;BR IF YES
2701 013250 000764 BR 1$ ;LOOP
2702
2703
2704 ;*****
2705 ; SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
2706 ;START AT 244
2707 ;THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
2708 ;SPACE FOWARD (TAPE BLANK MONO CAN BE ADJUSTED), OR AFTER READ OR
2709 ;WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
2710 ;(SIGNAL MONO CAN BE CHECKED).
2711 ;*****
2712 013252 012706 001100 SFFGSB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2713 013256 013704 001206 MOV #TACSL,TACS ;SETUP THE TA11 STATUS AND
2714 013262 013705 001212 MOV #TADBL,TADB ;DATA BUFFER REGISTERS
2715 013266 000005 RESET ;RESET THE WORLD
2716 013270 012737 013252 001110 MOV #SFFGSB,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2717 013276 004737 013626 JSR PC,@#NXTDRV ;GO SETUP FOR NEXT DRIVE
2718 013302 010314 MOV DRIVE,@TACS ;SELECT DRIVE
2719 013304 112714 000017 MOV# #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
2720 013310 032714 000040 100$: BIT #READY,@TACS ;WAIT ON READY
2721 013314 001775 BEQ 100$
```

```

2722 013316 112714 000013 1s:  MOVB  #SFFGIGO,@TACS      ;SPACE INTO A FILE GAP
2723 013322 104412          WAITREADY          ;WAIT ON READY
2724 013324 032714 020000      BIT  #LEADER,@TACS      ;AT "CLEAR LEADER"?
2725 013330 001772          BEQ  1s              ;BR IF NO
2726 013332 000000          HALT                ;STOP AT "EOT"
2727 013334 000746          BR    SFFGSB         ;LOOP ON CONT.
2728
2729
2730
2731 ;*****
2731 ;   BACK SPACE FILE GAP
2732 ;START AT 250
2733 ;THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
2734 ;*****
2735 013336 000005          BSFGSB: RESET        ;RESET THE WORLD
2736 013340 012737 013336 001110      MOV  #BSFGSB,0##LPERR  ;LOOP ON ERROR ADDRESS
2737 013346 010314          MOV  DRIVE,@TACS      ;SELECT DRIVE
2738 013350 112714 000007      1s:  MOVB  #BSFG!GO,@TACS  ;BACK SPACE A FILE GAP
2739 013354 104412          WAITREADY          ;WAIT ON READY
2740 013356 032714 020000      BIT  #LEADER,@TACS      ;AT "CLEAR LEADER"?
2741 013362 001772          BEQ  1s              ;BR IF NO
2742 013364 000000          HALT                ;STOP AT BOT
2743 013366 000763          BR    BSFGSB         ;START OVER ON CONT.
2744
2745
2746 ;*****
2746 ;   SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
2747 ;*****
2748
2749 013370 005000          SETBUF: CLR  R0
2750 013372 000000          HALT
2751 013374 022767 000176 165536      CMP  #SWREG,SWR        ;OPERATOR PUTS BYTE COUNT IN SWR<7!0>
2752 013402 001001          BNE  20s             ;SOFTWARE SWITCH REG.?
2753 013404 104405          GTSWR                ;NO-LET HIM BE
2754 013406          ;GET NEW VALUE
2755 013406 157700 165526      20s:  BISB  @SWR,R0         ;CONTINUE
2756 013412 001006          BNE  2s              ;PICKUP THE BYTE COUNT
2757 013414 105777 165521      TSTB @SWR+1           ;BR IF NON-ZERO
2758 013420 001402          BEQ  1s              ;CHECK IF GREATER THAN 377
2759 013422 012700 000376      MOV  #376,R0         ;BR IF NO
2760 013426 005200          INC  R0              ;SET FOR MAX ALLOWED
2761 013430 010037 013500      1s:  MOV  R0,@#BLKLM      ;MAKE IT 377 OR 1
2762 013434 005037 013502      CLR  @#PATRN        ;SETUP THE BLOCK LIMIT
2763 013440 000000          HALT
2764 013442 022767 000176 165470      CMP  #SWREG,SWR        ;OPERATOR PUTS PATTERN IN SWR<7!0>
2765 013450 001001          BNE  21s             ;SOFTWARE SWITCH REG.?
2766 013452 104405          GTSWR                ;NO-LET HIM BE
2767 013454          ;GET NEW VALUE
2768 013454 117737 165460 013502      MOVB @SWP,@#PATRN    ;CONTINUE
2769 013462 000000          HALT                ;PICK UP THE PATTERN
2770 013464 022767 000176 165446      CMP  #SWREG,SWR        ;SET OPERATIONAL SWITCHES
2771 013472 001001          BNE  22s             ;SOFTWARE SWITCH REG.?
2772 013474 104405          GTSWR                ;NO-LET HIM BE
2773 013476          ;GET NEW VALUE
2774 013476 000207          RTS  PC              ;CONTINUE
2775 013500 000000          BLKLM: 0
2776 013502 000000          PATRN: 0
2777          ;RETURN
          ;READ AND WRITE BLOCK SIZE
          ;PATTERN TO GO ON THE TAPE
    
```

```

2778
2779 ;*****
2780 ;   WRITE ROUTINE FOR THE MANUAL OPERATIONS
2781 ;*****
2782 013504 013701 013500      WRBLK: MOV  @#BLKLM,R1  ;PICKUP THE BLOCK SIZE
2783 013510 112714 000003      MOVB #WRITE!GO,@TACS  ;START A WRITE
2784 013514 104413          1s:  WAITXFER          ;WAIT ON TRANSFER REQUEST
2785 013516 032714 000040      BIT  #READY,@TACS     ;DID READY SET?
2786 013522 001010          BNE  3s              ;BR IF YES
2787 013524 005301          DEC  R1              ;COUNT THIS REQUEST
2788 013526 002403          BLT  2s              ;BR IF TIME FOR ILBS
2789 013530 113715 013502      MOVB @#PATRN,@TADB   ;PUT DATA ON TAPE
2790 013534 000767          BR  1s              ;LOOP
2791 013536 052714 000020      2s:  BIS  #ILBS,@TACS   ;WRITE CRC AND SHUT DOWN
2792 013542 104412          WAITREADY          ;WAIT ON THE READY FLAG
2793 013544 000207          3s:  RTS  PC
2794
2795
2796 ;*****
2797 ;   READ ROUTINE FOR THE MANUAL OPERATIONS
2798 ;*****
2799 013546 013702 013500      RDBLK: MOV  @#BLKLM,R2  ;PICKUP THE BLOCK SIZE
2800 013552 013700 013502      MOV  @#PATRN,R0       ;USE THIS DATA PATTERN TO COMPARE TO
2801 013556 112714 000005      MOVB #READ!GO,@TACS  ;START A READ
2802 013562 104413          1s:  WAITXFER          ;WAIT ON TRANSFER REQUEST
2803 013564 032714 000040      BIT  #READY,@TACS     ;IS READY SET?
2804 013570 001012          BNE  3s              ;BR IF YES
2805 013572 005302          DEC  R2              ;COUNT THIS REQUEST
2806 013574 002405          BLT  2s              ;BR IF TIME FOR ILBS
2807 013576 011501          MOV  @TADB,R1        ;READ THE DATA BUFFER
2808 013600 120001          CMPB R0,R1           ;CHECK THE DATA
2809 013602 001767          BEQ  1s              ;BR IF OK
2810 013604 104007          ERROR 7             ;BAD DATA
2811 013606 000406          BR  4s              ;GET OUT
2812 013610 052714 000020      2s:  BIS  #ILBS,@TACS   ;READ ILBS
2813 013614 104412          WAITREADY          ;WAIT ON READY
2814 013616 005714          3s:  TST  @TACS         ;CHECK FOR ERROR
2815 013620 100001          BPL  4s              ;BR IF NONE
2816 013622 104001          ERROR 1             ;ERROR OCCURRED
2817 013624 000207          4s:  RTS  PC              ;RETURN
2818
2819
2820 ;*****
2821 ;   ROUTINE TO CHANGE DRIVES
2822 013626 105777 165306      NXTDRV: TSTB @SWR      ;IS SW07 ON A (1)?
2823 013632 100416          BMI  3s              ;BR IF YES
2824 013634 005003          CLR  DRIVE          ;SET DRIVE TO "A"
2825 013636 013701 001230      MOV  @#DRVPNT,R1     ;GET DRIVE POINTER
2826 013642 122127 000101      CMPB (R1)+,#"A"     ;IS IT DRIVE "A"?
2827 013646 001402          BEQ  1s              ;BR IF YES
2828 013650 012703 000400      MOV  #UNIT,DRIVE    ;SET DRIVE TO "B"
2829 013654 105711          1s:  TSTB (R1)         ;LAST DRIVE BEEN SELECTED
2830 013656 001002          BNE  2s              ;BR IF NO
2831 013660 012701 001224      MOV  #DRVKEY,R1     ;RESET DRIVE POINTER
2832 013664 010137 001230      2s:  MOV  R1,@#DRVPNT  ;SAVE DRIVE POINTER FOR NEXT TIME
2833 013670 000207          3s:  RTS  PC              ;GO BACK
    
```

```

2834 ;*****
2835 ;
2836 ;.SBTTL ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2837 ;
2838 ;CALL:
2839 ; MOV #DRVKEY,R0
2840 ; JSR PC,#EXAM ;R1 IS DESTROYED
2841 ; NORMAL RETURN
2842 ; ERROR RETURN
2843
2844 013672 013701 001206 EXAM: MOV @#TACSL,R1 ;PICKUP THE "CONTROL & STATUS" REG. ADR.
2845 013676 005011 CLR (R1) ;DRIVE="A", FUNCTION="WFG"
2846 013700 122710 000101 CMPB #*A,(R0) ;EXAMINE DRIVE "A"?
2847 013704 001402 BEQ 1$ ;BR IF YES
2848 013706 052711 000400 BIS #UNIT,(R1) ;SELECT DRIVE "B"
2849 013712 032711 000040 1$: BIT #READY,(R1) ;WAIT ON READY
2850 013716 001775 BEQ 1$
2851 013720 005711 TST (R1) ;ANY ERROR?
2852 013722 100024 BPL 4$ ;BR IF NO
2853 013724 032711 001000 BIT #OFFLINE,(R1) ;ERROR DUE TO "OFF LINE"?
2854 013730 001017 BNE 3$ ;BR IF YES
2855 013732 032711 010000 BIT #WRTLOCK,(R1) ;ERROR DUE TO "WRITE LOCK"?
2856 013736 001411 BEQ 2$ ;BR IF NO
2857 013740 122777 000201 165172 CMPB #BIT07:BIT00,#SWR ;"READONLY" SELECTED? (RD1PAS)
2858 013746 001412 BEQ 4$ ;BR IF YES
2859 013750 122777 000203 165162 CMPB #BIT07:BIT01:BIT00,#SWR ;(RD2PAS)?
2860 013756 001406 BEQ 4$ ;BR IF YES
2861 013760 000403 BR 3$ ;TAKE THE ERROR EXIT
2862 013762 032711 020000 2$: BIT #LEADER,(R1) ;ERROR DUE TO "CLEAR LEADER"?
2863 013766 001002 BNE 4$ ;BR IF YES
2864 013770 062716 000002 3$: ADD #2,(SP) ;TAKE ERROR RETURN
2865 013774 000207 4$: RTS PC ;RETURN
    
```

```

2866 ;.SBTTL TYPE ROUTINE
2867 ;
2868 ;*****
2869 ;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2870 ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2871 ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2872 ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2873 ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2874 ;*
2875 ;*CALL:
2876 ;*1) USING A TRAP INSTRUCTION
2877 ;* TYPE ,MESADR ;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2878 ;*OR
2879 ;* TYPE
2880 ;* MESADR
2881 ;*
2882
2883 013776 105767 165155 6TYPE: TSTB $TPFLG ;IS THERE A TERMINAL?
2884 014002 100002 BPL 1$ ;BR IF YES
2885 014004 000000 HALT ;HALT HERE IF NO TERMINAL
2886 014006 000407 BR 3$ ;LEAVE
2887 014010 010046 1$: MOV R0,-(SP) ;SAVE R0
2888 014012 017600 000002 MOV @2(SP),R0 ;GET ADDRESS OF ASCIZ STRING
2889 014016 112046 2$: MOVB (R0)+,-(SP) ;PUSH CHARACTER TO BE TYPED ONTO STACK
2890 014020 001005 4$: BNE 4$ ;BR IF IT ISN'T THE TERMINATOR
2891 014022 005726 TST (SP)+ ;IF TERMINATOR POP IT OFF THE STACK
2892 014024 012600 60$: MOV (SP)+,R0 ;RESTORE R0
2893 014026 062716 000002 3$: ADD #2,(SP) ;ADJUST RETURN PC
2894 014032 000002 RTI ;RETURN
2895 014034 122716 000011 4$: CMPB #HT,(SP) ;BRANCH IF <HT>
2896 014040 001430 BEQ 8$
2897 014042 122716 000200 CMPB #CRLF,(SP) ;BRANCH IF NOT <CRLF>
2898 014046 001006 BNE 5$
2899 014050 005726 TST (SP)+ ;POP <CR><LF> EQUIV
2900 014052 104401 TYPE ;TYPE A CR AND LF
2901 014054 001177 SCRLF
2902 014056 105067 000130 CLRBR $CHARCNT ;CLEAR CHARACTER COUNT
2903 014062 000755 BR 2$ ;GET NEXT CHARACTER
2904 014064 004767 000056 5$: JSR PC,$TYPEC ;GO TYPE THIS CHARACTER
2905 014070 126726 165062 6$: CMPB $FILLC,(SP)+ ;IS IT TIME FOR FILLER CHARS.?
2906 014074 001350 BNE 2$ ;IF NO GO GET NEXT CHAR.
2907 014076 016746 165052 MOV $NULL,-(SP) ;GET # OF FILLER CHARS. NEEDED
2908 ;AND THE NULL CHAR.
2909 014102 105366 000001 7$: DECB 1(SP) ;DOES A NULL NEED TO BE TYPED?
2910 014106 002770 BLT 6$ ;BR IF NO--GO POP THE NULL OFF OF STACK
2911 014110 004767 000032 JSR PC,$TYPEC ;GO TYPE A NULL
2912 014114 105367 000072 DECB $CHARCNT ;DO NOT COUNT AS A COUNT
2913 014120 000770 BR 7$ ;LOOP
2914
2915 ;HORIZONTAL TAB PROCESSOR
2916
2917 014122 112716 000040 8$: MOVB #* ,(SP) ;REPLACE TAB WITH SPACE
2918 014126 004767 000014 9$: JSR PC,$TYPEC ;TYPE A SPACE
2919 014132 132767 000007 000052 BITB #7,$CHARCNT ;BRANCH IF NOT AT
2920 014140 001372 BNE 9$ ;TAB STOP
2921 014142 005726 TST (SP)+ ;POP SPACE OFF STACK
    
```

```

2922 014144 000724          BR      2$          ;;GET NEXT CHARACTER
2923 014146 105777 164776  $TYPEC: TSTB  0$TPS          ;;WAIT UNTIL PRINTER IS READY
2924 014152 100375          BPL      $TYPEC
2925 014154 116677 000002 164770  MOVB    2(SP),0$TPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2926 014162 122766 000015 000002  CMPB    $CR,2(SP)   ;;IS CHARACTER A CARRIAGE RETURN?
2927 014170 001003          BNE      1$          ;;BRANCH IF NO
2928 014172 105067 000014          CLRB    $CHARCNT   ;;YES--CLEAR CHARACTER COUNT
2929 014176 000406          BR      $TYPEX     ;;EXIT
2930 014200 122766 000012 000002 1$:  CMPB    $LF,2(SP)   ;;IS CHARACTER A LINE FEED?
2931 014206 001402          BEQ     $TYPEX     ;;BRANCH IF YES
2932 014210 105227          INCB    (PC)+      ;;COUNT THE CHARACTER
2933 014212 000000          $CHARCNT:WORD 0    ;;CHARACTER COUNT STORAGE
2934 014214 000207          $TYPEX: RTS      PC
2935
2936          ,SBTTL READ AN OCTAL NUMBER FROM THE TTY
2937
2938          ;*****
2939          ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2940          ;*CHANGE IT TO BINARY.
2941          ;*CALL:
2942          ;* RDOCT          ;;READ AN OCTAL NUMBER
2943          ;* RETURN HERE  ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2944          ;*              ;;HIGH ORDER BITS ARE IN $HIOCT
2945
2946 014216 011646          $RDOCT: MOV    (SP),-(SP)  ;;PROVIDE SPACE FOR THE
2947 014220 016666 000004 000002  MOV    4(SP),2(SP)      ;;INPUT NUMBER
2948 014226 010046          MOV    R0,-(SP)        ;;PUSH R0 ON STACK
2949 014230 010146          MOV    R1,-(SP)        ;;PUSH R1 ON STACK
2950 014232 010246          MOV    R2,-(SP)        ;;PUSH R2 ON STACK
2951 014234 104410          1$:  RDLIN          ;;READ AN ASCII LINE
2952 014236 012600          MOV    (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
2953 014240 005001          CLR    R1              ;;CLEAR DATA WORD
2954 014242 005002          CLR    R2
2955 014244 112046          2$:  MOVB    (R0)+,-(SP)  ;;PICKUP THIS CHARACTER
2956 014246 001412          BEQ    3$              ;;IF ZERO GET OUT
2957 014250 006301          ASL    R1              ;;*2
2958 014252 006102          ROL    R2
2959 014254 006301          ASL    R1              ;;*4
2960 014256 006102          ROL    R2
2961 014260 006301          ASL    R1              ;;*8
2962 014262 006102          ROL    R2
2963 014264 042716 177770          BIC    #'C7,(SP)      ;;STRIP THE ASCII JUNK
2964 014270 062601          ADD    (SP)+,R1       ;;ADD IN THIS DIGIT
2965 014272 000764          BR     2$              ;;LOOP
2966 014274 005726          3$:  TST    (SP)+        ;;CLEAN TERMINATOR FROM STACK
2967 014276 010166 000012          MOV    R1,12(SP)     ;;SAVE THE RESULT
2968 014302 010267 000010          MOV    R2,$HIOCT
2969 014306 012602          MOV    (SP)+,R2      ;;POP STACK INTO R2
2970 014310 012601          MOV    (SP)+,R1      ;;POP STACK INTO R1
2971 014312 012600          MOV    (SP)+,R0      ;;POP STACK INTO R0
2972 014314 000002          RTI                    ;;RETURN
2973 014316 000000          $HIOCT: WORD 0        ;;HIGH ORDER BITS GO HERE
2974          ,SBTTL TTY INPUT ROUTINE
2975
2976          ;*****
2977          .ENABL LSB
    
```

```

2978
2979          ;*****
2980          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2981          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2982          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2983          ;*WHEN OPERATING IN TTY FLAG MODE.
2984 014320 022767 000176 164612  SCKSWR: CMP    $SWREG,SWR  ;;IS THE SOFT-SWR SELECTED?
2985 014326 001074          BNE     15$          ;;BRANCH IF NO
2986 014330 105777 164610          TSTB   0$TKS        ;;CHAR THERE?
2987 014334 100071          BPL     15$          ;;IF NO, DON'T WAIT AROUND
2988 014336 117746 164604          MOVB   0$TKB,-(SP)  ;;SAVE THE CHAR
2989 014342 042716 177600          BIC    #'C177,(SP)  ;;STRIP-OFF THE ASCII
2990 014346 022726 000007          CMP    #7,(SP)+     ;;IS IT A CONTROL G?
2991 014352 001062          BNE     15$          ;;NO, RETURN TO USER
2992 014354 126727 164554 000001  CMPB   $AUTOB,#1    ;;ARE WE RUNNING IN AUTO-MODE?
2993 014362 001456          BEQ     15$          ;;BRANCH IF YES
2994
2995 014364 104401 015045          TYPE   ,SCNTLG      ;;ECHO THE CONTROL-G (^G)
2996 014370 104401 015052          SGTSWR: TYPE   ,SMSWR  ;;TYPE CURRENT CONTENTS
2997 014374 016746 163576          MOV    SWREG,-(SP)  ;;SAVE SWREG FOR TYPEOUT
2998 014400 104402          TYPOC  ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2999 014402 104401 015063          TYPE   ,SMNEW      ;;PROMPT FOR NEW SWR
3000 014406 005046          19$:  CLR    -(SP)      ;;CLEAR COUNTER
3001 014410 005046          CLR    -(SP)        ;;THE NEW SWR
3002 014412 105777 164526          7$:  TSTB   0$TKS        ;;CHAR THERE?
3003 014416 100375          BPL     7$          ;;IF NOT TRY AGAIN
3004
3005 014420 117746 164522          MOVB   0$TKB,-(SP)  ;;PICK UP CHAR
3006 014424 042716 177600          BIC    #'C177,(SP)  ;;MAKE IT 7-BIT ASCII
3007
3008
3009
3010 014430 021627 000025          9$:  CMP    (SP),#25    ;;IS IT A CONTROL-U?
3011 014434 001005          BNE     10$          ;;BRANCH IF NOT
3012 014436 104401 015040          TYPE   ,SCNTLU     ;;YES, ECHO CONTROL-U (^U)
3013 014442 062706 000006          20$:  ADD    #6,SP     ;;IGNORE PREVIOUS INPUT
3014 014446 000757          BR     19$          ;;LET'S TRY IT AGAIN
3015
3016
3017 014450 021627 000015          10$:  CMP    (SP),#15    ;;IS IT A <CR>?
3018 014454 001022          BNE     16$          ;;BRANCH IF NO
3019 014456 005766 000004          TST    4(SP)        ;;YES, IS IT THE FIRST CHAR?
3020 014462 001403          BEQ     11$          ;;BRANCH IF YES
3021 014464 016677 000002 164446  MOV    2(SP),0$SWR  ;;SAVE NEW SWR
3022 014472 062706 000006          11$:  ADD    #6,SP     ;;CLEAR UP STACK
3023 014476 104401 001177          14$:  TYPE   ,SCRLF     ;;ECHO <CR> AND <LF>
3024 014502 126727 164427 000001  CMPB   $INTAG,#1    ;;RE-ENABLE TTY KBD INTERRUPTS?
3025 014510 001003          BNE     15$          ;;BRANCH IF NOT
3026 014512 012777 000100 164424  MOV    #100,0$TKS   ;;RE-ENABLE TTY KBD INTERRUPTS
3027 014520 000002          15$:  RTI                    ;;RETURN
3028 014522 004767 177420          16$:  JSR    PC,$TYPEC   ;;ECHO CHAR
3029 014526 021627 000000          CMP    (SP),#60    ;;CHAR < 0?
3030 014532 002420          BLT    18$          ;;BRANCH IF YES
3031 014534 021627 000067          CMP    (SP),#67    ;;CHAR > 7?
3032 014540 003015          BGT    18$          ;;BRANCH IF YES
3033 014542 042726 000060          BIC    #60,(SP)+   ;;STRIP-OFF ASCII
    
```



```

3034 014546 005766 000002          TST      2(SP)          ;;IS THIS THE FIRST CHAR
3035 014552 001403                   BEQ      17$          ;;BRANCH IF YES
3036 014554 006316                   ASL      (SP)        ;;NO, SHIFT PRESENT
3037 014556 006316                   ASL      (SP)        ;; CHAR OVER TO MAKE
3038 014560 006316                   ASL      (SP)        ;; ROOM FOR NEW ONE.
3039 014562 005266 000002          17$: INC      2(SP)          ;;KEEP COUNT OF CHAR
3040 014566 056616 177776          BIS      -2(SP),(SP) ;;SET IN NEW CHAR
3041 014572 000707                   BR       7$          ;;GET THE NEXT ONE
3042 014574 104401 001176          18$: TYPE   $QUES     ;;TYPE ?<CR><LF>
3043 014600 000720                   BR       20$          ;;SIMULATE CONTROL-U
3044                                     .DSABL  LSB
3045
3046
3047
3048                                     ;*****
3049                                     ;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3050                                     ;*CALL:
3051                                     ;*      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
3052                                     ;*      RETURN HERE   ;;CHARACTER IS ON THE STACK
3053                                     ;*                                     ;;WITH PARITY BIT STRIPPED OFF
3054
3055 014602 011646                   SRDCHR: MOV    (SP),-(SP) ;;PUSH DOWN THE PC
3056 014604 016666 000004 000002   MOV    4(SP),2(SP)      ;;SAVE THE PS
3057 014612 105777 164326          1$: TSTB   @STKS        ;;WAIT FOR
3058 014616 100375                   BPL     1$            ;;A CHARACTER
3059 014620 117766 164322 000004   MOVB   @STKB,4(SP)     ;;READ THE TTY
3060 014626 042766 177600 000004   BIC    #'C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
3061 014634 026627 000004 000023   CMP    4(SP),#23      ;;IS IT A CONTROL-S?
3062 014642 001013                   BNE     3$            ;;BRANCH IF NO
3063 014644 105777 164274          2$: TSTB   @STKS        ;;WAIT FOR A CHARACTER
3064 014650 100375                   BPL     2$            ;;LOOP UNTIL ITS THERE
3065 014652 117766 164270          MOVB   @STKB,-(SP)     ;;GET CHARACTER
3066 014656 042716 177600          BIC    #'C177,(SP)    ;;MAKE IT 7-BIT ASCII
3067 014662 022627 000021          CMP    (SP)+,#21      ;;IS IT A CONTROL-Q?
3068 014666 001366                   BNE     2$            ;;IF NOT DISCARD IT
3069 014670 000750                   BR      1$            ;;YES, RESUME
3070 014672 026627 000004 000140   3$: CMP    4(SP),#140   ;;IS IT UPPER CASE?
3071 014700 002407                   BLT     4$            ;;BRANCH IF YES
3072 014702 026627 000004 000175   CMP    4(SP),#175    ;;IS IT A SPECIAL CHAR?
3073 014710 003003                   BGT     4$            ;;BRANCH IF YES
3074 014712 042766 000040 000004   BIC    #40,4(SP)     ;;MAKE IT UPPER CASE
3075 014720 000002                   4$: RTI              ;;GO BACK TO USER
3076
3077                                     ;*****
3078                                     ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3079                                     ;*CALL:
3080                                     ;*      RDLIN         ;;INPUT A STRING FROM THE TTY
3081                                     ;*      RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3082                                     ;*                                     ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
3083 014722 010346                   SRDLIN: MOV    R3,-(SP) ;;SAVE R3
3084 014724 012703 015030          1$: MOV    #STTYIN,R3  ;;GET ADDRESS
3085 014730 022703 015040          2$: CMP    #STTYIN+8,,R3 ;;BUFFER FULL?
3086 014734 101405                   BLOS   4$            ;;BR IF YES
3087 014736 104407                   RDCHR   (SP)+,(R3)    ;;GO READ ONE CHARACTER FROM THE TTY
3088 014740 112613                   MOVB   (SP)+,(R3)    ;;GET CHARACTER
3089 014742 122713 000177          10$: CMPB  #177,(R3)   ;;IS IT A RUBOUT

```

```

3090 014746 001003                   BNE     3$            ;;SKIP IF NOT
3091 014750 104401 001176          4$: TYPE   $QUES     ;;TYPE A "?"
3092 014754 000763                   BP      1$            ;;CLEAR THE BUFFER AND LOOP
3093 014756 111367 000044          3$: MOVB   (R3),9$    ;;ECHO THE CHARACTER
3094 014762 104401 015026          TYPE   ,9$
3095 014766 122723 000015          CMPB   #15,(R3)+     ;;CHECK FOR RETURN
3096 014772 001356                   BNE     2$            ;;LOOP IF NOT RETURN
3097 014774 105063 177777          CLR    -1(R3)       ;;CLEAR RETURN (THE 15)
3098 015000 104401 001200          TYPE   ,$LF         ;;TYPE A LINE FEED
3099 015004 012603                   MOV     (SP)+,R3     ;;RESTORE R3
3100 015006 011646                   MOV     (SP),-(SP)   ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3101 015010 016666 000004 000002   MOV     4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
3102 015016 012766 015030 000004   MOV     #STTYIN,4(SP)
3103 015024 000002                   RTI
3104 015026 000000                   9$: .BYTE  0          ;;RETURN
3105 015027 000000                   .BYTE  0          ;;STORAGE FOR ASCII CHAR. TO TYPE
3106 015030 000010                   .BYTE  0          ;;TERMINATOR
3107 015040 052536 005015 000000   STTYIN: .BLKB  8.    ;;RESERVE 8 BYTES FOR TTY INPUT
3108 015045 136 006507 000012 000012 SCNTLG: .ASCIZ /"U/<15><12> ;;CONTROL "U"
3109 015052 005015 053523 020122 SMSWR: .ASCIZ /"G/<15><12> ;;CONTROL "G"
3110 015060 020075 000000
3111 015063 040 047040 053505 SMNEW: .ASCIZ / NEW = /
3112 015070 036440 000040
3113                                     .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
3114
3115                                     ;*****
3116                                     ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3117                                     ;*OCTAL (ASCII) NUMBER AND TYPE IT.
3118                                     ;*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3119                                     ;*CALL:
3120                                     ;*      MOV      NUM,-(SP) ;;NUMBER TO BE TYPED
3121                                     ;*      TYPOS   ;;CALL FOR TYPEOUT
3122                                     ;*      .BYTE   N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3123                                     ;*      .BYTE   M          ;;M=1 OR 0
3124                                     ;*                                     ;;1=TYPE LEADING ZEROS
3125                                     ;*                                     ;;0=SUPPRESS LEADING ZEROS
3126
3127                                     ;*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3128                                     ;*TYPOS OR STYPOC
3129                                     ;*CALL:
3130                                     ;*      MOV      NUM,-(SP) ;;NUMBER TO BE TYPED
3131                                     ;*      TYPON   ;;CALL FOR TYPEOUT
3132
3133                                     ;*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3134                                     ;*CALL:
3135                                     ;*      MOV      NUM,-(SP) ;;NUMBER TO BE TYPED
3136                                     ;*      TYPOC   ;;CALL FOR TYPEOUT
3137
3138 015074 017646 000000 000000   STYPOS: MOV     0(SP),-(SP) ;;PICKUP THE MODE
3139 015100 116667 000001 000211   MOVB   1(SP),0$FILL  ;;LOAD ZERO FILL SWITCH
3140 015106 112667 000207 000000   MOVB   (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
3141 015112 062716 000002 000000   ADD    #2,(SP)      ;;ADJUST RETURN ADDRESS
3142 015116 000406                   BR     STYPON
3143 015120 112767 000001 000171 STYPOC: MOVB   #1,$0FILL ;;SET THE ZERO FILL SWITCH
3144 015126 112767 000006 000165   MOVB   #6,$OMODE+1  ;;SET FOR SIX(6) DIGITS
3145 015134 112767 000005 000154 STYPON: MOVB   #5,$OCNT ;;SET THE ITERATION COUNT

```

```

3146 015142 010346      MOV      R3,-(SP)      ;;SAVE R3
3147 015144 010446      MOV      R4,-(SP)      ;;SAVE R4
3148 015146 010546      MOV      R5,-(SP)      ;;SAVE R5
3149 015150 116704      MOV      R4,$MODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
3150 015154 005404      NEG      R4
3151 015156 062704      ADD      R4,R4         ;;SUBTRACT IT FOR MAX. ALLOWED
3152 015162 110467      MOV      R4,$MODE      ;;SAVE IT FOR USE
3153 015166 116704      MOV      R4,$FILL,R4   ;;GET THE ZERO FILL SWITCH
3154 015172 016605      MOV      R3,12(SP),R5  ;;PICKUP THE INPUT NUMBER
3155 015176 005003      CLR      R3           ;;CLEAR THE OUTPUT WORD
3156 015200 006105      ROL      R5           ;;ROTATE MSB INTO "C"
3157 015202 000404      BR       36           ;;GO DO MSB
3158 015204 006105      ROL      R5           ;;FORM THIS DIGIT
3159 015206 006105      ROL      R5
3160 015210 006105      ROL      R5
3161 015212 010503      MOV      R5,R3
3162 015214 006103      ROL      R3           ;;GET LSB OF THIS DIGIT
3163 015216 105367      DEC      R3           ;;TYPE THIS DIGIT?
3164 015222 100016      BPL      78           ;;BR IF NO
3165 015224 042703      BIC      R3,$177770,R3 ;;GET RID OF JUNK
3166 015230 001002      BNE      48           ;;TEST FOR 0
3167 015232 005704      TST      R4           ;;SUPPRESS THIS 0?
3168 015234 001403      BEQ      56           ;;BR IF YES
3169 015236 005204      INC      R4           ;;DON'T SUPPRESS ANYMORE 0'S
3170 015240 052703      BIS      R3,$'0,R3    ;;MAKE THIS DIGIT ASCII
3171 015244 052703      BIS      R3,$',R3     ;;MAKE ASCII IF NOT ALREADY
3172 015250 110367      MOV      R3,R5        ;;SAVE FOR TYPING
3173 015254 104401      TYPE     R3,$8        ;;GO TYPE THIS DIGIT
3174 015260 105367      DEC      R3           ;;COUNT BY 1
3175 015264 003347      BGT      28           ;;BR IF MORE TO DO
3176 015266 002402      BLT      68           ;;BR IF DONE
3177 015270 005204      INC      R4           ;;INSURE LAST DIGIT ISN'T A BLANK
3178 015272 000744      BR       28           ;;GO DO THE LAST DIGIT
3179 015274 012605      MOV      R5,(SP)+,R5  ;;RESTORE R5
3180 015276 012604      MOV      R4,(SP)+,R4  ;;RESTORE R4
3181 015300 012603      MOV      R3,(SP)+,R3  ;;RESTORE R3
3182 015302 016666      MOV      R3,2(SP),R4  ;;SET THE STACK FOR RETURNING
3183 015310 012616      MOV      R3,(SP)+,R3  ;;
3184 015312 000002      RTI          ;;RETURN
3185 015314 000           .BYTE     0           ;;STORAGE FOR ASCII DIGIT
3186 015315 000           .BYTE     0           ;;TERMINATOR FOR TYPE ROUTINE
3187 015316 000           .BYTE     0           ;;OCTAL DIGIT COUNTER
3188 015317 000           .BYTE     0           ;;ZERO FILL SWITCH
3189 015320 000000      .WORD     0           ;;NUMBER OF DIGITS TO TYPE
    
```

```

3190      .SBTTL TRAP DECODER
3191
3192      ;;*****
3193      ;;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3194      ;;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3195      ;;OF THE DESIRED ROUTINE, THEN USING THE ADDRESS OBTAINED IT WILL
3196      ;;GO TO THAT ROUTINE.
3197
3198 015322 010046      STRAP:  MOV      R0,-(SP)      ;;SAVE R0
3199 015324 016600      MOV      R0,2(SP),R0      ;;GET TRAP ADDRESS
3200 015330 005740      TST      R0               ;;BACKUP BY 2
3201 015332 111000      MOV      R0,(R0),R0      ;;GET RIGHT BYTE OF TRAP
3202 015334 006300      ASL      R0               ;;POSITION FOR INDEXING
3203 015336 016000      MOV      R0,STRPAD(R0),R0 ;;INDEX TO TABLE
3204 015342 000200      RTS          ;;GO TO ROUTINE
3205
3206
3207      ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3208
3209 015344 011646      STRAP2: MOV      (SP),-(SP)   ;;MOVE THE PC DOWN
3210 015346 016666      MOV      4(SP),2(SP)      ;;MOVE THE PSW DOWN
3211 015354 000002      RTI          ;;RESTORE THE PSW
3212
3213      .SBTTL TRAP TABLE
3214
3215      ;;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3216      ;;*BY THE "TRAP" INSTRUCTION.
3217
3218      ;      ROUTINE
3219      ;      -----
3220 015356 015344      STRPAD: .WORD     STRAP2
3221 015360 013776      STYPE   ;;CALL=STYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
3222 015362 015120      STYOC   ;;CALL=TYOC      TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3223 015364 015074      STYPOS  ;;CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
3224 015366 015134      STYPON  ;;CALL=TYPON     TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
3225
3226 015370 014370      $GTSWR  ;;CALL=GTSWR     TRAP+5(104405) GET SOFT-SWR SETTING
3227
3228 015372 014320      $CKSWR  ;;CALL=CKSWR     TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
3229 015374 014602      $RDCHR  ;;CALL=RDCHR     TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
3230 015376 014722      $RDLIN  ;;CALL=RDLIN     TRAP+10(104410) TTY TYPEIN STRING ROUTINE
3231 015400 014216      $RDOCT  ;;CALL=RDOCT     TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
3232 015402 012014      WAIT,ON,READY ;;CALL=WAITREADY     TRAP+12(104412) WAIT ON THE READY BIT TO
3233 015404 012120      WAIT,FOR,XFER ;;CALL=WAITXFER     TRAP+13(104413) WAIT ON XFER REQ.
    
```

```

3234                                ,SBTTL  POWER DOWN AND UP ROUTINES
3235
3236                                ;*****
3237                                ;POWER DOWN ROUTINE
3238 015406 012737 015552 000024 SPWRDN: MOV    #SILLUP,##PWRVEC ;;SET FOR FAST UP
3239 015414 012737 000340 000026 MOV    #340,##PWRVEC+2 ;;PRIO:7
3240 015422 010046 MOV    R0,-(SP) ;;PUSH R0 ON STACK
3241 015424 010146 MOV    R1,-(SP) ;;PUSH R1 ON STACK
3242 015426 010246 MOV    R2,-(SP) ;;PUSH R2 ON STACK
3243 015430 010346 MOV    R3,-(SP) ;;PUSH R3 ON STACK
3244 015432 010446 MOV    R4,-(SP) ;;PUSH R4 ON STACK
3245 015434 010546 MOV    R5,-(SP) ;;PUSH R5 ON STACK
3246 015436 017746 163476 MOV    @SWR,-(SP) ;;PUSH @SWR ON STACK
3247 015442 010667 000110 MOV    SP,$SAVR6 ;;SAVE SP
3248 015446 012737 015460 000024 MOV    #SPWRUP,##PWRVEC ;;SET UP VECTOR
3249 015454 000000 HALT
3250 015456 000776 BR      -2                ;;HANG UP
3251
3252                                ;*****
3253                                ;POWER UP ROUTINE
3254 015460 012737 015552 000024 SPWRUP: MOV    #SILLUP,##PWRVEC ;;SET FOR FAST DOWN
3255 015466 016706 000064 MOV    $SAVR6,SP ;;GET SP
3256 015472 005067 000060 CLR    $SAVR6 ;;WAIT LOOP FOR THE TTY
3257 015476 005267 000054 1$: INC    $SAVR6 ;;WAIT FOR THE INC
3258 015502 001375 BNE    1$ ;;OF WORD
3259 015504 012677 163430 MOV    (SP)+,@SWR ;;POP STACK INTO @SWR
3260 015510 012605 MOV    (SP)+,R5 ;;POP STACK INTO R5
3261 015512 012604 MOV    (SP)+,R4 ;;POP STACK INTO R4
3262 015514 012603 MOV    (SP)+,R3 ;;POP STACK INTO R3
3263 015516 012602 MOV    (SP)+,R2 ;;POP STACK INTO R2
3264 015520 012601 MOV    (SP)+,R1 ;;POP STACK INTO R1
3265 015522 012600 MOV    (SP)+,R0 ;;POP STACK INTO R0
3266 015524 012737 015406 000024 MOV    #SPWRDN,##PWRVEC ;;SET UP THE POWER DOWN VECTOR
3267 015532 012737 000340 000026 MOV    #340,##PWRVEC+2 ;;PRIO:7
3268 015540 104401 TYPE ;;REPORT THE POWER FAILURE
3269 015542 015560 SPWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
3270 015544 012716 MOV    (PC)+,(SP) ;;RESTART AT PWRST
3271 015546 002222 SPWRAD: .WORD PWRST ;;RESTART ADDRESS
3272 015550 000002 RTI
3273 015552 000000 SILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
3274 015554 000776 BR      -2                ;; BEFORE THE POWER DOWN WAS COMPLETE
3275 015556 000000 $SAVR6: 0 ;;PUT THE SP HERE
3276 015560 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
3277 015566 000122
3278                                ,EVEN
    
```

```

3279 015570 042200 044522 042526 MSGDRV: .ASCIZ <CRLF>/DRIVE(S)?/
3280 015576 051450 037451 000 MSGASK: .ASCIZ <15><12>/TACS?/
3281 015603 015 052012 041501 MSGVEC: .ASCIZ /VECTOR?/
3282 015610 037523 000
3283 015613 126 041505 047524 MSGPRI: .ASCIZ /PRIORITY?/
3284 015620 037522 000
3285 015623 120 044522 051117 MSGGOK: .ASCIZ <15><12>/OK?/
3286 015630 052111 037531 000 MYES: .ASCIZ /YES/<15><12>/
3287 015635 124 041501 036523 MTACS: .ASCIZ /TACS=/
3288 015642 000
3289 015643 000 040524 041104 MTADB: .ASCIZ / TADB=/
3290 015650 000075
3291 015652 053040 041505 047524 MTAVEC: .ASCIZ / VECTOR=/
3292 015660 036522 000
3293 015663 040 051120 047511 MTAPRI: .ASCIZ / PRIORITY=/
3294 015670 044522 054524 000075
3295 015676 005015 045517 000077 MSGOK: .ASCIZ <15><12>/OK?/
3296 015704 042531 006523 000012 MYES: .ASCIZ /YES/<15><12>/
3297 015712 052123 052101 051525 EM1: .ASCIZ /STATUS PROBLEM/
3298 015720 050040 047522 046102
3299 015726 046505 000
3300 015731 042 042522 042101 EM2: .ASCIZ /"READY" FAILED TO SET/
3301 015736 021131 043040 044501
3302 015744 042514 020104 047524
3303 015752 051440 052105 000
3304 015757 042 051124 047101 EM3: .ASCIZ /"TRANSFER REQUEST" FAILED TO SET/
3305 015764 043123 051105 051040
3306 015772 050505 042525 052123
3307 016000 020042 040506 046111
3308 016006 042105 052040 020117
3309 016014 042523 000124
3310 016020 044124 020105 051127 EM4: .ASCIZ /THE WRONG FLAG SET/
3311 016026 047117 020107 046106
3312 016034 043501 051440 052105
3313 016042 000
3314 016043 103 052517 052116 EM5: .ASCIZ /COUNT PATTERN FAILED/
3315 016050 050040 052101 042524
3316 016056 047122 043040 044501
3317 016064 042514 000104
3318 016070 040504 040524 050040 EM6: .ASCIZ /DATA PROBLEM/
3319 016076 047522 046102 046505
3320 016104 000
3321 016105 101 042104 042522 EM10: .ASCIZ /ADDRESS FAILED/
3322 016112 051523 043040 044501
3323 016120 042514 000104
3324 016124 041520 020040 020040 DH1: .ASCIZ /PC TACS/
3325 016132 020040 040524 051503
3326 016140 000
3327 016141 120 020103 020040 DH2: .ASCIZ /PC TACS WAIT ADDRESS/
3328 016146 020040 052040 041501
3329 016154 020123 020040 053440
3330 016162 044501 020124 042101
3331 016170 051104 051505 000123
3332 016176 041520 020040 020040 DH5: .ASCIZ /PC TACS EXPECT RCV*D/
3333 016204 020040 040524 051503
3334 016212 020040 020040 054105
    
```

DZTAA,NEW POWER DOWN AND UP ROUTINES

```

3335 016220 042520 052103 020040
3336 016226 041522 023526 000104
3337 016234 041520 020040 020040 DH6: ,ASCIZ /PC TADB/
3338 016242 020040 040524 041104
3339 016250 000
3340 016251 120 020103 020040 DH10: ,ASCIZ /PC ADDRESS/
3341 016256 020040 040440 042104
3342 016264 042522 051523 000
3343 016272 ,EVEN
3344 016272 001116 001162 000000 DT1: ,WORD $ERRPC,$REG0,0
3345 016300 001116 001162 001202 DT2: ,WORD $ERRPC,$REG0,$AVPC,0
3346 016306 000000
3347 016310 001116 001162 001124 DT5: ,WORD $ERRPC,$REG0,$GDDAT,$BDDAT,0
3348 016316 001126 000000
3349 016322 001116 001164 000000 DT6: ,WORD $ERRPC,$REG1,0
3350 016330 001116 001122 000000 DT10: ,WORD $ERRPC,$BDADR,0
3351 016336 001116 001206 000000 DT201: ,WORD $ERRPC,$TACSL,0
3352
3353 016344 001116 000000 DT202: ,WORD $ERRPC,0
3354
3355 016350 040524 030461 043040 EM201: ,ASCIZ "TA11 FAILED TO RESPOND"
3356 016356 044501 042514 020104
3357 016364 047524 051040 051505
3358 016372 047520 042116 000
3359 016377 116 020117 051104 EM202: ,ASCIZ "NO DRIVE AVAILABLE"
3360 016404 053111 020105 053101
3361 016412 044501 040514 046102
3362 016420 000105
3363 016422 041520 020040 020040 DH201: ,ASCIZ /PC TACS/
3364 016430 020040 040524 051503
3365 016436 000
3366 016437 120 000103 DH202: ,ASCIZ /PC/
3367 000001 ,END
    
```

DZTAA,NEW CROSS REFERENCE TABLE -- USER SYMBOLS

```

ASKADR 012332 572 2494#
ASKDRV 012222 569 2461# 2484
ASKKEY 001232 394# 659# 695
BEGINX 001760 566# 602 604 652 654
BEGIN1 001356 231 485#
BEGIN2 001410 232 492#
BEGIN3 001416 233 491 494#
BEGIN4 001424 234 496#
BGNCMN 001430 488 490 493 495 497#
BIT0 = 000001 166#
BIT00 = 000001 156# 166 208 956 1415 2857 2859
BIT01 = 000002 155# 165 207 947 1105 1106 1110 1119 1120 1125 2859
BIT02 = 000004 154# 164 206 938 1078 1079 1083 1084 1092 1093 1098 1402 1404
BIT03 = 000010 153# 163 205 929 1051 1052 1056 1057 1065 1066 1071
BIT04 = 000020 152# 162 204 920 1024 1025 1029 1030 1038 1039 1044
BIT05 = 000040 151# 161 203 1161 1382 1383 1386
BIT06 = 000100 150# 160 202 911 997 998 1002 1003 1011 1012 1017
BIT07 = 000200 149# 159 201 902 1362 1363 1366 2857 2859
BIT08 = 000400 148# 158 200 893 970 971 975 976 984 985 990 1413 1440
      1441 2279
BIT09 = 001000 147# 157 199 1171 1342 1343 1346 2287 2351
BIT1 = 000002 165#
BIT10 = 002000 146# 198 884 1322 1323 1326 2335
BIT11 = 004000 145# 197 875 1302 1303 1306 2294
BIT12 = 010000 144# 196 1182 1282 1283 1286
BIT13 = 020000 143# 195 1192 1262 1263 1266 2342
BIT14 = 040000 142# 194 866 1242 1243 1246 2265
BIT15 = 100000 141# 193 670 1202 1222 1223 1226
BIT2 = 000004 164#
BIT3 = 000010 163#
BIT4 = 000020 162#
BIT5 = 000040 161#
BIT6 = 000100 160#
BIT7 = 000200 159#
BIT8 = 000400 158#
BIT9 = 001000 157#
BLKLIM 013500 2761# 2775# 2782 2799
BPTVEC= 000014 173#
BSBC = 000010 186# 1486 1795 1938 1945
BSFG = 000006 185# 1482 1780 1909 1916 2738
BSFGSB 013336 241 2735# 2736 2743
CHKADR 002002 568 571 590#
CHKDRV 002062 628#
CKSWR = 104406 2264 2327 2350 3228#
CR = 000015 81# 2926 2936
CRCERR= 040000 194#
CRLF = 000200 82# 554 2897 2936 3279
CURDRV 001234 395# 688# 689
DDISP = 177570 88# 364 525
DH1 016124 416 3324#
DH10 016251 458 3340#
DH2 016141 422 428 434 3327#
DH201 016422 465 3363#
DH202 016437 470 3366#
DH5 016176 440 452 3332#
DH6 016234 446 3337#
    
```







