

TA11

MANUAL INTERVENTION TEST
MD-11-DZTAC-C

EP DZTAC C DL
COPYRIGHT 1976
FICHE 1 OF 1

JAN 1978
digital
MADE IN USA

The microfiche card contains 54 frames of data, arranged in a grid of 9 rows and 6 columns. The last row contains only 4 frames. Each frame displays a different screen of data from a manual intervention test, showing various alphanumeric characters, numbers, and symbols. The data is presented in a structured, tabular format, likely representing test results or system status information.

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZTAC-C-D
PRODUCT NAME: TA11 MANUAL INTERVENTION TEST
PRODUCT DATE: JULY, 1976
MAINTAINER: DIAGNOSTIC ENGINEERING

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 MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITALS COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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, DIGITAL EQUIPMENT 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

MAINDEC-11-DZTAA
MAINDEC-11-DZTAB

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 TYPE INSTRUCTIONS ON THE TTY FOR THE OPERATOR TO FOLLOW. AFTER THE OPERATOR HAS PERFORMED THE REQUIRED OPERATION HE WILL PRESS "CARRIAGE RETURN" AND THE PROGRAM WILL CONTINUE.

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

4.3.1 DRIVE SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECTION OF DRIVES "A" AND "B" TO BE TESTED.

NOTE: IF LOAD MEDIUM IS CASSETTE WITH STANDARD VECTOR PROGRAM WILL RESPOND AS IF STARTED AT 210.

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

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

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

4.3.1.1 DRIVE SELECTION EXAMPLES

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

4.3.2 ADDRESS SELECTION

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

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

1. BUS ADDRESS OF THE CONTROL AND STATUS REGISTER (TACS)
 2. VECTOR ADDRESS
 3. PRIORITY LEVEL
- AND THE OPERATOR MUST RESPOND WITH THE DESIRED PARAMETER OR A CARRIAGE RETURN (WHICH IMPLIES LEAVE AS IS). WHEN ALL PARAMETERS HAVE BEEN DEFINED THE PROGRAM WILL TYPE THEM BACK OUT AND ASK IF THEY ARE OK AT WHICH TIME THE OPERATOR RESPONDES WITH A "Y" OR A "CARRIAGE RETURN" FOR "YES" ANYTHING ELSE IS A "NO".

4.3.2.1 ADDRESS SELECTION EXAMPLES

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

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

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.

5.2.2 THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS TRAPCATCHER

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

5.2.3 ERROR

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

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

5.2.4 TRAP

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

5.2.5 THE FOLLOWING SUBROUTINES ARE CALLED BY A JSR

5.2.5.1 STYPEC

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 ASKADR

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

5.2.5.5 ASKQUES

THIS ROUTINE WILL TYPE DIRECTIONS ON THE TTY AND WAITS
FOR A RESPONSE OF A "Y", "N" OR "CARRIAGE RETURN."

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 PROMPTED FOR THE VALUE WITH "SWR=XXXXXX 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 PROMPTED FOR THE VALUE WITH "SWR=XXXXXX 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.

8. MISCELLANEOUS**8.1 EXECUTION TIME**

IS OPERATOR DEPENDENT BUT SHOULD TAKE NO MORE THAN 2 MINUTES.

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100.

8.3 PASS COUNT

A PROGRAM PASS THRU COUNT IS KEPT IN "SPASS".

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

R3, R4 AND R5 ARE RESERVED THROUGH OUT THE PROGRAM FOR "DRIVE", "TACS" AND "TADB"

9. PROGRAM DESCRIPTION

THIS PROGRAM IS A SEQUENCE OF SMALL TESTS THAT CHECK THE TAI1 FOR PROPER OPERATION.

THE TESTS CAN BE GROUPED INTO THE FOLLOWING GENERAL GROUPS.

1. TEST "OFFLINE" WITH TU60 POWER OFF
2. TEST "OFFLINE" WITH CASSETTE REMOVED
3. TEST "WRITE LOCK" CIRCUITRY

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
405	ERROR POINTER TABLE
466	START OF TEST
491	INITIALIZE THE COMMON TAGS
529	TYPE PROGRAM NAME
536	GET VALUE FOR SOFTWARE SWITCH REGISTER
702	T1 ROUTINE TO DETERMINE TIME OF WAIT LOOPS
720	*****MANUAL INTERVENTION*****
721	T2 SETUP FOR MANUAL INTERVENTION
737	T3 SETUP FOR POWER DOWN TU60 TEST
753	T4 TEST "OFFLINE" WHEN TU60 IS POWERED DOWN
798	T5 POWER UP THE TU60
813	T6 TEST "OFFLINE" WHEN DRIVE IS EMPTY
829	T7 PUT DRIVE "ONLINE"
844	T10 PUT DATA ON TAPE FOR WRITE LOCK TEST
896	T11 TEST "WRITE LOCK" WHEN AT CLEAR LEADER
1000	T12 TEST "WRITE LOCK" WITH "CRC ERROR"
1101	T13 TEST "WRITE LOCK" WITHOUT ANY ERRORS
1194	T14 TEST "WRITE LOCK" WITH FILE GAP=1
1295	T15 TRY WRITING ON TAPE WHEN "WRITE LOCKED"
1341	T16 TRY "WFG" WHEN WRITE LOCKED
1372	T17 TEST "WFG" ON CLEAR LEADER AND WRITE LOCKED
1393	T20 TEST "WRITE" WITH WRITE LOCK ON AND AT CLEAR LEADER
1412	T21 TEST "WRITE ENABLE"
1430	T22 END OF TEST CODE
1440	END OF PASS ROUTINE
1476	SCOPE HANDLER ROUTINE
1540	ERROR HANDLER ROUTINE
1592	ERROR TIMEOUT ROUTINE
1628	ROUTINE TO WAIT ON THE READY BIT TO SET
1657	ROUTINE TO WAIT ON TRANSFER REQUEST
1686	ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
1721	ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY
1786	TYPE DIRECTIONS TO OPERATOR AND WAIT FOR RESPONSE
1826	***** MANUAL ADJUSTMENT ROUTINES *****
1836	WRITE FILE GAPS FROM "BOT" TO "EOT"
1861	WRITE CONTINUOUS BLOCKS OF DATA
1893	READ CONTINUOUS BLOCKS OF DATA
1916	WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
1953	READ A BLOCK OF DATA AND A FILE GAP
1981	SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
2008	BACK SPACE FILE GAP
2025	SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
2059	WRITE ROUTINE FOR THE MANUAL OPERATIONS
2077	READ ROUTINE FOR THE MANUAL OPERATIONS

2102	ROUTINE TO CHANGE DRIVES
2118	ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABLITY
2148	TYPE ROUTINE
2218	TTY INPUT ROUTINE
2357	READ AN OCTAL NUMBER FROM THE TTY
2395	BINARY TO OCTAL (ASCII) AND TYPE
2472	TRAP DECODER
2495	TRAP TABLE
2516	POWER DOWN AND UP ROUTINES

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```
.TITLE TA11 MANUAL INTERVENTION TEST MAINDEC-11-DZTAC-C
;*COPYRIGHT (C) 1973,1976
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY JIM LACEY
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C1),MAR 24, 1976.
;*
;*****
;*****
;*****
.REM!
```

GENERAL INFORMATION ABOUT THE TA11/TU60 CASSETTE

ADDRESS MNEMONIC DESCRIPTION

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

TACS REGISTER DESCRIPTION

```
-----
```

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

```

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 ENT,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 CRLP= 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= 00 ;*GENERAL REGISTER
92 000001 R1= 01 ;*GENERAL REGISTER
93 000002 R2= 02 ;*GENERAL REGISTER
94 000003 R3= 03 ;*GENERAL REGISTER
95 000004 R4= 04 ;*GENERAL REGISTER
96 000005 R5= 05 ;*GENERAL REGISTER
97 000006 R6= 06 ;*GENERAL REGISTER
98 000007 R7= 07 ;*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
 114 040000
 115 020000
 116 010000
 117 004000
 118 002000
 119 001000
 120 000400
 121 000200
 122 000100
 123 000040
 124 000020
 125 000010
 126 000004
 127 000002
 128 000001
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141 100000
 142 040000
 143 020000
 144 010000
 145 004000
 146 002000
 147 001000
 148 000400
 149 000200
 150 000100
 151 000040
 152 000020
 153 000010
 154 000004
 155 000002
 156 000001
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168

SW15= 100000
 SW14= 40000
 SW13= 20000
 SW12= 10000
 SW11= 4000
 SW10= 2000
 SW09= 1000
 SW08= 400
 SW07= 200
 SW06= 100
 SW05= 40
 SW04= 20
 SW03= 10
 SW02= 4
 SW01= 2
 SW00= 1
 .EQUIV SW09,SW9
 .EQUIV SW08,SW8
 .EQUIV SW07,SW7
 .EQUIV SW06,SW6
 .EQUIV SW05,SW5
 .EQUIV SW04,SW4
 .EQUIV SW03,SW3
 .EQUIV SW02,SW2
 .EQUIV SW01,SW1
 .EQUIV SW00,SW0

;DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
 BIT14= 40000
 BIT13= 20000
 BIT12= 10000
 BIT11= 4000
 BIT10= 2000
 BIT09= 1000
 BIT08= 400
 BIT07= 200
 BIT06= 100
 BIT05= 40
 BIT04= 20
 BIT03= 10
 BIT02= 4
 BIT01= 2
 BIT00= 1
 .EQUIV BIT09,BIT9
 .EQUIV BIT08,BIT8
 .EQUIV BIT07,BIT7
 .EQUIV BIT06,BIT6
 .EQUIV BIT05,BIT5
 .EQUIV BIT04,BIT4
 .EQUIV BIT03,BIT3
 .EQUIV BIT02,BIT2
 .EQUIV BIT01,BIT1
 .EQUIV BIT00,BIT0

;BASIC "CPU" TRAP VECTOR ADDRESSES

C2

169	000004	ERRVEC= 4
170	000010	RESVEC= 10
171	000014	TBITVEC=14
172	000014	TRTVEC= 14
173	000014	BPTVEC= 14
174	000020	IOTVEC= 20
175	000024	PWRVEC= 24
176	000030	EMTVEC= 30
177	000034	TRAPVEC=34
178	000060	TKVEC= 60
179	000064	TPVEC= 64
180	000240	PIROVEC=240

;;TIME OUT AND OTHER ERRORS
;;RESERVED AND ILLEGAL INSTRUCTIONS
;;"T" BIT
;;TRACE TRAP
;;BREAKPOINT TRAP (BPT)
;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
;;POWER FAIL
;;EMULATOR TRAP (EMT) **ERROR**
;;"TRAP" TRAP
;;TTY KEYBOARD VECTOR
;;TTY PRINTER VECTOR
;;PROGRAM INTERRUPT REQUEST VECTOR

```
181 ;TA11 FUNCTIONS
182 WFG= 0 ;WRITE FILE GAP FUNCTION
183 WRITE= 2 ;WRITE FUNCTION
184 READ= 4 ;READ FUNCTION
185 BSFG= 6 ;BACK SPACE FILE GAP FUNCTION
186 BSBG= 10 ;BACK SPACE BLOCK GAP FUNCTION
187 SFFG= 12 ;SPACE FWD FILE GAP FUNCTION
188 SFBG= 14 ;SPACE FWD BLOCK GAP FUNCTION
189 REWIND= 16 ;REWIND FUNCTION
190 ;
191
192 ;TA11 BIT ASSIGNMENT
193 ERROR= BIT15
194 CRCERR= BIT14
195 LEADER= BIT13
196 WRTLOCK=BIT12
197 FGAP= BIT11
198 TINERR= BIT10
199 OFFLINE=BIT09
200 UNIT= BIT08
201 TR_REQ= BIT07
202 INT_EN= BIT06
203 READY= BIT05
204 ILBS= BIT04
205 FUNC2= BIT03
206 FUNC1= BIT02
207 FUNC0= BIT01
208 GO= BIT00
209 FUNCTION= FUNC2+FUNC1+FUNC0
210 ;
211 ;
212 ;
213
214 ;SPECIAL REGISTERS
215 DRIVE= 03 ;R3 CONTAINS THE DRIVE UNDER TEST
216 TACS= 04 ;R4 IS USED AS A POINTER TO THE TACS REGISTER
217 TADB= 05 ;R5 IS USED AS A POINTER TO THE TADB REGISTER.
218
219 ;
220 ;
```

```

221          .SBTTL TRAP CATCHER
222
223          000000
224          .=0
225          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
226          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
227          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
228          000174 000174
229          000176 000000
230          .SBTTL STARTING ADDRESS(ES)
231          000200 000137 001326      JMP      00BEGIN1      ;;JUMP TO STARTING ADDRESS OF PROGRAM
232          000204 000137 001360      JMP      00BEGIN2      ;SELECT DRIVE(S) BEFORE STARTING TEST
233          000210 000137 001366      JMP      00BEGIN3      ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
234          000214 000137 001730      JMP      00BEGINX     ;SETUP FOR MANUAL LOOPING
235          000220 000137 007604      JMP      00MFGSUB     ;WRITE FILE GAP FROM BOT TO EOT
236          000224 000137 007670      JMP      00WRTSUB    ;WRITE CONTINUOUS BLOCKS OF DATA
237          000230 000137 007756      JMP      00RDSUB     ;READ CONTINUOUS BLOCKS OF DATA
238          000234 000137 010036      JMP      00NGPBLK    ;WRITE FILE GAP AND A BLOCK OF DATA
239          000240 000137 010140      JMP      00RGPBLK    ;READ BLOCK OF DATA AND INTO A FILE GAP
240          000244 000137 010234      JMP      00SFFGSB    ;SPACE FWD FILE GAP FROM BOT TO EOT
241          000250 000137 010320      JMP      00BSFGSB    ;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.
;/;;;/

.REN 1

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 THE LOCATION SWR (=1140) MUST CONTAIN ADDRESS "SWREG" (=176).
;;; PLACE VALUE INTO 176 ;;;
;; REGISTERS 3, 4, AND 5 ;;
;; 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
; ;;

000500
000500 017714 000434
000504 000775

.=500
LOOP1: MOV @SWR,@TACS ;LOAD TACS
BR LOOP1 ;LOOP

```
295 ;;.....
296
297 ;WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
298 ;;.....
299      000600      .=600
300
301 000600 000005 LOOP2: RESET ;CLEAR ALL FLAGS
302 000602 010314 MOV DRIVE,@TACS ;SELECT DRIVE
303 000604 112714 000017 MOVB @REWIND!GO,@TACS ;START A REWIND
304 000610 032714 000040 18: BIT @READY,@TACS ;WAIT ON REWIND TO FINISH
305 000614 001775 BEQ 18
306 000616 112714 000003 MOVB @WRITE!GO,@TACS ;START A WRITE
307 000622 105714 28: TSTB @TACS ;CHECK FOR TRANSFER REQUEST
308 000624 100003 BPL 38 ;BR IF NOT SET
309 000626 017715 000306 MOV @SWR,@TADB ;SEND DATA TO TA11
310 000632 000773 BR 28 ;LOOP
311 000634 032714 000040 38: BIT @READY,@TACS ;DID READY SET?
312 000640 001357 BNE LOOP2 ;START OVER IF YES
313 000642 000767 BR 28 ;LOOP
314
315
316 ;;.....
317
318 ;READ FROM BOT TO EOT
319 ;;.....
320
321      000700      .=700
322
323 000700 000005 LOOP3: RESET ;CLEAR ALL FLAGS
324 000702 010314 MOV DRIVE,@TACS ;SELECT DRIVE
325 000704 112714 000017 MOVB @REWIND!GO,@TACS ;START A REWIND
326 000710 032714 000040 18: BIT @READY,@TACS ;WAIT ON REWIND TO FINISH
327 000714 001775 BEQ 18
328 000716 112714 000005 MOVB @READ!GO,@TACS ;START A READ
329 000722 105714 28: TSTB @TACS ;CHECK TRANSFER REQ
330 000724 100002 BPL 38 ;BR IF NOT SET
331 000726 011500 MOV @TADB,R0 ;PICKUP THE DATA
332 000730 000774 BR 28 ;LOOP
333 000732 032714 000040 38: BIT @READY,@TACS ;CHECK READY
334 000736 001360 BNE LOOP3 ;START OVER
335 000740 000770 BR 28 ;LOOP
```

```

336          .SBTTL COMMON TAGS
337
338          ;*****
339          ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
340          ;*USED IN THE PROGRAM.
341
342          001100          .=1100
343 001100          000000          SCHTAG:          .; START OF COMMON TAGS
344 001100          000000          SPASS:          .WORD          0          ;CONTAINS PASS COUNT
345 001102          000          STSTNM:         .BYTE          0          ;CONTAINS THE TEST NUMBER
346 001103          000          SERFLG:         .BYTE          0          ;CONTAINS ERROR FLAG
347 001104          000000          SIGHT:          .WORD          0          ;CONTAINS SUBTEST ITERATION COUNT
348 001106          000000          SLPADR:         .WORD          0          ;CONTAINS SCOPE LOOP ADDRESS
349 001110          000000          SLPERR:         .WORD          0          ;CONTAINS SCOPE RETURN FOR ERRORS
350 001112          000000          SERTTL:         .WORD          0          ;CONTAINS TOTAL ERRORS DETECTED
351 001114          000          SITENB:         .BYTE          0          ;CONTAINS ITEM CONTROL BYTE
352 001115          001          SERMAX:         .BYTE          1          ;CONTAINS MAX. ERRORS PER TEST
353 001116          000000          SERRPC:         .WORD          0          ;CONTAINS PC OF LAST ERROR INSTRUCTION
354 001120          000000          SGADR:          .WORD          0          ;CONTAINS ADDRESS OF "GOOD" DATA
355 001122          000000          SBDADR:         .WORD          0          ;CONTAINS ADDRESS OF "BAD" DATA
356 001124          000000          SGDDAT:         .WORD          0          ;CONTAINS "GOOD" DATA
357 001126          000000          SBDDAT:         .WORD          0          ;CONTAINS "BAD" DATA
358 001130          000000          .WORD          0          ;RESERVED--NOT TO BE USED
359 001132          000000          .WORD          0
360 001134          000          SAUTOB:         .BYTE          0          ;AUTOMATIC MODE INDICATOR
361 001135          000          SINTAG:         .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          STKS:          177560          ;TTY KBD STATUS
366 001146          177562          STKB:          177562          ;TTY KBD BUFFER
367 001150          177564          STPS:          177564          ;TTY PRINTER STATUS REG. ADDRESS
368 001152          177566          STPB:          177566          ;TTY PRINTER BUFFER REG. ADDRESS
369 001154          000          SNULL:         .BYTE          0          ;CONTAINS NULL CHARACTER FOR FILLS
370 001155          002          SFILLS:        .BYTE          2          ;CONTAINS # OF FILLER CHARACTERS REQUIRED
371 001156          012          SFILLC:        .BYTE          12         ;INSERT FILL CHARS. AFTER A "LINE FEED"
372 001157          000          STPFLG:        .BYTE          0          ;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
373 001160          000000          SREGAD:        .WORD          0          ;CONTAINS THE ADDRESS FROM
374          ;WHICH (SREG0) WAS OBTAINED
375 001162          000000          SREG0:         .WORD          0          ;CONTAINS ((SREGAD)+0)
376 001164          000000          SREG1:         .WORD          0          ;CONTAINS ((SREGAD)+2)
377 001166          000000          STINES:        0          ;MAX. NUMBER OF ITERATIONS
378 001170          000000          SESCAPE:       0          ;ESCAPE ON ERROR ADDRESS
379 001172          177607          000377          SBELL:         .ASCIZ          <207><377><377> ;CODE FOR BELL
380 001176          077          SQUES:         .ASCII          /?/          ;QUESTION MARK
381 001177          015          SCRLF:         .ASCII          <15>          ;CARRIAGE RETURN
382 001200          000012          SLF:          .ASCIZ          <12>          ;LINE FEED
383          ;*****
384 001202          000000          SAVPC:         .WORD          0          ;STORAGE FOR THE PC
385 001204          000000          SAVPS:         .WORD          0          ;STORAGE FOR THE PS
386
387 001206          177500          TACSL:         177500          ;LOW BYTE ADDRESS OF TACS
388 001210          177501          TACSH:         177501          ;HIGH BYTE ADDRESS OF TACS
389 001212          177502          TADBL:         177502          ;LOW BYTE ADDRESS OF TADB
390 001214          177503          TADBH:         177503          ;HIGH BYTE ADDRESS OF TADB
391 001216          000260          000262          TAVEC:         260,262          ;TA11 VECTOR ADDRESS

```

TA11 MANUAL INTERVENTION TEST
DZTACC.NEW COMMON TAGS

MAINDEC-11-DZTAC-C

NACY11 27(732) 11-JUN-76 13:51 PAGE 10

SEQ 0023

392	001222	000300	
393	001224	000000	000000
394	001230	001224	
395	001232	000000	
396	001234	000000	

TAPRIO: 300
DRVKEY: 0,0
DRVPNT: DRVKEY
ASKKEY: 0
CURDRV: 0

;TA11 BR LEVEL 6
;DRIVE SELECT KEY:

;CURRENT DRIVE BEING TESTED

397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411 001236
 412
 413
 414
 415
 416 001236 013061
 417 001240 013227
 418 001242 013340
 419 001244 000000
 420
 421
 422 001246 013100
 423 001250 013244
 424 001252 013346
 425 001254 000000
 426
 427
 428 001256 013126
 429 001260 013244
 430 001262 013346
 431 001264 000000
 432
 433
 434 001266 013167
 435 001270 013244
 436 001272 013346
 437 001274 000000
 438
 439
 440 001276 013212
 441 001300 013301
 442 001302 013356
 443 001304 000000
 444
 445

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

;NOTE: ALL NUMBERS ARE TYPED AS 6-DIGIT OCTAL NUMBERS

;ITEM 1
 EM1 ;STATUS PROBLEM
 DH1 ;PC TACS
 DT1 ;\$ERRPC \$REG0
 0

;ITEM 2
 EM2 ;READY FAILED TO SET
 DH2 ;PC TACS WAIT ADDRESS
 DT2 ;\$ERRPC \$REG0 SAVPC
 0

;ITEM 3
 EM3 ;TRANSFER REQUEST FAILED TO SET
 DH2 ;PC TACS WAIT ADDRESS
 DT2 ;\$ERRPC \$REG0 SAVPC
 0

;ITEM 4
 EM4 ;THE WRONG FLAG SET
 DH2 ;PC TACS WAIT ADDRESS
 DT2 ;\$ERRPC \$REG0 SAVPC
 0

;ITEM 5
 EM5 ;DATA PROBLEM
 DH5 ;PC TACS EXPECT RCV'D
 DT5 ;\$ERRPC \$REG0 \$GDDAT \$BDDAT
 0

446	001306		ITEMS2:	;ITEMS 201-202
447				
448	001306	013402	EN201	;TA11 FAILED TO RESPOND
449	001310	013454	DH201	;PC TACS
450	001312	013370	DT201	;SERRPC TACS
451	001314	000000	0	;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS
452				
453	001316	013431	EN202	;NO DRIVES AVAILABLE
454	001320	013471	DH202	;PC
455	001322	013376	DT202	;SERRPC
456	001324	000000	0	;
457				

```

458 ;////////////////////////////////////
459 ;////////////////////////////////////
460 ;:*****
461
462 ;BEGIN1 IS FOR NORMAL START
463 ;BEGIN2 IS FOR DRIVE SELECTION
464 ;BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
465 ;BEGIN4 IS FOR MANUAL OPERATION
466
467 ;:*****
468
469 001326 005005 BEGIN1: CLR R5 ;NORMAL START
470 001330 012737 041101 001224 MOV 0"AB,00DRVKEY
471 001336 122737 000005 000041 CMPB 05,0041 ;CASSETTE DDP?
472 001344 001015 BNE BGNCHN ;GO BEGIN COMMON CODE IF NO
473 001346 022737 000260 001216 CMP 0260,00TAVEC ;STANDARD VECTOR?
474 001354 001011 BNE BGNCHN ;GO BEGIN COMMON CODE IF NO
475 001356 000403 BR BEGIN3 ;GET DRIVES AND ADDRESSES
476 001360 012705 000001 BEGIN2: MOV 01,R5 ;ASK FOR DRIVES FLAG
477 001364 000405 BR BGNCHN ;BEGIN COMMON CODE
478 001366 012705 000002 BEGIN3: MOV 02,R5 ;ASK FOR DRIVES AND ADDRESSES
479 001372 000402 BR BGNCHN
480 001374 012705 000003 BEGIN4: MOV 03,R5
481 001400 BGNCHN:
482 SBTTL INITIALIZE THE COMMON TAGS
483 ;:CLEAR THE COMMON TAGS (SCHTAG) AREA
484 001400 012706 001100 MOV 0SCHTAG,R6 ;:FIRST LOCATION TO BE CLEARED
485 001404 005026 CLR (R6)+ ;:CLEAR MEMORY LOCATION
486 001406 022706 001140 CMP 0SWR,R6 ;:DONE?
487 001412 001374 BNE -6 ;:LOOP BACK IF NO
488 001414 012706 001100 MOV 0STACK,SP ;:SETUP THE STACK POINTER
489 ;:INITIALIZE A FEW VECTORS
490 001420 012737 006046 000020 MOV 00SCOPE,00IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
491 001426 012737 000340 000022 MOV 0340,00IOTVEC+2 ;:LEVEL 7
492 001434 012737 006320 000030 MOV 00ERROR,00EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
493 001442 012737 000340 000032 MOV 0340,00EMTVEC+2 ;:LEVEL 7
494 001450 012737 012304 000034 MOV 00TRAP,00TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
495 001456 012737 000340 000036 MOV 0340,00TRAPVEC+2 ;:LEVEL 7
496 001464 012737 012370 000024 MOV 00PWRDN,00PWRVEC ;:POWER FAILURE VECTOR
497 001472 012737 000340 000026 MOV 0340,00PWRVEC+2 ;:LEVEL 7
498 001500 016767 004270 004260 MOV 0ENDCT,0EOPCT ;:SETUP END-OF-PROGRAM COUNTER
499 001506 005067 177454 CLR 0TIMES ;:INITIALIZE NUMBER OF ITERATIONS
500 001512 005067 177452 CLR 0ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
501 001516 112767 000001 177371 MOVB 01,0ERMAX ;:ALLOW ONE ERROR PER TEST
502 001524 012767 001524 177354 MOV 0,,0LPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
503 001532 012767 001532 177350 MOV 0,,0LPERR ;:SETUP THE ERROR LOOP ADDRESS
504 ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
505 ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
506 001540 013746 000004 MOV 00ERRVEC,-(SP) ;:SAVE ERROR VECTOR
507 001544 012737 001600 000004 MOV 0640,00ERRVEC ;:SET UP ERROR VECTOR
508 001552 012767 177570 177360 MOV 0DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
509 001560 012767 177570 177354 MOV 0DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
510 001566 022777 177777 177344 CMP 0-1,0SWR ;:TRY TO REFERENCE HARDWARE SWR
511 001574 001012 BNE 666 ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
512 ;:AND THE HARDWARE SWR IS NOT = -1
513 001576 000403 BR 656 ;:BRANCH IF NO TIMEOUT
    
```

m

```

514 001600 012716 001606      648:  MOV    #658,(SP)      ;;SET UP FOR TRAP RETURN
515 001604 000002
516 001606 012767 000176 177324 658:  MOV    #SWREG,SWR      ;;POINT TO SOFTWARE SWR
517 001614 012767 000174 177320  MOV    #DISPREG,DISPLAY
518 001622 012637 000004      668:  MOV    (SP)+,#ERRVEC  ;;RESTORE ERROR VECTOR
519
520
521      .SBTTL  TYPE PROGRAM NAME
      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
522 001626 005227 177777      INC    #-1          ;;FIRST TIME?
523 001632 001036      BNE    HERE        ;;BRANCH IF NO
524 001634 022737 006014 000042  CMP    #SENDAD,#42  ;;ACT-11?
525 001642 001432      BEQ    HERE        ;;BRANCH IF YES
526 001644 104401 001702      TYPE   ,MSGID      ;;TYPE ASCIZ STRING
527      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
528 001650 005737 000042      TST    #42         ;;ARE WE RUNNING UNDER XDP/ACT?
529 001654 001006      BNE    678         ;;BRANCH IF YES
530 001656 026727 177256 000176  CMP    SWR,#SWREG   ;;SOFTWARE SWITCH REG SELECTED?
531 001664 001005      BNE    688         ;;BRANCH IF NO
532 001666 104405      GTSWR              ;;GET SOFT-SWR SETTINGS
533 001670 000403      BR     688
534 001672 112767 000001 177234 678:  MOVB   #1,#AUTOB   ;;SET AUTO-MODE INDICATOR
535 001700      688:
536 001700 000413      BR     HERE        ;;GET OVER THE ASCIZ
537      ;;MSGID:      .ASCIZ <CRLF>/MAINDEC-11-DZTAC-C/<CRLF>
538 001730      HERE:
539      ;;*****
540      ;;*****
541
542      ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
543      ;
544      ;      R5=3  MANUAL OPERATIONS
545      ;      R5=2  ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
546      ;      R5=1  ASK FOR DRIVE(S)
547      ;      R5=0  DON'T ASK FOR ANYTHING
548      ;
549      ;;*****
550 001730 010504      BEGINX: MOV    R5,R4      ;COPY R5
551 001732 005305      DEC    R5          ;ASK FOR DRIVES?
552 001734 002406      BLT    CHKADR      ;BR IF NO
553 001736 004737 007036      JSR    PC,#ASKDRV  ;GO GET DRIVES TO BE TESTED
554 001742 005305      DEC    R5          ;ASK FOR ADDRESSES?
555 001744 002402      BLT    CHKADR      ;BR IF NO
556 001746 004737 007146      JSR    PC,#ASKADR  ;GO GET TA11 ADDRESSES
557      ;;*****
558      ;;*****
559
560      ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
561      ;
562      ;I.  TIMEOUT OCCURRED
563      ;      A. TYPE ERROR MESSAGE
564      ;      B. EXAMINE R4
565      ;          1. R4>0 GOTO BEGINX
566      ;          2. R4=0 EXAMINE (42)
567      ;              A. (42)=0 GOTO BEGINX
568      ;              B. (42)>0 GOTO SENDAD
569      ;

```

```

570                                     ;II.  TIMEOUT DIDN'T OCCUR
571                                     ;      A. CONTINUE
572                                     ;
573                                     ;*****
574 001752 012737 001770 000004 CHKADR: MOV 010,00ERRVEC ;IN CASE OF TIMEOUTS
575 001760 005000                                     ;USE AS A SWITCH
576 001762 005777 177220                                     ;SEE IF TA11 RESPONDS
577 001766 000402                                     ;BR IF NO TIMEOUT
578 001770 005200                                     ;COME HERE ON TIMEOUT
579 001772 022626                                     ;CLEANUP THE STACK
580 001774 012737 000006 000004 20: MOV 0ERRVEC+2,00ERRVEC ;RESTORE TIMEOUT VECTOR
581 002002 005700                                     ;DID A TIMEOUT OCCUR?
582 002004 001412                                     ;BR IF NO
583 002006 104201                                     ;TA11 FAILED TO RESPOND
584 002010 012705 000002                                     ;DRIVES & ADDRESSES
585 002014 005704                                     ;OPERATOR INPUTS?
586 002016 001344                                     ;BR IF YES
587 002020 013700 000042                                     ;GET MONITOR RETURN ADDRESS
588 002024 001741                                     ;BR IF NO MONITOR
589 002026 000137 006014                                     ;GO TO END
590 002032                                     30:
  
```

591
 592
 593
 594
 595
 596
 597
 598
 599
 600
 601
 602
 603
 604
 605
 606
 607
 608
 609
 610
 611
 612 002032 012700 001224
 613 002036 004737 010654
 614 002042 000410
 615 002044 116010 000001
 616 002050 001412
 617 002052 004737 010654
 618 002056 000407
 619 002060 005010
 620 002062 000405
 621 002064 005200
 622 002066 004737 010654
 623 002072 000401
 624 002074 105010
 625 002076 012700 001224
 626 002102 010037 001230
 627 002106 121060 000001
 628 002112 001002
 629 002114 105060 000001
 630 002120 005710
 631 002122 001401
 632 002124 000412
 633 002126 104202
 634 002130 012705 000002
 635 002134 005704
 636 002136 001274
 637 002140 013700 000042
 638 002144 001671
 639 002146 000137 006014
 640 002152 020427 000003
 641 002156 001002
 642 002160 016704 175613
 643 002164 010437 001232
 644 002170 000405
 645 002172 104401 001702
 646 002176 012737 001224 001230

```

;*****
;*****
;MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED
;
;I. DESIRED DRIVES CAN NOT BE TESTED
; A. TYPE ERROR MESSAGE
; B. EXAMINE R4
;     1. R4>0 GOTO BEGINX
;     2. R4=0 EXAMINE (42)
;         A. (42)=0 GOTO BEGINX
;         B. (42)>0 GOTO SENDAD
;
;II. BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
; A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
; B. CONTINUE IN PROGRAM
;
;III. DESIRED DRIVE(S) CAN BE TESTED
; A. CONTINUE IN PROGRAM
;*****
CHKDRV: MOV     0DRVKEY,R0      ;PICKUP ADDRESS OF ASCII DRIVE KEY
        JSR    PC,00EXAM     ;GO EXAMINE FIRST DRIVE
        BR     10           ;OK TO TEST---GO CHECK NEXT
        MOVB   1(R0),(R0)    ;REPLACE 1ST WITH 2ND
        BEQ    20           ;BR IF NO 2ND DRIVE SELECTED
        JSR    PC,00EXAM     ;GO EXAMINE DRIVE
        BR     20           ;OK TO TEST
        CLR    (R0)         ;CLEAR DRIVE CODES
        BR     20
10:     INC    R0            ;POINT TO 2ND
        JSR    PC,00EXAM     ;GO EXAMINE DRIVE
        BR     20           ;OK TO TEST
        CLRB   (R0)         ;CLEAR 2ND
        BR     20           ;RESET ADDRESS POINTERS
20:     MOV    0DRVKEY,R0
        MOV    R0,00DRVPNT
        CMPB   (R0),1(R0)    ;1ST = 2ND?
        BNE   30           ;BR IF NO
        CLRB   1(R0)        ;YES---CLEAR 2ND
        TST    (R0)         ;ANY DRIVES?
        BEQ    50           ;BR IF NO
        BR     MANUAL
50:     ERROR  202          ;NO DRIVES AVAILABLE
        MOV    02,R5        ;DRIVES & ADDRESS
        TST    R4           ;OPERATOR INPUTS?
        BNE   BEGINX       ;BR IF YES
        MOV    0042,R0      ;GET MONITOR RETURN ADDRESS
        BEQ    BEGINX       ;NO MONITOR
        JMP    00SENDAD     ;GO TO END
MANUAL: CMP    R4,03
        BNE   OK
        MOV    -1,R4
OK:     MOV    R4,00ASKKEY
        BR     START
PWRST: TYPE   ,NSCID       ;POWER FAIL RESTART
        MOV    0DRVKEY,00DRVPNT
  
```

```

647 002204 013777 001220 177004 START: MOV 00TAVEC+2,0TAVEC ;SETUP TA11 TRAP VECTOR
648 002212 005077 177002 CLR 0TAVEC+2
649 002216 012737 000340 177776 MOV 0340,00PS ;LOCKOUT ALL I/O INT
650 002224 013704 001206 MOV 00TACSL,TACS ;SETUP TACS
651 002230 013705 001212 MOV 00TADBL,TADB ;SETUP TADB
652 002234 005737 001100 TST 00SPASS ;IF FIRST PASS SETUP FOR EXTRA LONG WAIT LOOPS
653 002240 001003 BNE 10 ;OTHERWISE USE OLD VALUES
654 002242 012737 077777 006732 MOV 0*CBIT15,00MAXCNT
655 002250 005037 001102 10: CLR 000TSTNM ;ZERO THE TEST NUMBER
656 002254 005003 CLR DRIVE ;SET DRIVE TO "A"
657 002256 013701 001230 MOV 00DRVPNT,R1 ;GET DRIVE POINTER
658 002262 121127 000101 CMPB (R1),0'A ;IS IT DRIVE "A"?
659 002266 001402 BEQ TDRV ;BR IF YES
660 002270 012703 000400 MOV 0UNIT,DRIVE ;SET DRIVE TO "B"
661 002274 TDRV:
662 002274 104401 002302 TYPE ,658 ;;TYPE ASCIZ STRING
663 002300 000411 BR 648 ;;GET OVER THE ASCIZ
664 ;;658: .ASCIZ <15><12>*TESTING DRIVE *
665 648:
666 002324 112167 176704 MOVB (R1)+,CURDRV ;SETUP TO TYPE CURRENT DRIVE
667 002330 104401 001234 TYPE ,CURDRV
668 002334 104401 001177 TYPE ,0CRLF ;TYPE A CR & LF
669 002340 105711 TSTB (R1) ;LAST DRIVE BEEN SELECTED
670 002342 001002 BNE 10 ;BR IF NO
671 002344 012701 001224 MOV 0DRVKEY,R1 ;RESET DRIVE POINTER
672 002350 010137 001230 10: MOV R1,00DRVPNT ;SAVE DRIVE POINTER FOR NEXT TIME
673 002354 005737 001232 TST 00ASKKEY ;GO START TESTING IF NO MANUAL
674 002360 002007 BGE 20 ; OPERATIONS REQUESTED
675 002362 005000 CLR R0
676 002364 000000 HALT ;GIVE CONTROL TO THE OPERATOR
677 002366 022767 000176 176544 CMP 0SWREG,SWR ;USING S/W SWITCH REG?
678 002374 001001 BNE 200 ;NO- GET OUT
679 002376 104405 GTSWR ;GET VALUE
680 002400 200: ;CONTINUE
681 ;THIS CODE IS FOR ACT11 & DDP
682 002400 005737 000042 20: TST 0042 ;IS THERE A MONITOR?
683 002404 001406 BEQ TST1 ;GO START TESTING IF NO
684 002406 010314 MOV DRIVE,0TACS ;IF YES SELECT DRIVE
685 002410 112714 000017 MOVB 0REWIND!GO,0TACS ;SEND TAPE TO BOT
686 002414 032714 000040 30: BIT 0READY,0TACS ;WAIT ON READY
687 002420 001775 BEQ 30 ;FALL THRU IF READY=1

```

```

688 ;////////////////////////////////////
689 ;////////////////////////////////////
690 ;*****
691 ;THIS ISN'T A REAL TEST BUT A SMALL ROUTINE TO DETERMINE THE MAX,
692 ;TIME FOR THE WAIT LOOPS (WAIT FOR "READY" AND "TRANSFER REQUEST")
693 ;*****
694 ;*TEST 1 ROUTINE TO DETERMINE TIME OF WAIT LOOPS
695 ;*****
696 002422 000004 TST1: SCOPE
697 002424 012767 000001 176534 MOV 01,0TIMES ;;DO 1 ITERATION
698 002432 005737 001100 TST 00SPASS ;IS THIS THE FIRST PASS?
699 002436 001021 BNE TST2 ;;BR IF NO
700 002440 000005 RESET
701 002442 010314 MOV DRIVE,0TACS ;SELECT THE DRIVE
702 002444 112714 000017 MOVB 0REWIND!GO,0TACS ;START A REWIND
703 002450 104412 WAITREADY ;WAIT FOR READY
704 002452 112714 000001 MOVB 0WFG!GO,0TACS ;WRITE A FILE GAP
705 002456 104412 WAITREADY ;WAIT ON READY
706 002460 163737 006706 006732 SUB 00HGHTIM,00MAXCNT ;GET THE TIME IT TOOK
707 002466 005237 006732 INC 00MAXCNT ;MAKE IT BIGGER
708 002472 006137 006732 ROL 00MAXCNT
709 002476 006137 006732 ROL 00MAXCNT
710 ;*****
711 ;*TEST 2 SETUP FOR MANUAL INTERVENTION
712 ;*****
713 002502 000004 TST2: SCOPE
714 002504 012767 000001 176454 MOV 01,0TIMES ;;DO 1 ITERATION
715 002512 005767 176362 TST 0PASS ;IS THIS THE FIRST PASS?
716 002516 001053 BNE TST3 ;;SKIP TYPE OUT IF YES
717 002520 104401 002526 TYPE ,650 ;;TYPE ASCIZ STRING
718 002524 000423 BR 648 ;;GET OVER THE ASCIZ
719 ;650: .ASCIZ <15><12>"DIRECTIONS WILL BE TYPED ON THE TTY"
720 648:
721 002574 104401 002602 TYPE ,670 ;;TYPE ASCIZ STRING
722 002600 000422 BR 668 ;;GET OVER THE ASCIZ
723 ;670: .ASCIZ <15><12>"HIT 'CR' WHEN READY TO CONTINUE"<15><12>
724 668:
725 ;*****
726 ;*TEST 3 SETUP FOR POWER DOWN TU60 TEST
727 ;*****
728 002646 000004 TST3: SCOPE
729 002650 012767 000012 176310 MOV 010,,0TIMES ;;DO 10. ITERATIONS
730 002656 012767 002700 176222 MOV 010,0LPADR ;;SET SCOPE LOOP ADDRESS
731 002664 004037 007436 JSR R0,0ASKQUES
732 002670 012874 MTUPWR
733 002672 000402 BR 18 ;DO THIS TEST
734 002674 000137 003120 JMP 00TST6 ;SKIP POWER FAIL TESTS
735 002700 000005 18: RESET ;CLEAR INTERFACE
736 002702 010314 MOV DRIVE,0TACS ;SELECT DRIVE
737 002704 032714 001000 BIT 0OFFLINE,0TACS ;IS "OFFLINE" ON A ONE?
738 002710 001001 BKE TST4 ;;BR IF YES
739 002712 104001 ERROR 1 ;"OFFLINE" BIT ISN'T ON A ONE
740 ;*****
741 ;*TEST 4 TEST "OFFLINE" WHEN TU60 IS POWERED DOWN
742 ;*****
743 002714 000004 TST4: SCOPE
    
```

```

744 002716 012767 003062 176244      MOV      @TST5,@ESCAPE      ;;ESCAPE TO TEST 5 ON ERROR
745 002724 112714 000001      MOVB     @WFG!GO,@TACS      ;TRY TO DO A "WFG"
746 002730 104412      WAITREADY      ;WAIT IN "READY"
747 002732 005714      TST      @TACS              ;CHECK FOR AN "ERROR"
748 002734 100401      BMI      10                 ;BR IF "ERROR"=1
749 002736 104001      ERROR    1                  ;"ERROR"=0
750 002740 112714 000003      10:     MOVB     @WRITE!GO,@TACS ;TRY TO DO A "WRITE"
751 002744 104412      WAITREADY      ;WAIT ON "READY"
752 002746 005714      TST      @TACS              ;CHECK ERROR FLAG
753 002750 100401      BMI      20
754 002752 104001      ERROR    1                  ;"ERROR"=0
755 002754 112714 000005      20:     MOVB     @READ!GO,@TACS  ;TRY TO DO A "READ"
756 002760 104412      WAITREADY      ;WAIT ON "READY"
757 002762 005714      TST      @TACS              ;CHECK ERROR FLAG
758 002764 100401      BMI      30
759 002766 104001      ERROR    1                  ;"ERROR"=0
760 002770 112714 000007      30:     MOVB     @BSFG!GO,@TACS  ;TRY TO DO A "BSFG"
761 002774 104412      WAITREADY      ;WAIT ON "READY"
762 002776 005714      TST      @TACS              ;CHECK ERROR FLAG
763 003000 100401      BMI      40
764 003002 104001      ERROR    1                  ;"ERROR"=0
765 003004 112714 000011      40:     MOVB     @BSBG!GO,@TACS  ;TRY TO DO A "BSBG"
766 003010 104412      WAITREADY      ;WAIT ON "READY"
767 003012 005714      TST      @TACS              ;CHECK ERROR FLAG
768 003014 100401      BMI      50
769 003016 104001      ERROR    1                  ;"ERROR"=0
770 003020 112714 000013      50:     MOVB     @SFFG!GO,@TACS  ;TRY TO DO A "SFFG"
771 003024 104412      WAITREADY      ;WAIT ON "READY"
772 003026 005714      TST      @TACS              ;CHECK ERROR FLAG
773 003030 100401      BMI      60
774 003032 104001      ERROR    1                  ;"ERROR"=0
775 003034 112714 000015      60:     MOVB     @SFBG!GO,@TACS  ;TRY TO DO A "SFBG"
776 003040 104412      WAITREADY      ;WAIT ON "READY"
777 003042 005714      TST      @TACS              ;CHECK ERROR FLAG
778 003044 100401      BMI      70
779 003046 104001      ERROR    1                  ;"ERROR"=0
780 003050 112714 000017      70:     MOVB     @REWIND!GO,@TACS ;TRY TO DO A REWIND
781 003054 005714      TST      @TACS              ;CHECK ERROR FLAG
782 003056 100401      BMI      TST5                ;;BR IF ERROR=1
783 003060 104001      ERROR    1                  ;"ERROR"=0
784                                     ;;*****
785 ;*TEST 5          POWER UP THE TU60
786                                     ;;*****
787 003062 000004      TST5:   SCOPE
788 003064 012767 003104 176014      MOV      @20,@LPADR        ;;SET SCOPE LOOP ADDRESS
789 003072 004037 007436      JSR      R0,@ASKQUES
790 003076 012714      MPMRUP      ;MESSAGE POINTER
791 003100 000240      NOP
792 003102 000005      10:     RESET
793 003104 010314      20:     MOV      DRIVE,@TACS    ;SELECT DRIVE
794 003106 104412      WAITREADY      ;WAIT ON READY
795 003110 032714 001000      BIT      @OFFLINE,@TACS    ;DID OFFLINE CLEAR?
796 003114 001401      BEQ      TST6                ;;BR IF YES
797 003116 104001      ERROR    1                  ;OFFLINE = 1
798                                     ;;*****
799 ;*TEST 6          TEST "OFFLINE" WHEN DRIVE IS EMPTY
    
```

```

000 ;;*****
001 003120 000004 TST6: SCOPE
002 003122 012767 003146 175756 MOV 020,0LPADR ;;SET SCOPE LOOP ADDRESS
003 003130 004037 007436 JSR R0,00ASKQUES
004 003134 012732 MOFFLN ;MESSAGE POINTER
005 003136 000402 BR 10 ;DO TEST
006 003140 000137 003224 JMP 00TST10 ;SKIP THIS TEST
007 003144 000005 10: RESET ;CLEAR THE WORLD
008 003146 010314 20: NOV DRIVE,0TACS ;SELECT DRIVE
009 003150 042714 001000 BIC 0OFFLINE,0TACS ;TRY TO CLEAR OFFLINE INDICATION
010 003154 032714 001000 BIT 0OFFLINE,0TACS ;TEST FOR "OFFLINE"
011 003160 001001 BNE TST7 ;;BR IF "OFFLINE"=1
012 003162 104001 ERROR 1 ;"OFFLINE"=0
013 ;;*****
014 ;*TEST 7 PUT DRIVE "ONLINE"
015 ;;*****
016 003164 000004 TST7: SCOPE
017 003166 012767 003212 175712 MOV 020,0LPADR ;;SET SCOPE LOOP ADDRESS
018 003174 004037 007436 JSR R0,00ASKQUES
019 003200 013000 NTAPE
020 003202 000402 BR 10 ;DO TEST
021 003204 000137 003224 JMP 00TST10 ;SKIP THIS TEST
022 003210 000005 10: RESET
023 003212 010314 20: NOV DRIVE,0TACS ;SELECT DRIVE
024 003214 032714 001000 BIT 0OFFLINE,0TACS ;IS DRIVE OFFLINE?
025 003220 001401 BEQ TST10 ;;BR IF NO
026 003222 104001 ERROR 1 ;DRIVE IS OFFLINE
027 ;;*****
028 ;*TEST 10 PUT DATA ON TAPE FOR WRITE LOCK TEST
029 ;;*****
030 003224 000004 TST10: SCOPE
031 003226 012767 000001 175732 MOV 01,0TIMES ;;DO 1 ITERATION
032 003234 012767 003242 175726 MOV 010,0ESCAPE ;;ESCAPE TO 10 ON ERROR
033 ;*****
034 ;HANG IN THIS TEST UNTIL DATA IS ON TAPE WITH NO ERRORS
035 003242 000005 10: RESET
036 003244 010314 NOV DRIVE,0TACS
037 003246 112714 000017 MOV 0REWIND!GO,0TACS
038 003252 104412 WAITREADY ;WAIT ON READY
039 003254 005714 TST 0TACS ;CHECK FOR ERROR
040 003256 100001 BPL 20
041 003260 104001 ERROR 1 ;"ERROR" = 1
042 003262 20:
043 ;WRITE A BLOCK OF 6 BYTES
044 003262 012767 000006 000014 MOV 06,658 ;SETUP FOR 6 BYTES
045 003270 112714 000003 MOV 0WRITE!GO,0TACS ;START A WRITE
046 003274 104413 648: WAITXFER ;WAIT ON TRANSFER REQUEST
047 003276 112715 000377 MOV 0377,0TADB ;LOAD DATA BUFFER
048 003302 005327 DEC (PC)+ ;MORE TO DO?
049 003304 000000 658: 0 ;NUMBER OF BYTES TO WRITE GOES HERE
050 003306 003372 BGT 648 ;BR IF YES
051 003310 104413 WAITXFER ;WAIT ON TRANSFER REQUEST
052 003312 052714 000020 BIS 0ILBS,0TACS ;WRITE CRC
053 003316 104412 WAITREADY ;WAIT ON READY
054 ;WRITE A BLOCK OF 6 BYTES
055 003320 012767 000006 000014 MOV 06,678 ;SETUP FOR 6 BYTES
    
```

```

856 003326 112714 000003      MOVB  @WRITE!GO,@TACS      ;START A WRITE
857 003332 104413      668: WAITXFER                ;WAIT ON TRANSFER REQUEST
858 003334 112715 000377      MOVB  @377,@TADB           ;LOAD DATA BUFFER
859 003340 005327      DEC   (PC)+                ;MORE TO DO?
860 003342 000000      678: @                     ;NUMBER OF BYTES TO WRITE GOES HERE
861 003344 003372      BGT   668                  ;BR IF YES
862 003346 104413      WAITXFER                  ;WAIT ON TRANSER REQUEST
863 003350 052714 000020      BIS   @ILBS,@TACS         ;WRITE CRC
864 003354 104412      WAITREADY                 ;WAIT ON READY
865 003356 112714 000001      MOVB  @WFG!GO,@TACS       ;WRITE A FILE GAP
866 003362 104412      WAITREADY                 ;WAIT ON "READY"
867                                     ;WRITE A BLOCK OF 6 BYTES
868 003364 012767 000006 000014  MOV   @6,698               ;SETUP FOR 6 BYTES
869 003372 112714 000003      MOVB  @WRITE!GO,@TACS     ;START A WRITE
870 003376 104413      688: WAITXFER                ;WAIT ON TRANSFER REQUEST
871 003400 112715 000377      MOVB  @377,@TADB           ;LOAD DATA BUFFER
872 003404 005327      DEC   (PC)+                ;MORE TO DO?
873 003406 000000      698: @                     ;NUMBER OF BYTES TO WRITE GOES HERE
874 003410 003372      BGT   688                  ;BR IF YES
875 003412 104413      WAITXFER                  ;WAIT ON TRANSER REQUEST
876 003414 052714 000020      BIS   @ILBS,@TACS         ;WRITE CRC
877 003420 104412      WAITREADY                 ;WAIT ON READY
878                                     ;*****
879                                     ;*TEST 11      TEST "WRITE LOCK" WHEN AT CLEAR LEADER
880                                     ;*****
881 003422 000004      TST11: SCOPE
882 003424 012767 000001 175534  MOV   @1,@TINES           ;;DO 1 ITERATION
883 003432 012767 003462 175446  MOV   @10,@LPADR          ;;SET SCOPE LOOP ADDRESS
884 003440 012767 003772 175522  MOV   @TST12,@ESCAPE      ;;ESCAPE TO TEST 12 ON ERROR
885 003446 004037 007436      JSR   @R0,@ASKQUES        ;HAVE OPERATOR SET "WRITE LOCK"
886 003452 013021      MWRTLK
887 003454 000402      BR    10                  ;DO TEST
888 003456 000137 005716      JMP   @TST22              ;SKIP WRITE LOCK TESTS
889 003462                                     18:
890 003462 000005      RESET
891 003464 010314      MOV   @DRIVE,@TACS        ;CLEAR ALL
892 003466 112714 000017      MOVB  @REWIND!GO,@TACS    ;SELECT DRIVE
893 003472 104412      WAITREADY                 ;GO TO "BOT"
894 003474 005714      TST   @TACS               ;IS "ERROR"=1
895 003476 100001      BPL   28                  ;BR IF NO
896 003500 104001      ERROR 1                   ;"ERROR"=1
897 003502 032714 020000 28:  BIT   @LEADER,@TACS       ;CHECK FOR CLEAR LEADER
898 003506 001001      BNE   38                  ;BR IF ON CLEAR LEADER
899 003510 104001      ERROR 1                   ;REWIND DIDN'T GO TO BOT
900 003512                                     38:
901 003512 112714 000000      MOVB  @WFG,@TACS          ;LOAD A "WFG"
902 003516 032714 010000      BIT   @WRTLOCK,@TACS     ;IS "WRITE LOCK ERROR" = "1"
903 003522 001001      BNE   648                 ;BR IF YES
904 003524 104001      ERROR 1                   ;"WRITE LOCK ERROR" NOT EQUAL "1"
905 003526                                     648:
906 003526 112714 000002      MOVB  @WRITE,@TACS        ;LOAD A "WRITE"
907 003532 032714 010000      BIT   @WRTLOCK,@TACS     ;IS "WRITE LOCK ERROR" = "1"
908 003536 001001      BNE   658                 ;BR IF YES
909 003540 104001      ERROR 1                   ;"WRITE LOCK ERROR" NOT EQUAL "1"
910 003542                                     658:
911 003542 112714 000004      MOVB  @READ,@TACS        ;LOAD A "READ"
    
```

912	003546	032714	010000	BIT	0WRTLOCK,0TACS	;IS "WRITE LOCK ERROR" = "0"
913	003552	001401		BEG	660	;BR IF YES
914	003554	104001		ERROR	1	;"WRITE LOCK ERROR" NOT EQUAL "0"
915	003556		660:			
916	003556	112714	000006	MOVB	0BSFG,0TACS	;LOAD A "BSFG"
917	003562	032714	010000	BIT	0WRTLOCK,0TACS	;IS "WRITE LOCK ERROR" = "0"
918	003566	001401		BEG	670	;BR IF YES
919	003570	104001		ERROR	1	;"WRITE LOCK ERROR" NOT EQUAL "0"
920	003572		670:			
921	003572	112714	000010	MOVB	0BSBG,0TACS	;LOAD A "BSBG"
922	003576	032714	010000	BIT	0WRTLOCK,0TACS	;IS "WRITE LOCK ERROR" = "0"
923	003602	001401		BEG	680	;BR IF YES
924	003604	104001		ERROR	1	;"WRITE LOCK ERROR" NOT EQUAL "0"
925	003606		680:			
926	003606	112714	000012	MOVB	0SFPG,0TACS	;LOAD A "SFPG"
927	003612	032714	010000	BIT	0WRTLOCK,0TACS	;IS "WRITE LOCK ERROR" = "0"
928	003616	001401		BEG	690	;BR IF YES
929	003620	104001		ERROR	1	;"WRITE LOCK ERROR" NOT EQUAL "0"
930	003622		690:			
931	003622	112714	000014	MOVB	0SFPG,0TACS	;LOAD A "SFPG"
932	003626	032714	010000	BIT	0WRTLOCK,0TACS	;IS "WRITE LOCK ERROR" = "0"
933	003632	001401		BEG	700	;BR IF YES
934	003634	104001		ERROR	1	;"WRITE LOCK ERROR" NOT EQUAL "0"
935	003636		700:			
936	003636	112714	000016	MOVB	0REWIND,0TACS	;LOAD A "REWIND"
937	003642	032714	010000	BIT	0WRTLOCK,0TACS	;IS "WRITE LOCK ERROR" = "0"
938	003646	001401		BEG	710	;BR IF YES
939	003650	104001		ERROR	1	;"WRITE LOCK ERROR" NOT EQUAL "0"
940	003652		710:			
941	003652	112714	000000	MOVB	0WFG,0TACS	;CHECK "ERROR" WITH "WFG"
942	003656	005714		TST	0TACS	;SAMPLE THE "ERROR" BIT
943	003660	100401		BMI	720	;BR IF "ERROR" = 1
944	003662	104001		ERROR	1	;"ERROR" NOT = 1
945	003664		720:			
946	003664	112714	000002	MOVB	0WRITE,0TACS	;CHECK "ERROR" WITH "WRITE"
947	003670	005714		TST	0TACS	;SAMPLE THE "ERROR" BIT
948	003672	100401		BMI	730	;BR IF "ERROR" = 1
949	003674	104001		ERROR	1	;"ERROR" NOT = 1
950	003676		730:			
951	003676	112714	000004	MOVB	0READ,0TACS	;CHECK "ERROR" WITH "READ"
952	003702	005714		TST	0TACS	;SAMPLE THE "ERROR" BIT
953	003704	100401		BMI	740	;BR IF "ERROR" = 1
954	003706	104001		ERROR	1	;"ERROR" NOT = 1
955	003710		740:			
956	003710	112714	000006	MOVB	0BSFG,0TACS	;CHECK "ERROR" WITH "BSFG"
957	003714	005714		TST	0TACS	;SAMPLE THE "ERROR" BIT
958	003716	100401		BMI	750	;BR IF "ERROR" = 1
959	003720	104001		ERROR	1	;"ERROR" NOT = 1
960	003722		750:			
961	003722	112714	000010	MOVB	0BSBG,0TACS	;CHECK "ERROR" WITH "BSBG"
962	003726	005714		TST	0TACS	;SAMPLE THE "ERROR" BIT
963	003730	100401		BMI	760	;BR IF "ERROR" = 1
964	003732	104001		ERROR	1	;"ERROR" NOT = 1
965	003734		760:			
966	003734	112714	000012	MOVB	0SFPG,0TACS	;CHECK "ERROR" WITH "SFPG"
967	003740	005714		TST	0TACS	;SAMPLE THE "ERROR" BIT

```

968 003742 100401      BNI      778      ;BR IF "ERROR" = 1
969 003744 104001      ERROR    1        ;"ERROR" NOT = 1
970 003746           778:
971 003746 112714 000014      MOVB    @SFBG,@TACS ;CHECK "ERROR" WITH "SFBG"
972 003752 005714      TST     @TACS     ;SAMPLE THE "ERROR" BIT
973 003754 100401      BNI      788      ;BR IF "ERROR" = 1
974 003756 104001      ERROR    1        ;"ERROR" NOT = 1
975 003760           788:
976 003760 112714 000016      MOVB    @REWIND,@TACS ;CHECK "ERROR" WITH "REWIND"
977 003764 005714      TST     @TACS     ;SAMPLE THE "ERROR" BIT
978 003766 100001      BPL     798      ;BR IF "ERROR" = 0
979 003770 104001      ERROR    1        ;"ERROR" NOT = 0
980 003772
981 ;;.....
982 ;*TEST 12      TEST "WRITE LOCK" WITH "CRC ERROR"
983 ;;.....
984 003772 000004      TST12: SCOPE
985 003774 012767 004052 175104      MOV     @10,@LPADR ;;SET SCOPE LOOP ADDRESS
986 004002 012767 004332 175160      MOV     @TST13,@ESCAPE ;;ESCAPE TO TEST 13 ON ERROR
987 004010 000005      RESET
988 004012 010314      MOV     DRIVE,@TACS ;SELECT DRIVE
989 004014 112714 000017      MOVB    @REWIND!GO,@TACS ;GO TO BOT
990 004020 104412      WAITREADY ;WAIT ON READY
991 004022 112714 000005      MOVB    @READ!GO,@TACS ;GET OVER FIRST BLOCK OF DATA
992 004026 104413      WAITXFER
993 004030 105715      TSTB   @TADB      ;KNOCK DOWN "XFER REQ"
994 004032 104413      WAITXFER
995 004034 052714 000020      BIS     @ILBS,@TACS ;SHUT DOWN GENERATE CRC ERR
996 004040 104412      WAITREADY
997 004042 032714 040000      BIT     @CRCERR,@TACS ;IS CRC ERROR FLAG=1?
998 004046 001001      BNE     10        ;BR IF YES
999 004050 104001      ERROR    1        ;NO CRC ERROR
1000 004052           10:
1001 004052 112714 000000      MOVB    @WFG,@TACS ;CHECK "ERROR" WITH "WFG"
1002 004056 005714      TST     @TACS     ;SAMPLE THE "ERROR" BIT
1003 004060 100401      BNI      648      ;BR IF "ERROR" = 1
1004 004062 104001      ERROR    1        ;"ERROR" NOT = 1
1005 004064           648:
1006 004064 112714 000002      MOVB    @WRITE,@TACS ;CHECK "ERROR" WITH "WRITE"
1007 004070 005714      TST     @TACS     ;SAMPLE THE "ERROR" BIT
1008 004072 100401      BNI      658      ;BR IF "ERROR" = 1
1009 004074 104001      ERROR    1        ;"ERROR" NOT = 1
1010 004076           658:
1011 004076 112714 000004      MOVB    @READ,@TACS ;CHECK "ERROR" WITH "READ"
1012 004102 005714      TST     @TACS     ;SAMPLE THE "ERROR" BIT
1013 004104 100401      BNI      668      ;BR IF "ERROR" = 1
1014 004106 104001      ERROR    1        ;"ERROR" NOT = 1
1015 004110           668:
1016 004110 112714 000006      MOVB    @BSFG,@TACS ;CHECK "ERROR" WITH "BSFG"
1017 004114 005714      TST     @TACS     ;SAMPLE THE "ERROR" BIT
1018 004116 100001      BPL     678      ;BR IF "ERROR" = 0
1019 004120 104001      ERROR    1        ;"ERROR" NOT = 0
1020 004122           678:
1021 004122 112714 000010      MOVB    @BSBG,@TACS ;CHECK "ERROR" WITH "BSBG"
1022 004126 005714      TST     @TACS     ;SAMPLE THE "ERROR" BIT
1023 004130 100001      BPL     688      ;BR IF "ERROR" = 0
    
```

1024	004132	104001		ERROR	1	; "ERROR" NOT = 0	
1025	004134						
1026	004134	112714	000012	688:	MOVB	%SFFG,%TACS	; CHECK "ERROR" WITH "SFFG"
1027	004140	005714			TST	%TACS	; SAMPLE THE "ERROR" BIT
1028	004142	100001			BPL	698	; BR IF "ERROR" = 0
1029	004144	104001			ERROR	1	; "ERROR" NOT = 0
1030	004146						
1031	004146	112714	000014	698:	MOVB	%SFBG,%TACS	; CHECK "ERROR" WITH "SFBG"
1032	004152	005714			TST	%TACS	; SAMPLE THE "ERROR" BIT
1033	004154	100001			BPL	708	; BR IF "ERROR" = 0
1034	004156	104001			ERROR	1	; "ERROR" NOT = 0
1035	004160						
1036	004160	112714	000016	708:	MOVB	%REWIND,%TACS	; CHECK "ERROR" WITH "REWIND"
1037	004164	005714			TST	%TACS	; SAMPLE THE "ERROR" BIT
1038	004166	100001			BPL	718	; BR IF "ERROR" = 0
1039	004170	104001			ERROR	1	; "ERROR" NOT = 0
1040	004172						
1041	004177	112714	000000	718:	MOVB	%WFG,%TACS	; LOAD A "WFG"
1042	004176	032714	010000		BIT	%WRTLOCK,%TACS	; IS "WRITE LOCK ERROR" = "1"
1043	004202	001001			BNE	728	; BR IF YES
1044	004204	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "1"
1045	004206						
1046	004206	112714	000002	728:	MOVB	%WRITE,%TACS	; LOAD A "WRITE"
1047	004212	032714	010000		BIT	%WRTLOCK,%TACS	; IS "WRITE LOCK ERROR" = "1"
1048	004216	001001			BNE	738	; BR IF YES
1049	004220	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "1"
1050	004222						
1051	004222	112714	000004	738:	MOVB	%READ,%TACS	; LOAD A "READ"
1052	004226	032714	010000		BIT	%WRTLOCK,%TACS	; IS "WRITE LOCK ERROR" = "0"
1053	004232	001401			BEQ	748	; BR IF YES
1054	004234	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1055	004236						
1056	004236	112714	000006	748:	MOVB	%BSFG,%TACS	; LOAD A "BSFG"
1057	004242	032714	010000		BIT	%WRTLOCK,%TACS	; IS "WRITE LOCK ERROR" = "0"
1058	004246	001401			BEQ	758	; BR IF YES
1059	004250	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1060	004252						
1061	004252	112714	000010	758:	MOVB	%BSBG,%TACS	; LOAD A "BSBG"
1062	004256	032714	010000		BIT	%WRTLOCK,%TACS	; IS "WRITE LOCK ERROR" = "0"
1063	004262	001401			BEQ	768	; BR IF YES
1064	004264	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1065	004266						
1066	004266	112714	000012	768:	MOVB	%SFFG,%TACS	; LOAD A "SFFG"
1067	004272	032714	010000		BIT	%WRTLOCK,%TACS	; IS "WRITE LOCK ERROR" = "0"
1068	004276	001401			BEQ	778	; BR IF YES
1069	004300	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1070	004302						
1071	004302	112714	000014	778:	MOVB	%SFBG,%TACS	; LOAD A "SFBG"
1072	004306	032714	010000		BIT	%WRTLOCK,%TACS	; IS "WRITE LOCK ERROR" = "0"
1073	004312	001401			BEQ	788	; BR IF YES
1074	004314	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1075	004316						
1076	004316	112714	000016	788:	MOVB	%REWIND,%TACS	; LOAD A "REWIND"
1077	004322	032714	010000		BIT	%WRTLOCK,%TACS	; IS "WRITE LOCK ERROR" = "0"
1078	004326	001401			BEQ	798	; BR IF YES
1079	004330	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"

```

1000 004332          798:
1001                ;;*****
1002                ;*TEST 13      TEST "WRITE LOCK" WITHOUT ANY ERRORS
1003                ;;*****
1004 004332 000004   TST13:  SCOPE
1005 004334 012767 004364 174544   MOV      010,0LPADR      ;;SET SCOPE LOOP ADDRESS
1006 004342 012767 004644 174620   MOV      0TST14,0ESCAPE ;;ESCAPE TO TEST 14 ON ERROR
1007 004350 000005   RESET
1008 004352 010314   MOV      DRIVE,0TACS    ;SELECT DRIVE
1009 004354 032714 067000   BIT      0CRCERR!LEADER!FGAP!TIMERR!OFFLINE,0TACS ;CHECK STATUS
1010 004360 001401   BEQ      10             ;BR IF OK TO TEST "WRITE LOCK"
1011 004362 104001   ERROR    1             ;TAPE ISN'T POSITIONED PROPERLY
1012 004364          18:
1013 004364 112714 000000   MOVB     0WFG,0TACS     ;LOAD A "WFG"
1014 004370 032714 010000   BIT      0WRTLOCK,0TACS ;IS "WRITE LOCK ERROR" = "1"
1015 004374 001001   BNE      648           ;BR IF YES
1016 004376 104001   ERROR    1             ;"WRITE LOCK ERROR" NOT EQUAL "1"
1017 004400          648:
1018 004400 112714 000002   MOVB     0WRITE,0TACS   ;LOAD A "WRITE"
1019 004404 032714 010000   BIT      0WRTLOCK,0TACS ;IS "WRITE LOCK ERROR" = "1"
1020 004410 001001   BNE      658           ;BR IF YES
1021 004412 104001   ERROR    1             ;"WRITE LOCK ERROR" NOT EQUAL "1"
1022 004414          658:
1023 004414 112714 000004   MOVB     0READ,0TACS    ;LOAD A "READ"
1024 004420 032714 010000   BIT      0WRTLOCK,0TACS ;IS "WRITE LOCK ERROR" = "0"
1025 004424 001401   BEQ      668           ;BR IF YES
1026 004426 104001   ERROR    1             ;"WRITE LOCK ERROR" NOT EQUAL "0"
1027 004430          668:
1028 004430 112714 000006   MOVB     0BSFG,0TACS    ;LOAD A "BSFG"
1029 004434 032714 010000   BIT      0WRTLOCK,0TACS ;IS "WRITE LOCK ERROR" = "0"
1030 004440 001401   BEQ      678           ;BR IF YES
1031 004442 104001   ERROR    1             ;"WRITE LOCK ERROR" NOT EQUAL "0"
1032 004444          678:
1033 004444 112714 000010   MOVB     0BSBG,0TACS    ;LOAD A "BSBG"
1034 004450 032714 010000   BIT      0WRTLOCK,0TACS ;IS "WRITE LOCK ERROR" = "0"
1035 004454 001401   BEQ      688           ;BR IF YES
1036 004456 104001   ERROR    1             ;"WRITE LOCK ERROR" NOT EQUAL "0"
1037 004460          688:
1038 004460 112714 000012   MOVB     0SFFG,0TACS    ;LOAD A "SFFG"
1039 004464 032714 010000   BIT      0WRTLOCK,0TACS ;IS "WRITE LOCK ERROR" = "0"
1040 004470 001401   BEQ      698           ;BR IF YES
1041 004472 104001   ERROR    1             ;"WRITE LOCK ERROR" NOT EQUAL "0"
1042 004474          698:
1043 004474 112714 000014   MOVB     0SFBG,0TACS    ;LOAD A "SFBG"
1044 004500 032714 010000   BIT      0WRTLOCK,0TACS ;IS "WRITE LOCK ERROR" = "0"
1045 004504 001401   BEQ      708           ;BR IF YES
1046 004506 104001   ERROR    1             ;"WRITE LOCK ERROR" NOT EQUAL "0"
1047 004510          708:
1048 004510 112714 000016   MOVB     0REWIND,0TACS   ;LOAD A "REWIND"
1049 004514 032714 010000   BIT      0WRTLOCK,0TACS ;IS "WRITE LOCK ERROR" = "0"
1050 004520 001401   BEQ      718           ;BR IF YES
1051 004522 104001   ERROR    1             ;"WRITE LOCK ERROR" NOT EQUAL "0"
1052 004524          718:
1053 004524 112714 000000   MOVB     0WFG,0TACS     ;CHECK "ERROR" WITH "WFG"
1054 004530 005714   TST      0TACS          ;SAMPLE THE "ERROR" BIT
1055 004532 100401   BMI      728           ;BR IF "ERROR" = 1
    
```

```

1136 004534 104001          ERROR      1          ;"ERROR" NOT = 1
1137 004536          728:      MOVB      @WRITE,@TACS      ;CHECK "ERROR" WITH "WRITE"
1138 004536 112714 000002      TST      @TACS              ;SAMPLE THE "ERROR" BIT
1139 004542 005714          BMI      738              ;BR IF "ERROR" = 1
1140 004544 100401          ERROR      1          ;"ERROR" NOT = 1
1141 004546 104001          738:      MOVB      @READ,@TACS      ;CHECK "ERROR" WITH "READ"
1142 004550          TST      @TACS              ;SAMPLE THE "ERROR" BIT
1143 004550 112714 000004      BPL      748              ;BR IF "ERROR" = 0
1144 004554 005714          ERROR      1          ;"ERROR" NOT = 0
1145 004556 100001          748:      MOVB      @BSFG,@TACS      ;CHECK "ERROR" WITH "BSFG"
1146 004560 104001          TST      @TACS              ;SAMPLE THE "ERROR" BIT
1147 004562          BPL      758              ;BR IF "ERROR" = 0
1148 004562 112714 000006      ERROR      1          ;"ERROR" NOT = 0
1149 004566 005714          758:      MOVB      @BSBG,@TACS      ;CHECK "ERROR" WITH "BSBG"
1150 004570 100001          TST      @TACS              ;SAMPLE THE "ERROR" BIT
1151 004572 104001          BPL      768              ;BR IF "ERROR" = 0
1152 004574          ERROR      1          ;"ERROR" NOT = 0
1153 004574 112714 000010      768:      MOVB      @SFFG,@TACS      ;CHECK "ERROR" WITH "SFFG"
1154 004600 005714          TST      @TACS              ;SAMPLE THE "ERROR" BIT
1155 004602 100001          BPL      778              ;BR IF "ERROR" = 0
1156 004604 104001          ERROR      1          ;"ERROR" NOT = 0
1157 004606          778:      MOVB      @SFBG,@TACS      ;CHECK "ERROR" WITH "SFBG"
1158 004606 1127.4 000012      TST      @TACS              ;SAMPLE THE "ERROR" BIT
1159 004612 005714          BPL      788              ;BR IF "ERROR" = 0
1160 004614 100001          ERROR      1          ;"ERROR" NOT = 0
1161 004616 104001          788:      MOVB      @REWIND,@TACS      ;CHECK "ERROR" WITH "REWIND"
1162 004620          TST      @TACS              ;SAMPLE THE "ERROR" BIT
1163 004620 112714 000014      BPL      798              ;BR IF "ERROR" = 0
1164 004624 005714          ERROR      1          ;"ERROR" NOT = 0
1165 004626 100001          798:      MOVB      @REWIND,@TACS      ;CHECK "ERROR" WITH "REWIND"
1166 004630 104001          TST      @TACS              ;SAMPLE THE "ERROR" BIT
1167 004632          BPL      798              ;BR IF "ERROR" = 0
1168 004632 112714 000016      ERROR      1          ;"ERROR" NOT = 0
1169 004636 005714          798:      MOVB      @REWIND,@TACS      ;CHECK "ERROR" WITH "REWIND"
1170 004640 100001          TST      @TACS              ;SAMPLE THE "ERROR" BIT
1171 004642 104001          BPL      798              ;BR IF "ERROR" = 0
1172 004644          ERROR      1          ;"ERROR" NOT = 0
1173          ;*****
1174          ;*TEST 14      TEST "WRITE LOCK" WITH FILE GAP=1
1175          ;*****
1176 004644 000004      TST14:  SCOPE
1177 004646 012767 004726 174232      MOV      @16,@LPADR      ;;SET SCOPE LOOP ADDRESS
1178 004654 012767 005206 174306      MOV      @TST15,@ESCAPE  ;;ESCAPE TO TEST 15 ON ERROR
1179 004662 000005          RESET                  ;CLEAR THE INTERFACE
1180 004664 010314          MOV      @DRIVE,@TACS      ;SELECT @TACS
1181 004666 112714 000017      MOVB     @REWIND!GO,@TACS
1182 004672 104412          WAITREADY
1183 004674 112714 000005      MOVB     @READ!GO,@TACS      ;START A READ
1184 004700 104413          WAITXFER
1185 004702 052714 000020      BIS      @ILBS,@TACS      ;SHUT DOWN
1186 004706 104412          WAITREADY
1187 004710 112714 000013      MOVB     @SFFG!GO,@TACS      ;GO TO THE FILE GAP
1188 004714 104412          WAITREADY              ;WAIT ON "READY"
1189 004716 032714 004000      BIT      @FGAP,@TACS      ;CHECK FOR FILE GAP
1190 004722 001001          BNE      18              ;BR IF IN A GAP
1191 004724 104001          ERROR      1          ;FILE GAP = 0
    
```

m2

m3

1192	004726			16:			
1193	004726	112714	000000		MOVB	%WFG,%TACS	;LOAD A "WFG"
1194	004732	032714	010000		BIT	%WRTLOCK,%TACS	;IS "WRITE LOCK ERROR" = "1"
1195	004736	001001			BNE	648	;BR IF YES
1196	004740	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "1"
1197	004742			648:			
1198	004742	112714	000002		MOVB	%WRITE,%TACS	;LOAD A "WRITE"
1199	004746	032714	010000		BIT	%WRTLOCK,%TACS	;IS "WRITE LOCK ERROR" = "1"
1200	004752	001001			BNE	658	;BR IF YES
1201	004754	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "1"
1202	004756			658:			
1203	004756	112714	000004		MOVB	%READ,%TACS	;LOAD A "READ"
1204	004762	032714	010000		BIT	%WRTLOCK,%TACS	;IS "WRITE LOCK ERROR" = "0"
1205	004766	001401			BEQ	668	;BR IF YES
1206	004770	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1207	004772			668:			
1208	004772	112714	000006		MOVB	%BSFG,%TACS	;LOAD A "BSFG"
1209	004776	032714	010000		BIT	%WRTLOCK,%TACS	;IS "WRITE LOCK ERROR" = "0"
1210	005002	001401			BEQ	678	;BR IF YES
1211	005004	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1212	005006			678:			
1213	005006	112714	000010		MOVB	%BSBG,%TACS	;LOAD A "BSBG"
1214	005012	032714	010000		BIT	%WRTLOCK,%TACS	;IS "WRITE LOCK ERROR" = "0"
1215	005016	001401			BEQ	688	;BR IF YES
1216	005020	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1217	005022			688:			
1218	005022	112714	000012		MOVB	%SFSG,%TACS	;LOAD A "SFSG"
1219	005026	032714	010000		BIT	%WRTLOCK,%TACS	;IS "WRITE LOCK ERROR" = "0"
1220	005032	001401			BEQ	698	;BR IF YES
1221	005034	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1222	005036			698:			
1223	005036	112714	000014		MOVB	%SFBG,%TACS	;LOAD A "SFBG"
1224	005042	032714	010000		BIT	%WRTLOCK,%TACS	;IS "WRITE LOCK ERROR" = "0"
1225	005046	001401			BEQ	708	;BR IF YES
1226	005050	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1227	005052			708:			
1228	005052	112714	000016		MOVB	%REWIND,%TACS	;LOAD A "REWIND"
1229	005056	032714	010000		BIT	%WRTLOCK,%TACS	;IS "WRITE LOCK ERROR" = "0"
1230	005062	001401			BEQ	718	;BR IF YES
1231	005064	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1232	005066			718:			
1233	005066	112714	000000		MOVB	%WFG,%TACS	;CHECK "ERROR" WITH "WFG"
1234	005072	005714			TST	%TACS	;SAMPLE THE "ERROR" BIT
1235	005074	100401			BMI	728	;BR IF "ERROR" = 1
1236	005076	104001			ERROR	1	; "ERROR" NOT = 1
1237	005100			728:			
1238	005100	112714	000002		MOVB	%WRITE,%TACS	;CHECK "ERROR" WITH "WRITE"
1239	005104	005714			TST	%TACS	;SAMPLE THE "ERROR" BIT
1240	005106	100401			BMI	738	;BR IF "ERROR" = 1
1241	005110	104001			ERROR	1	; "ERROR" NOT = 1
1242	005112			738:			
1243	005112	112714	000004		MOVB	%READ,%TACS	;CHECK "ERROR" WITH "READ"
1244	005116	005714			TST	%TACS	;SAMPLE THE "ERROR" BIT
1245	005120	100401			BMI	748	;BR IF "ERROR" = 1
1246	005122	104001			ERROR	1	; "ERROR" NOT = 1
1247	005124			748:			

```

1248 005124 112714 000006      MOVB 0BSFG,0TACS      ;CHECK "ERROR" WITH "BSFG"
1249 005130 005714              TST 0TACS             ;SAMPLE THE "ERROR" BIT
1250 005132 100001              BPL 750               ;BR IF "ERROR" = 0
1251 005134 104001              ERROR 1               ;"ERROR" NOT = 0
1252 005136                    750:
1253 005136 112714 000010      MOVB 0BSBG,0TACS      ;CHECK "ERROR" WITH "BSBG"
1254 005142 005714              TST 0TACS             ;SAMPLE THE "ERROR" BIT
1255 005144 100001              BPL 760               ;BR IF "ERROR" = 0
1256 005146 104001              ERROR 1               ;"ERROR" NOT = 0
1257 005150                    760:
1258 005150 112714 000012      MOVB 0SFPG,0TACS      ;CHECK "ERROR" WITH "SFPG"
1259 005154 005714              TST 0TACS             ;SAMPLE THE "ERROR" BIT
1260 005156 100001              BPL 770               ;BR IF "ERROR" = 0
1261 005160 104001              ERROR 1               ;"ERROR" NOT = 0
1262 005162                    770:
1263 005162 112714 000014      MOVB 0SFBG,0TACS      ;CHECK "ERROR" WITH "SFBG"
1264 005166 005714              TST 0TACS             ;SAMPLE THE "ERROR" BIT
1265 005170 100401              BMI 780               ;BR IF "ERROR" = 1
1266 005172 104001              ERROR 1               ;"ERROR" NOT = 1
1267 005174                    780:
1268 005174 112714 000016      MOVB 0REWIND,0TACS    ;CHECK "ERROR" WITH "REWIND"
1269 005200 005714              TST 0TACS             ;SAMPLE THE "ERROR" BIT
1270 005202 100001              BPL 790               ;BR IF "ERROR" = 0
1271 005204 104001              ERROR 1               ;"ERROR" NOT = 0
1272 005206                    790:
1273                                ;*****
1274                                ;*TEST 15      TRY WRITING ON TAPE WHEN "WRITE LOCKED"
1275                                ;*****
1276 005206 000004      TST15: SCOPE
1277 005210 012767 000001 173750  MOV 01,0TIMES      ;;DO 1 ITERATION
1278 005216 012767 005266 173662  MOV 030,0LPADR     ;;SET SCOPE LOOP ADDRESS
1279 005224 012767 005362 173736  MOV 0TST16,0ESCAPE ;;ESCAPE TO TEST 16 ON ERROR
1280 005232 000005      RESET              ;CLEAR ALL
1281 005234 010314      MOV DRIVE,0TACS    ;SELECT DRIVE
1282 005236 112714 000017      MOVB 0REWIND!GO,0TACS ;GO TO BOT
1283 005242 104412      WAITREADY          ;WAIT ON READY FLAG
1284 005244 112714 000005 18:      MOVB 0READ!GO,0TACS ;START A READ
1285 005250 104413      WAITXFER           ;WAIT ON "XFER REQ"
1286 005252 052714 000020      BIS 0ILBS,0TACS     ;SHUT DOWN
1287 005256 104412      WAITREADY          ;WAIT ON READY
1288 005260 112714 000015      MOVB 0SFBG!GO,0TACS ;SPACE OVER THE 2ND BLOCK
1289 005264 104412      WAITREADY          ;WAIT ON READY FLAG
1290 005266 000005 38:      RESET              ;CLEAR ALL FLAGS
1291 005270 010314      MOV DRIVE,0TACS    ;SELECT DRIVE
1292 005272 104412      WAITREADY          ;WAIT ON READY
1293 005274 112714 000003      MOVB 0WRITE!GO,0TACS ;START A "WRITE"
1294 005300 005001      CLR R1              ;TIMER
1295 005302 105201 48:      INCB R1              ;COUNT THE COUNTER GIVING
1296 005304 001376      BNE 48              ; "XFER REQ" TIME TO SET IF IT
1297                                ; IS GOING TO
1298 005306 105714      TSTB 0TACS          ;CHECK "XFER REQ"
1299 005310 100001      BPL 50              ;BR IF IT DIDN'T SET
1300 005312 104001      ERROR 1             ;"XFER REQ" SET
1301 005314 104412 58:      WAITREADY          ;WAIT ON READY
1302 005316 005714      TST 0TACS           ;CHECK ERROR FLAG
1303 005320 100401      BMI 60              ;BR IF IT IS SET
    
```

```

1304 005322 104001          ERROR 1          ;ERROR ISN'T SET
1305 005324 112714 000017 68:  MOVB  $REWIND!GO,@TACS ;GO TO BOT
1306 005330 104412          WAITREADY      ;WAIT ON READY FLAG
1307 005332 012701 000002          MOV  02,R1      ;GET OVER THE TWO DATA BLOCKS
1308 005336 112714 000005 78:  MOVB  $READ!GO,@TACS ;START A READ
1309 005342 005301          DEC  R1         ;COUNT THIS BLOCK
1310 005344 002405          BLT  00         ;BR IF OVER THE 2ND BLOCK?
1311 005346 104413          WAITXFER      ;WAIT ON "XFER REQ"
1312 005350 052714 000020          BIS  0ILBS,@TACS ;SHUT DOWN
1313 005354 104412          WAITREADY      ;WAIT ON READY
1314 005356 000767          BR   78        ;GO DO ANOTHER BLOCK
1315 005360 104412 08:  WAITREADY      ;IF "XFER REQ" COMES UP THERE
1316                                     ; IS MORE THAN TWO BLOCKS ON TAPE
1317                                     ; "WRITE" CAUSED DATA TO BE WRITTEN ON TAP
1318                                     ;
1319                                     ;*****
1320 ;*TEST 16 TRY "WFG" WHEN WRITE LOCKED
1321 ;*****
1321 005362 000004 TST16: SCOPE
1322 005364 012767 000001 173574 MOV  01,@TIMES ;DO 1 ITERATION
1323 005372 012767 005500 173570 MOV  0TST17,@ESCAPE ;ESCAPE TO TEST 17 ON ERROR
1324 005400 000005          RESET
1325 005402 010314          MOV  DRIVE,@TACS ;SELECT DRIVE
1326 005404 112714 000017          MOVB  $REWIND!GO,@TACS ;GO TO BOT
1327 005410 104412          WAITREADY      ;WAIT ON READY
1328 005412 112714 000005          MOVB  $READ!GO,@TACS ;GET PAST THE FIRST BLOCK
1329 005416 104413          WAITXFER      ;WAIT ON TRANSFER REQ.
1330 005420 052714 000020          BIS  0ILBS,@TACS ;SHUT DOWN
1331 005424 104412          WAITREADY      ;WAIT ON READY
1332 005426 000005          RESET          ;CLEAR THE WORLD
1333 005430 010314          MOV  DRIVE,@TACS ;SELECT DRIVE
1334 005432 104412          WAITREADY      ;WAIT ON THE READY FLAG
1335 005434 112714 000001          MOVB  $WFG!GO,@TACS ;START A "WFG"
1336 005440 104412          WAITREADY      ;WAIT ON READY
1337 005442 112714 000017          MOVB  $REWIND!GO,@TACS ;GO TO BOT
1338 005446 104412          WAITREADY
1339 005450 112714 000005          MOVB  $READ!GO,@TACS ;GET PAST 1ST BLOCK
1340 005454 104413          WAITXFER
1341 005456 052714 000020          BIS  0ILBS,@TACS ;SHUT DOWN
1342 005462 104412          WAITREADY
1343 005464 112714 000005          MOVB  $READ!GO,@TACS ;SEE IF 2ND BLOCK IS STILL ON TAPE
1344 005470 104413          WAITXFER      ;IF READY COMES UP THE "WFG"
1345                                     ; COMMAND WROTE ON THE TAPE
1346 005472 052714 000020          BIS  0ILBS,@TACS ;SHUT DOWN
1347 005476 104412          WAITREADY      ;WAIT ON READY
1348                                     ;*****
1349 ;*TEST 17 TEST "WFG" ON CLEAR LEADER AND WRITE LOCKED
1350 ;*****
1351 005500 000004 TST17: SCOPE
1352 005502 012767 000001 173456 MOV  01,@TIMES ;DO 1 ITERATION
1353 005510 012767 005532 173370 MOV  010,@LPADR ;SET SCOPE LOOP ADDRESS
1354 005516 012767 005564 173444 MOV  0TST20,@ESCAPE ;ESCAPE TO TEST 20 ON ERROR
1355 005524 000005          RESET
1356 005526 010314          MOV  DRIVE,@TACS ;SELECT DRIVE
1357 005530 104412          WAITREADY      ;WAIT ON THE READY FLAG
1358 005532 112714 000017 18:  MOVB  $REWIND!GO,@TACS ;GO TO BOT
1359 005536 104412          WAITREADY
    
```

```

1360 005540 112714 000001      MOVB    @WFG!GO,@TACS      ;START A "WFG"
1361 005544 104412      WAITREADY                    ;WAIT FOR READY
1362 005546 005714      TST     @TACS                ;CHECK ERROR
1363 005550 100401      BMI     20                    ;BR IF ERROR=1
1364 005552 104001      ERROR  1                      ;ERROR ISN'T ON AN ONE
1365 005554 032714 020000 20:   BIT     @LEADER,@TACS      ;CHECK IF STILL ON CLEAR LEADER
1366 005560 001001      BNE     TST20                 ;GO TO THE NEXT TEST IF IT IS
1367 005562 104001      ERROR  1                      ;NOT ON CLEAR LEADER
1368                                     ;*****
1369                                     ;*TEST 20      TEST "WRITE" WITH WRITE LOCK ON AND AT CLEAR LEADER
1370                                     ;*****
1371 005564 000004      TST20: SCOPE
1372 005566 012767 000001 173372   MOV     @1,@TIMES           ;;DO 1 ITERATION
1373 005574 012767 005610 173304   MOV     @10,@LPADR          ;;SET SCOPE LOOP ADDRESS
1374 005602 012767 005642 173360   MOV     @TST21,@ESCAPE     ;;ESCAPE TO TEST 21 ON ERROR
1375 005610 000005 10:   RESET                    ;CLEAR THE WORLD
1376 005612 010314      MOV     DRIVE,@TACS         ;SELECT THE DRIVE
1377 005614 104412      WAITREADY                    ;WAIT ON READY
1378 005616 112714 000017      MOVB    @REWIND!GO,@TACS    ;GO TO BOT
1379 005622 104412      WAITREADY                    ;WAIT TILL READY = 1
1380 005624 112714 000003      MOVB    @WRITE!GO,@TACS     ;TRY TO START A WRITE
1381 005630 104412      WAITREADY                    ;IF TRANSFER REQ. COMES UP
1382                                     ;THERE IS A PROBLEM WITH WRITE LOCK
1383 005632 032714 020000      BIT     @LEADER,@TACS      ;MAKE SURE STILL A CLEAR LEADER
1384 005636 001001      BNE     TST21                 ;BR IF AT CLEAR LEADER
1385 005640 104001      ERROR  1                      ;TAPE MOVED OFF OF CLEAR LEADER
1386                                     ;*****
1387                                     ;*TEST 21      TEST "WRITE ENABLE"
1388                                     ;*****
1389 005642 000004      TST21: SCOPE
1390 005644 012767 000012 173314   MOV     @10,@TIMES           ;;DO 10; ITERATIONS
1391 005652 012767 005670 173226   MOV     @10,@LPADR          ;;SET SCOPE LOOP ADDRESS
1392 005660 004037 007436      JSR     R0,@ASKQUES
1393 005664 013040      MLAST
1394 005666 000240      NOP
1395 005670 000005 10:   RESET
1396 005672 010314      MOV     DRIVE,@TACS         ;SELECT DRIVE
1397 005674 112714 000017      MOVB    @REWIND!GO,@TACS    ;GO TO BOT
1398 005700 104412      WAITREADY                    ;WAIT ON READY
1399 005702 112714 000000      MOVB    @WFG,@TACS          ;LOAD A "WFG" COMMAND
1400 005706 032714 011000      BIT     @WRTLOCK!OFFLINE,@TACS ;MAKE SURE THE CASSETTE IS
1401 005712 001401      BEQ     TST22                 ; "WRITE ENABLED" AND "ON LINE"
1402 005714 104001      ERROR  1
1403                                     ;*****
1404                                     ;*TEST 22      END OF TEST CODE
1405                                     ;*****
1406 005716 000004      TST22: SCOPE
1407 005720 012767 000001 173240   MOV     @1,@TIMES           ;;DO 1 ITERATION
1408 005726 000005      RESET
1409 005730 010314      MOV     DRIVE,@TACS         ;SELECT DRIVE
1410 005732 112714 000017      MOVB    @REWIND!GO,@TACS    ;GO TO BOT
1411 005736 104412      WAITREADY
1412 .SBTTL  END OF PASS ROUTINE
1413
1414                                     ;*****
1415                                     ;*INCREMENT THE PASS NUMBER (@PASS)
    
```

DA

```

1416 ;*TYPE "END PASS"
1417 ;*IF THERES A MONITOR GO TO IT
1418 ;*IF THERE ISN'T JUMP TO START
1419 ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
1420 ;*SENDMG CAN BE CHANGED TO 7.
1421
1422 005740 ;SEOP:
1423 005740 000004 SCOPE
1424 005742 005067 173134 CLR STSTN ;;ZERO THE TEST NUMBER
1425 005746 005067 173214 CLR STIMES ;;ZERO THE NUMBER OF ITERATIONS
1426 005752 005267 173122 INC SPASS ;;INCREMENT THE PASS NUMBER
1427 005756 042767 100000 173114 BIC 0100000,SPASS ;;DON'T ALLOW A NEG. NUMBER
1428 005764 005327 DEC (PC)+ ;;LOOP?
1429 005766 000001 ;SEOPCT: .WORD 1
1430 005770 003015 BGT SDOAGN ;;YES
1431 005772 012737 MOV (PC)+,0(PC)+ ;;RESTORE COUNTER
1432 005774 000001 ;SENDCT: .WORD 1
1433 005776 005766 ;SEOPCT
1434 006000 104401 006033 TYPE ,SENDMG ;;TYPE "END PASS"
1435 006004 013700 000042 ;GET42: MOV 0042,R0 ;;GET MONITOR ADDRESS
1436 006010 001405 BEQ SDOAGN ;;BRANCH IF NO MONITOR
1437 006012 000005 RESET ;;CLEAR THE WORLD
1438 006014 004710 ;SENDAD: JSR PC,(R0) ;;GO TO MONITOR
1439 006016 000240 NOP ;;SAVE ROOM
1440 006020 000240 NOP ;;FOR
1441 006022 000240 NOP ;;ACT11
1442 006024 ;SDOAGN:
1443 006024 000137 JMP 0(PC)+ ;;RETURN
1444 006026 002204 ;RTNAD: .WORD START
1445 006030 377 377 000 ;ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
1446 006033 015 042412 042116 ;SENDMG: .ASCIZ <15><12>/END PASS/
1447 006040 050040 051501 000123

```

```

1448 .SBTTL SCOPE HANDLER ROUTINE
1449
1450 ;*****
1451 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1452 ;*AND LOAD THE TEST NUMBER(=STSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1453 ;*AND LOAD THE ERROR FLAG (=SERFLG) INTO DISPLAY<15:00>
1454 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1455 ;*SW14=1 LOOP ON TEST
1456 ;*SW11=1 INHIBIT ITERATIONS
1457 ;*SW09=1 LOOP ON ERROR
1458 ;*SW08=1 LOOP ON TEST IN SWR<7:0>
1459 ;*CALL
1460 ;* SCOPE ;;SCOPE=IOT
1461
1462 006046 8SCOPE:
1463 006046 104406 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
1464 006050 032777 040000 173062 18: BIT 0BIT14,0SWR ;;LOOP ON PRESENT TEST?
1465 006056 001111 BNE 0OVER ;;YES IF SW14=1
1466 ;*****START OF CODE FOR THE XOR TESTER*****
1467 006060 000416 8XTSTR: BR 68 ;;IF RUNNING ON THE "XOR" TESTER CHANGE
1468 ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
1469 006062 013746 000004 MOV 00ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
1470 006066 012737 006106 000004 MOV 050,00ERRVEC ;;SET FOR TIMEOUT
1471 006074 005737 177060 TST 00177060 ;;TIME OUT ON XOR?
1472 006100 012637 000004 MOV (SP)+,00ERRVEC ;;RESTORE THE ERROR VECTOR
1473 006104 000463 BR 8SVLAD ;;GO TO THE NEXT TEST
1474 006106 022626 58: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
1475 006110 012637 000004 MOV (SP)+,00ERRVEC ;;RESTORE THE ERROR VECTOR
1476 006114 000423 BR 78 ;;LOOP ON THE PRESENT TEST
1477 006116 68:;*****END OF CODE FOR THE XOR TESTER*****
1478 006116 032777 000400 173014 BIT 0BIT00,0SWR ;;LOOP ON SPEC. TEST?
1479 006124 001404 BEQ 28 ;;BR IF NO
1480 006126 127767 173006 172746 CMPB 0SWR,0STSTNM ;;ON THE RIGHT TEST? SWR<7:0>
1481 006134 001462 BEQ 0OVER ;;BR IF YES
1482 006136 105767 172741 28: TSTB 0SERFLG ;;HAS AN ERROR OCCURRED?
1483 006142 001421 BEQ 38 ;;BR IF NO
1484 006144 126767 172745 172731 CMPB 0SERMAX,0SERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
1485 006152 101015 BHI 38 ;;BR IF NO
1486 006154 032777 001000 172756 BIT 0BIT09,0SWR ;;LOOP ON ERROR?
1487 006162 001404 BEQ 48 ;;BR IF NO
1488 006164 016767 172720 172714 78: MOV 0LPERR,0LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
1489 006172 000443 BR 0OVER
1490 006174 105067 172703 48: CLRB 0SERFLG ;;ZERO THE ERROR FLAG
1491 006200 005067 172762 CLR 0TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
1492 006204 000415 BR 18 ;;ESCAPE TO THE NEXT TEST
1493 006206 032777 004000 172724 38: BIT 0BIT11,0SWR ;;INHIBIT ITERATIONS?
1494 006214 001011 BNE 18 ;;BR IF YES
1495 006216 005767 172656 TST 0PASS ;;IF FIRST PASS OF PROGRAM
1496 006222 001406 BEQ 18 ;; INHIBIT ITERATIONS
1497 006224 005267 172654 INC 0ICNT ;;INCREMENT ITERATION COUNT
1498 006230 026767 172732 172646 CMP 0TIMES,0ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
1499 006236 002021 BGE 0OVER ;;BR IF MORE ITERATION REQUIRED
1500 006240 012767 000001 172636 18: MOV 01,0ICNT ;;REINITIALIZE THE ITERATION COUNTER
1501 006246 016767 000044 172712 MOV 0MXCNT,0TIMES ;;SET NUMBER OF ITERATIONS TO DO
1502 006254 105267 172622 8SVLAD: INCB 0STSTNM ;;COUNT TEST NUMBERS
1503 006260 011667 172622 MOV (SP),0LPADR ;;SAVE SCOPE LOOP ADDRESS
    
```

1504	006264	011667	172620		NOV	(SP),@LPERR	;;SAVE ERROR LOOP ADDRESS
1505	006270	005067	172674		CLR	@ESCAPE	;;CLEAR THE ESCAPE FROM ERROR ADDRESS
1506	006274	112767	000001	172613	NOVB	@1,@ERNAX	;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1507	006302	016777	172574	172632	SOVER: NOV	@STNH,@DISPLAY	;;DISPLAY TEST NUMBER
1508	006310	016716	172572		NOV	@LPADR,(SP)	;;FUDGE RETURN ADDRESS
1509	006314	000002			RTI		;;FIXES PS
1510	006316	003720			SNXCNT: 2000.		;;MAX. NUMBER OF ITERATIONS

B

```

1511 .SBTTL ERROR HANDLER ROUTINE
1512
1513 ;;;;;;;;;;;;;;
1514 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
1515 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
1516 ;*AND GO TO TYPERR ON ERROR
1517 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1518 ;*SW15=1 HALT ON ERROR
1519 ;*SW13=1 INHIBIT ERROR TYPEOUTS
1520 ;*SW10=1 BELL ON ERROR
1521 ;*SW09=1 LOOP ON ERROR
1522 ;*CALL
1523 ;* ERROR N ;;ERROR=ENT AND N=ERROR ITEM NUMBER
1524
1525 006320 ;ERROR:
1526 006320 104406 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
1527 006322 011437 001162 MOV 0TACS,000REG0 ;;SAVE THE STATUS REG.
1528 006326 011537 001164 MOV 0TADB,000REG1 ;;SAVE THE DATA BUFFER
1529 006332 010037 001124 MOV R0,000GDDAT ;R0 WILL CONTAIN THE GOOD DATA
1530 006336 010137 001126 MOV R1,000BDDAT ;R1 WILL CONTAIN THE BAD DATA
1531 006342 105267 172535 70: INCB 0ERFLG ;;SET THE ERROR FLAG
1532 006346 001775 BEQ 70 ;;DON'T LET THE FLAG GO TO ZERO
1533 006350 016777 172526 172564 MOV 0TSTNM,0DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
1534 006356 032777 002000 172554 BIT 0BIT10,0SWR ;;BELL ON ERROR?
1535 006364 001402 BEQ 10 ;;NO - SKIP
1536 006366 104401 001172 TYPE ,0BELL ;;RING BELL
1537 006372 005267 172514 10: INC 0ERTTL ;;COUNT THE NUMBER OF ERRORS
1538 006376 011667 172514 MOV (0SP),0ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
1539 006402 162767 000002 172506 SUB 02,0ERRPC
1540 006410 117767 172502 172476 MOVB 00ERRPC,0ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
1541 006416 032777 020000 172514 BIT 0BIT13,0SWR ;;SKIP TYPEOUT IF SET
1542 006424 001004 BNE 200 ;;SKIP TYPEOUTS
1543 006426 004767 000060 JSR PC,TYPERR ;;GO TO USER ERROR ROUTINE
1544 006432 104401 001177 TYPE ,0CRLF
1545 006436 200:
1546 006436 005777 172476 20: TST 0SWR ;;HALT ON ERROR
1547 006442 100002 BPL 30 ;;SKIP IF CONTINUE
1548 006444 000000 HALT ;;HALT ON ERROR!
1549 006446 104406 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
1550 006450 032777 001000 172462 30: BIT 0BIT09,0SWR ;;LOOP ON ERROR SWITCH SET?
1551 006456 001402 BEQ 40 ;;BR IF NO
1552 006460 016716 172424 MOV 0LPERR,(0SP) ;;FUDGE RETURN FOR LOOPING
1553 006464 005767 172500 40: TST 0ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
1554 006470 001402 BEQ 50 ;;BR IF NONE
1555 006472 016716 172472 MOV 0ESCAPE,(0SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
1556 006476 50:
1557 006476 022737 006014 000042 CMP 00ENDAD,0042 ;;ACT-11 AUTO-ACCEPT?
1558 006504 00100: BNE 60 ;;BRANCH IF NO
1559 006506 000000 HALT ;;YES
1560 006510 60:
1561 006510 000002 RTI ;;RETURN
1562
  
```

```

1563                                     ;,;*****
1564                                     ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
1565
1566 006512 104401 001177 TYPERR: TYPE ,SCRLF ;TYPE A CARRIAGE RETURN & LINE FEED
1567 006516 010046      MOV      RB,-(SP) ;SAVE RB
1568 006520 113700 001114      MOVB   000ITENB,RB ;PICKUP THE ITEM INDEX
1569 006524 005300      DEC      RB ;ADJUST THE INDEX
1570 006526 006300      ASL     RB ;SO IT WILL WORK FOR
1571 006530 006300      ASL     RB ;THE ERROR TABLE
1572 006532 006300      ASL     RB
1573 006534 062700 001236      ADD     00ERRTB,RB ;FORM THE TABLE POINTER
1574 006540 012067 000002      MOV     (RB)+,10 ;PICKUP "ERROR MESSAGE" POINTER
1575 006544 104401      TYPE    ;TYPE "ERROR MESSAGE"
1576 006546 000000      10:    0 ;"ERROR MESSAGE POINTER" GOES HERE
1577 006550 104401 001177      TYPE    ,SCRLF
1578 006554 012067 000004      MOV     (RB)+,20 ;PICKUP "DATA HEADER" POINTER
1579 006560 001404      BEQ    30 ;IF "0" DON'T TYPE
1580 006562 104401      TYPE    ;TYPE "DATA HEADER"
1581 006564 000000      20:    0 ;"DATA HEADER" POINTER GOES HERE
1582 006566 104401 001177      TYPE    ,SCRLF
1583 006572 012000      30:    MOV     (RB)+,RB ;PICKUP "DATA POINTER"
1584 006574 001004      BNE    50 ;IF THERE IS DATA TO TYPE GO DO IT
1585 006576 012600      40:    MOV     (SP)+,RB ;RESTORE RB
1586 006600 104401 001177      TYPE    ,SCRLF ;TYPE A CARRAGE RETURN&LINE FEED
1587 006604 000207      RTS     PC ;RETURN
1588 006606
1589 006606 013046      50:    MOV     0(RB)+,-(SP) ;;SAVE 0(RB)+ FOR TYPEOUT
1590                                     ;;TYPE DATA
1591 006610 104402      TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1592 006612 005710      TST     (RB) ;TERMINATOR?
1593 006614 001770      BEQ    40 ;BR IF YES
1594 006616 104401 006624      TYPE    ,60 ;TYPE 2 SPACES
1595 006622 000771      BR     50 ;LOOP
1596 006624 020040 000      60:    .ASCIZ  / / ;ASCII STRING OF 2 SPACES
1597 006630      .EVEN
  
```

```

1598                                     ;;*****
1599                                     ;ROUTINE TO WAIT ON THE READY BIT
1600
1601 006630                                WAIT,ON,READY:
1602 006630 005067 000044                CLR     WAIT2
1603 006634 016767 000072 000044        MOV     MAXCNT,HGHTIN
1604 006642 012637 001202                MOV     (SP)+,00SAVPC
1605 006646 162737 000002 001202        SUB     #2,00SAVPC
1606 006654 012637 001204                MOV     (SP)+,00SAVPS
1607 006660 032714 000040                WAIT1: BIT  0READY,0TACS
1608 006664 001013                        BNE     WAIT3
1609 006666 105714                        TSTB   0TACS
1610 006670 100002                        BPL     WAIT4
1611 006672 104004                        ERROR  4
1612 006674 000407                        BR      WAIT3
1613 006676 005227                WAIT4: INC  (PC)+
1614 006700 000000                WAIT2: 0
1615 006702 001366                        BNE     WAIT1
1616 006704 005327                        DEC     (PC)+
1617 006706 000000                HGHTIN: 0
1618 006710 003363                        BGT     WAIT1
1619 006712 104002                        ERROR  2
1620 006714 013746 001204                WAIT3: MOV  00SAVPS,-(SP)
1621 006720 013746 001202                MOV  00SAVPC,-(SP)
1622 006724 062716 000002                ADD  #2,(SP)
1623 006730 000002                        RTI
1624 006732 000000                MAXCNT: 0
1625
1626                                     ;;*****
1627                                     ;ROUTINE TO WAIT ON TRANSFER REQUEST
1628
1629 006734                                WAIT,FOR,XFER,REQ:
1630 006734 005067 000044                CLR     20
1631 006740 013767 006732 000044        MOV     00MAXCNT,30
1632 006746 012637 001202                MOV     (SP)+,00SAVPC
1633 006752 162737 000002 001202        SUB     #2,00SAVPC
1634 006760 012637 001204                MOV     (SP)+,00SAVPS
1635 006764 105714                10:   TSTB   0TACS
1636 006766 100414                        BNI     40
1637 006770 032714 000040                BIT  0READY,0TACS
1638 006774 001402                        BEQ     50
1639 006776 104004                        ERROR  4
1640 007000 000407                        BR      40
1641 007002 005227                50:   INC  (PC)+
1642 007004 000000                20:   0
1643 007006 001366                        BNE     10
1644 007010 005327                        DEC     (PC)+
1645 007012 000000                30:   0
1646 007014 003363                        BGT     10
1647 007016 104003                        ERROR  3
1648 007020 013746 001204                40:   MOV  00SAVPS,-(SP)
1649 007024 013746 001202                MOV  00SAVPC,-(SP)
1650 007030 062716 000002                ADD  #2,(SP)
1651 007034 000002                        RTI

```

```

1652 ;,*****
1653
1654 .SBTTL          ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
1655
1656 ;CALL
1657 ;             JSR      PC,00ASKDRV
1658 ;             RETURN
1659 ;             ;NOTE: R0 AND R1 ARE DESTROYED
1660 007036 104401 012552 ASKDRV: TYPE      ,MSGDRV          ;<CR LF>"DRIVE(S)? "
1661 007042 005067 172156 CLR          DRVKEY
1662 007046 104410 RDLIN
1663 007050 012600 MOV          (SP)+,R0          ;GO GET A DRIVE
1664 007052 105710 TSTB        0R0              ;SETUP TO CHECK FOR VALID DRIVE(S)
1665 007054 001425 BEQ          NOTLGL          ;WAS A DRIVE SELECTED?
1666 007056 012701 001224 MOV          @DRVKEY,R1       ;BR IF NO
1667 007062 122710 000101 LOOP:  CNPB        #'A,0R0          ;WAS DRIVE "A" SELECTED?
1668 007066 001002 BNE          NOTA           ;BR IF NO
1669 007070 112021 MOVB        (R0)+,(R1)+     ;SET KEY FOR DRIVE "A"
1670 007072 000411 BR           NEXT
1671 007074 122710 000102 NOTA:  CNPB        #'B,0R0          ;WAS DRIVE "B" SELECTED?
1672 007100 001002 BNE          NOTB           ;BR IF NO
1673 007102 112021 MOVB        (R0)+,(R1)+     ;SET KEY FOR DRIVE "B"
1674 007104 000404 BR           NEXT
1675 007106 122710 000054 NOTB:  CNPB        #54,0R0       ;WAS A COMMA TYPED?
1676 007112 001006 BNE          NOTLGL          ;BR IF NO
1677 007114 105720 TSTB        (R0)+           ;DUMP THE COMMA
1678 007116 105710 NEXT:  TSTB        0R0              ;TERMINATOR?
1679 007120 001406 BEQ          EXIT           ;BR IF YES
1680 007122 022701 001226 CNP          @DRVKEY+2,R1     ;TWO DRIVES SELECTED?
1681 007126 101755 BHI          LOOP
1682 007130 104401 001176 NOTLGL: TYPE      ,SQUES
1683 007134 000740 BR           ASKDRV
1684 007136 005767 172062 EXIT:  TST        DRVKEY
1685 007142 001772 BEQ          NOTLGL
1686 007144 000207 RTS          PC
1687

```

```

1688                                     ;;*****
1689                                     ;CALL
1690                                     ;JSR      PC,00ASKADR
1691
1692 007146 010046 ASKADR: MOV      R0,-(SP)          ;SAVE R0
1693 007150 104401 012566 18: TYPE      ,MSGASK          ;"TACS?"
1694 007154 104411 RDOCT          ;GET VALUE
1695 007156 012600 MOV      (SP)+,R0          ;PICK UP THE OCTAL NUMBER
1696 007160 001423 BEQ      38              ;IF "0" USE OLD VALUES
1697 007162 020027 160000 CMP      R0,0160000        ;MAKE SURE IT IS A BUS ADDRESS
1698 007166 103770 BLO      18
1699 007170 010037 001206 MOV      R0,00TACSL        ;SAVE TOE TACS
1700 007174 062700 000002 ADD      #2,R0            ;STEP TO TADB ADDRESS
1701 007200 010037 001212 MOV      R0,00TADBL        ;AND SAVE IT
1702 007204 013737 001206 001210 MOV      00TACSL,00TACSH  ;SET UP TACS UPPER
1703 007212 005237 001210 INC      00TACSH          ;BYTE POINTER
1704
1705 007216 013737 001212 001214 MOV      00TADBL,00TADBH  ;SET UP TADB UPPER
1706 007224 005237 001214 INC      00TADBH          ;BYTE POINTER
1707 007230 104401 012576 38: TYPE      ,MSGVEC          ;"VECTOR?"
1708 007234 104411 RDOCT
1709 007236 012600 MOV      (SP)+,R0
1710 007240 001411 BEQ      58
1711 007242 020027 001000 CMP      R0,01000        ;MAKE SURE ADDRESS IS IN VECTOR AREA
1712 007246 103370 BHS      38
1713 007250 010037 001216 MOV      R0,00TAVEC        ;SAVE AS VECTOR ADDRESS
1714 007254 062700 000002 ADD      #2,R0
1715 007260 010037 001220 MOV      R0,00TAVEC+2
1716 007264 104401 012606 58: TYPE      ,MSGPRI          ;ASK FOR PRIORITY
1717 007270 104411 RDOCT
1718 007272 012600 MOV      (SP)+,R0
1719 007274 001413 BEQ      68              ;IF "0" USE OLD VALUE
1720 007276 020027 000007 CMP      R0,07           ;MAKE SURE ITS VALID
1721 007302 101370 BHI      58
1722 007304 000300 SWAB     R0              ;PUT INTO HIGH BYTE
1723 007306 006200 ASR      R0              ;AND SHIFT
1724 007310 006200 ASR      R0              ;INTO PROPER
1725 007312 006200 ASR      R0              ;POSITION
1726 007314 042700 177437 BIC      #C<340>,R0      ;SAVE ONLY PRIORITY BITS
1727 007320 010037 001222 MOV      R0,00TAPRIO      ;STORE IT AWAY
1728 007324 104401 012620 68: TYPE      ,MTACS          ;TACS="
1729 007330 016746 171652 MOV      TACSL,-(SP)      ;;SAVE TACSL FOR TYPEOUT
1730 007334 104402 TYPOC     ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1731 007336 104401 012626 TYPE      ,MTADB          ;"TADB="
1732 007342 016746 171644 MOV      TADBL,-(SP)    ;;SAVE TADBL FOR TYPEOUT
1733 007346 104402 TYPOC     ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1734 007350 104401 012635 TYPE      ,MTAVEC        ;"VECTOR="
1735 007354 016746 171636 MOV      TAVEC,-(SP)    ;;SAVE TAVEC FOR TYPEOUT
1736 007360 104402 TYPOC     ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1737 007362 104401 012646 TYPE      ,MTAPRI        ;"PRIORITY="
1738 007366 016746 171630 MOV      TAPRIO,-(SP)   ;;SAVE TAPRIO FOR TYPEOUT
1739 007372 104402 TYPOC     ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1740 007374 104401 012661 TYPE      ,MSGOK          ;"OK?"
1741 007400 104407 RDCHR          ;GO READ ONE CHARACTER
1742 007402 012600 MOV      (SP)+,R0        ;GET IT
1743 007404 022700 000015 CMP      #15,R0          ;IS IT "CR"?

```

1744	007410	001406			BEQ	70		;BRANCH IF YES
1745	007412	022700	000131		CMP	0'Y,R0		;IS IT "Y"?
1746	007416	001403			BEQ	70		;IT WAS
1747	007420	104401	001176		TYPE	,SQUES		;TYPE "Y"
1748	007424	000651			BR	10		;AND LET HIM CORRECT THEM
1749	007426	104401	012667	70:	TYPE	,MYES		;TYPE OUT "YES"
1750	007432	012600			MOV	(SP)+,R0		;RESTORE R0
1751	007434	000207			RTS	PC		;AND RETURN

1752
1753
1754
1755
1756
1757
1758
1759
1760 007436
1761 007436 010146
1762 007440 012067 000006
1763 007444 104401 001177
1764 007450 104401
1765 007452 000000
1766 007454 104407
1767 007456 012601
1768 007460 020127 000015
1769 007464 001443
1770 007466 020127 000116
1771 007472 001415
1772 007474 020127 000131
1773 007500 001007
1774 007502 104401 007510
1775 007506 000403
1776
1777 007516
1778 007516 000426
1779 007520 104401 001176
1780 007524 000751
1781 007526
1782 007526 104401 007534
1783 007532 000415
1784
1785 007566
1786 007566 005720
1787 007570 005037 001166
1788 007574 104401 001177
1789 007600 012601
1790 007602 000200
1791

```

;*****
;*****
;ROUTINE TO TYPE DIRECTIONS TO OPERATOR AND WAIT FOR RESPONSE
;CALL:
;   JSR   R0,00ASKQUES
;   RETURN HERE ON A RESPONSE OF "Y"
;   RETURN HERE ON A RESPONSE OF "N"

ASKQUES:
      MOV   R1,-(SP)           ;SAVE R1
      MOV   (R0)+,28          ;PICKUP THE MESSAGE POINTER
      TYPE  ,8CRLF            ;TYPE A "CR" AND "LF"
18:   TYPE  ;TYPE THE MESSAGE
28:   0                        ;MESSAGE POINTER GOES HERE
      RDCHR ;GET ONE CHARACTER FROM THE TTY
      MOV   (SP)+,R1          ;AND PUT IT IN R1
      CMP   R1,015           ;CHECK THIS CHARACTER
      BEQ   48                ;BR IF ITS A "CR"
      CMP   R1,0'N           ;IS IT A "N"?
      BEQ   38                ;BR IF YES
      CMP   R1,0'Y           ;IS IT A "Y"
      BNE   58                ;BR IF NOT "Y"
      TYPE  ,658              ;;TYPE ASCIZ STRING
      BR    648               ;;GET OVER THE ASCIZ
;;658: .ASCIZ / YES/
648:
      BR    48                ;GO TO EXIT
58:   TYPE  ,8QUES           ;UNKNOWN REQUEST
      BR    18                ;GO ASK AGAIN
38:
      TYPE  ,678              ;;TYPE ASCIZ STRING
      BR    668               ;;GET OVER THE ASCIZ
;;678: .ASCIZ * NO---SKIPPING THIS TEST*
668:
      TST   (R0)+             ;STEP OVER "CR" RETURN
      CLR   008TIMES         ;CLEAR ITERATIONS
48:   TYPE  ,8CRLF
      MOV   (SP)+,R1         ;RESTORE R1
      RTS   R0                ;RETURN

```

1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806 007604 012706 001100
1807 007610 013704 001206
1808 007614 013705 001212
1809 007620 000005
1810 007622 012737 007604 001110
1811 007630 004737 010610
1812 007634 010314
1813 007636 112714 000017
1814 007642 032714 000040
1815 007646 001775
1816 007650 112714 000001
1817 007654 104412
1818 007656 032714 020000
1819 007662 001772
1820 007664 000000
1821 007666 000746
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837 007670 004737 010352
1838 007674 012706 001100
1839 007700 013704 001206
1840 007704 013705 001212
1841 007710 000005
1842 007712 012737 007674 001110
1843 007720 004737 010610
1844 007724 010314
1845 007726 112714 000017
1846 007732 032714 000040
1847 007736 001775

```

;////////////////////////////////////
;////////////////////////////////////
;THE FOLLOWING ROUTINES CAN BE USED TO MAKE ADJUSTMENTS TO THE TU60
;NOTE: 000 BEFORE USING ANY OF THE ROUTINES LOAD AND START AT 214 000
;////////////////////////////////////

```

```

;*****
; WRITE FILE GAPS FROM "BOT" TO "EOT"
; START AT 220
; THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND
; THE "WRITE DELAY MONO".
;*****
WFGSUB: MOV    @STACK,SP          ;KEEP THE STACK OUT OF THE WAY
        MOV    @TACSL,TACS      ;SETUP THE TA11 STATUS AND
        MOV    @TADBL,TADB      ;DATA BUFFER REGISTERS
        RESET                    ;RESET THE WORLD
        MOV    @WFGSUB,@@LPERR   ;SETUP THE LOOP ON ERROR ADDRESS
        JSR    PC,@NXTDRV        ;GO SETUP FOR NEXT DRIVE
        MOV    DRIVE,@TACS       ;SELECT DRIVE
        MOV    @REWIND!GO,@TACS  ;SEND TAPE TO "BOT"
1000:   BIT    @READY,@TACS      ;WAIT ON READY
        BEQ    1000
        MOV    @WFG!GO,@TACS     ;WRITE A FILE GAP
10:    MOV    @WFG!GO,@TACS     ;WAIT ON READY
        WAITREADY                ;AT "CLEAR LEADER"?
        BIT    @LEADER,@TACS    ;BR IF NO
        BEQ    10               ;STOP IF YES
        HALT                    ;LOOP ON CONT.
        BR     WFGSUB

```

```

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

```

```

;*****
WRTSUB: JSR    PC,@SETBUF        ;GET BLOCK SIZE AND PATTERN
WLOOP:  MOV    @STACK,SP        ;KEEP THE STACK OUT OF THE WAY
        MOV    @TACSL,TACS      ;SETUP THE TA11 STATUS AND
        MOV    @TADBL,TADB      ;DATA BUFFER REGISTERS
        RESET                    ;RESET THE WORLD
        MOV    @WLOOP,@@LPERR   ;SETUP THE LOOP ON ERROR ADDRESS
        JSR    PC,@NXTDRV        ;GO SETUP FOR NEXT DRIVE
        MOV    DRIVE,@TACS       ;SELECT DRIVE
        MOV    @REWIND!GO,@TACS  ;SEND TAPE TO "BOT"
1000:   BIT    @READY,@TACS      ;WAIT ON READY
        BEQ    1000

```

```

1040 007740 004737 010466      10: JSR    PC,00WRTBLK      ;WRITE A BLOCK
1049 007744 032714 020000      BIT    0LEADER,0TACS    ;AT "CLEAR LEADER"?
1050 007750 001773              BEQ    10                ;BR IF NO
1051 007752 000000              HALT                   ;STOP IF "EOT"
1052 007754 000747              BR     WLOOP           ;LOOP IF CONT.
1053
1054
1055 ;;.....
1056 ; READ CONTINUOUS BLOCKS OF DATA
1057 ; START AT 230
1058 ; THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
1059 ; AND THE "THRESHOLD POT".
1060 ;;.....
1061 007756 012706 001100      RDSUB: MOV    0STACK,SP      ;KEEP THE STACK OUT OF THE WAY
1062 007762 013704 001206      NOV    0TACSL,TACS        ;SETUP THE TA11 STATUS AND
1063 007766 013705 001212      NOV    0TADBL,TADB       ;DATA BUFFER REGISTERS
1064 007772 000005              RESET                  ;RESET THE WORLD
1065 007774 012737 007756 001110  MOV    0RDSUB,000LPERR    ;SETUP THE LOOP ON ERROR ADDRESS
1066 010002 004737 010610      JSR    PC,00NXTDRV       ;GO SETUP FOR NEXT DRIVE
1067 010006 010314              NOV    DRIVE,0TACS       ;SELECT DRIVE
1068 010010 112714 000017      MOVB   0REWIND!GO,0TACS   ;SEND TAPE TO "BOT"
1069 010014 032714 000040      1000: BIT    0READY,0TACS  ;WAIT ON READY
1070 010020 001775              BEQ    1000
1071 010022 004737 010530      10: JSR    PC,00RDBLK      ;READ A BLOCK
1072 010026 032714 020000      BIT    0LEADER,0TACS    ;AT "CLEAR LEADER"?
1073 010032 001351              BNE    RDSUB            ;BR IF YES---LOOP
1074 010034 000772              BR     10
1075
1076
1077 ;;.....
1078 ; WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
1079 ; START AT 234
1080 ; THE PROGRAM WILL HALT THREE(3) TIMES
1081 ; AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
1082 ; HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
1083 ; HALT 2 --- SWR<7:0> = PATTERN DESIRED
1084 ; HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
1085 ; THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
1086 ; IF USING SOFTWARE SWITCH REGISTER, AFTER
1087 ; EACH HALT OPERATOR WILL BE PROMPTED
1088 ; FOR THE VALUE WITH "SWR=XXXXXX NEW="
1089 ; AND THE "GAP TIME MONO".
1090 ;;.....
1091 010036 004737 010352      WGPBLK: JSR   PC,00SETBUF   ;GET BLOCK SIZE AND PATTERN
1092 010042 012706 001100      WGBLOP: NOV   0STACK,SP    ;KEEP THE STACK OUT OF THE WAY
1093 010046 013704 001206      NOV    0TACSL,TACS        ;SETUP THE TA11 STATUS AND
1094 010052 013705 001212      NOV    0TADBL,TADB       ;DATA BUFFER REGISTERS
1095 010056 000005              RESET                  ;RESET THE WORLD
1096 010060 012737 010042 001110  MOV    0WGBLOP,000LPERR   ;SETUP THE LOOP ON ERROR ADDRESS
1097 010066 004737 010610      JSR    PC,00NXTDRV       ;GO SETUP FOR NEXT DRIVE
1098 010072 010314              NOV    DRIVE,0TACS       ;SELECT DRIVE
1099 010074 112714 000017      MOVB   0REWIND!GO,0TACS   ;SEND TAPE TO "BOT"
1100 010100 032714 000040      1000: BIT    0READY,0TACS  ;WAIT ON READY
1101 010104 001775              BEQ    1000
1102 010106 112714 000001      10:  MOVB   0WFG!GO,0TACS   ;WRITE A FILE GAP
1103 010112 104412              WAITREADY              ;WAIT ON READY
    
```

WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT

```

1904 010114 032714 020000          BIT      @LEADER,@TACS          ;AT "CLEAR LEADER"?
1905 010120 001005                   BNE      20                      ;BR IF YES
1906 010122 004737 010466          JSR      PC,@WRTBLK             ;WRITE A BLOCK
1907 010126 032714 020000          BIT      @LEADER,@TACS          ;AT "CLEAR LEADER"?
1908 010132 001765                   BEQ      10                      ;BR IF NO
1909 010134 000000          20:    HALT                    ;STOP AT "EOT"
1910 010136 000741                   BR       WGBLOP                 ;START OVER ON CONT.
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920 010140 012706 001100          RGPBLK: MOV     @STACK,SP          ;KEEP THE STACK OUT OF THE WAY
1921 010144 013704 001206          MOV     @TACSL,TACS            ;SETUP THE TA11 STATUS AND
1922 010150 013705 001212          MOV     @TADBL,TADB           ;DATA BUFFER REGISTERS
1923 010154 000005                   RESET                          ;RESET THE WORLD
1924 010156 012737 010140 001110    MOV     @RGPBLK,@@LPERR        ;SETUP THE LOOP ON ERROR ADDRESS
1925 010164 004737 010610          JSR      PC,@NXTDRV            ;GO SETUP FOR NEXT DRIVE
1926 010170 010314                   MOV     DRIVE,@TACS            ;SELECT DRIVE
1927 010172 112714 000017          MOV     @REWIND!GO,@TACS       ;SEND TAPE TO "BOT"
1928 010176 032714 000040          1000:  BIT     @READY,@TACS      ;WAIT ON READY
1929 010202 001775                   BEQ     1000
1930 010204 004737 010530          10:    JSR      PC,@RDBLK         ;READ A BLOCK OF DATA
1931 010210 032714 020000          BIT     @LEADER,@TACS          ;AT "CLEAR LEADER"?
1932 010214 001351                   BNE     RGPBLK                 ;BR IF YES
1933 010216 112714 000015          MOV     @SFBG!GO,@TACS         ;GET INTO A FILE GAP
1934 010222 104412                   WAITREADY
1935 010224 032714 020000          BIT     @LEADER,@TACS          ;AT "CLEAR LEADER"?
1936 010230 001343                   BNE     RGPBLK                 ;BR IF YES
1937 010232 000764                   BR      10                     ;LOOP
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948 010234 012706 001100          SFFGSB: MOV     @STACK,SP        ;KEEP THE STACK OUT OF THE WAY
1949 010240 013704 001206          MOV     @TACSL,TACS            ;SETUP THE TA11 STATUS AND
1950 010244 013705 001212          MOV     @TADBL,TADB           ;DATA BUFFER REGISTERS
1951 010250 000005                   RESET                          ;RESET THE WORLD
1952 010252 012737 010234 001110    MOV     @SFFGSB,@@LPERR        ;SETUP THE LOOP ON ERROR ADDRESS
1953 010260 004737 010610          JSR      PC,@NXTDRV            ;GO SETUP FOR NEXT DRIVE
1954 010264 010314                   MOV     DRIVE,@TACS            ;SELECT DRIVE
1955 010266 112714 000017          MOV     @REWIND!GO,@TACS       ;SEND TAPE TO "BOT"
1956 010272 032714 000040          1000:  BIT     @READY,@TACS      ;WAIT ON READY
1957 010276 001775                   BEQ     1000
1958 010300 112714 000013          10:    MOV     @SFFG!GO,@TACS     ;SPACE INTO A FILE GAP
1959 010304 104412                   WAITREADY                      ;WAIT ON READY

```

```
1960 010306 032714 020000 BIT 0LEADER,0TACS ;AT "CLEAR LEADER"?
1961 010312 001772 BEQ 10 ;BR IF NO
1962 010314 000000 HALT ;STOP AT "EOT"
1963 010316 000746 BR BSFGSB ;LOOP ON CONT.
1964
1965
1966 ;,;.....
1967 ; BACK SPACE FILE GAP
1968 ;START AT 250
1969 ;THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
1970 ;,;.....
1971 010320 000005 BSFGSB: RESET ;RESET THE WORLD
1972 010322 012737 010320 001110 MOV 0BSFGSB,000LPERR ;LOOP ON ERROR ADDRESS
1973 010330 010314 MOV DRIVE,0TACS ;SELECT DRIVE
1974 010332 112714 000007 10: MOVB 0BSFG!GO,0TACS ;BACK SPACE A FILE GAP
1975 010336 104412 WAITREADY ;WAIT ON READY
1976 010340 032714 020000 BIT 0LEADER,0TACS ;AT "CLEAR LEADER"?
1977 010344 001772 BEQ 10 ;BR IF NO
1978 010346 000000 HALT ;STOP AT BOT
1979 010350 000763 BR BSFGSB ;START OVER ON CONT.
1980
1981
1982 ;,;.....
1983 ; SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
1984 ;,;.....
1985 010352 005000 SETBUF: CLR R0
1986 010354 000000 HALT
1987 010356 022767 000176 170554 CMP 0SWREG,SWR ;OPERATOR PUTS BYTE COUNT IN SWR<7:0>
1988 010364 001001 BNE 200 ;USING S/W SWITCH REG?
1989 010366 104405 GTSWR ;NO- GET OUT
1990 010370 200: ;GET VALUE
1991 010370 157700 170544 BISS 0SWR,R0 ;PICKUP THE BYTE COUNT
1992 010374 001006 BNE 20 ;BR IF NON-ZERO
1993 010376 105777 170537 TSTB 0SWR+1 ;CHECK IF GREATER THAN 377
1994 010402 001402 BEQ 10 ;BR IF NO
1995 010404 012703 000376 MOV 0376,R0 ;SET FOR MAX ALLOWED
1996 010410 005200 10: INC R0 ;MAKE IT 377 OR 1
1997 010412 010037 010462 20: MOV R0,00BLKLIN ;SETUP THE BLOCK LIMIT
1998 010416 005037 010464 CLR 00PATRN
1999 010422 000000 HALT ;OPERATOR PUTS PATTERN IN SWR<7:0>
2000 010424 022767 000176 170506 CMP 0SWREG,SWR ;USING S/W SWITCH REG?
2001 010432 001001 BNE 210 ;NO- GET OUT
2002 010434 104405 GTSWR ;GET VALUE
2003 010436 210: ;CONTINUE
2004 010436 117737 170476 010464 MOVB 0SWR,00PATRN ;PICK UP THE PATTERN
2005 010444 000000 HALT ;SET OPERATIONAL SWITCHES
2006 010446 022767 000176 170464 CMP 0SWREG,SWR ;USING S/W SWITCH REG?
2007 010454 001001 BNE 220 ;NO- GET OUT
2008 010456 104405 GTSWR ;GET VALUE
2009 010460 220: ;CONTINUE
2010 010460 000207 RTS PC ;RETURN
2011 010462 000000 BLKLIN: 0 ;READ AND WRITE BLOCK SIZE
2012 010464 000000 PATRN: 0 ;PATTERN TO GO ON THE TAPE
2013
2014
2015 ;,;.....
```

2016
 2017
 2018 010466 013701 010462
 2019 010472 112714 000003
 2020 010476 104413
 2021 010500 032714 000040
 2022 010504 001010
 2023 010506 005301
 2024 010510 002403
 2025 010512 113715 010464
 2026 010516 000767
 2027 010520 052714 000020
 2028 010524 104412
 2029 010526 000207
 2030
 2031
 2032
 2033
 2034
 2035 010530 013702 010462
 2036 010534 013700 010464
 2037 010540 112714 000005
 2038 010544 104413
 2039 010546 032714 000040
 2040 010552 001012
 2041 010554 005302
 2042 010556 002405
 2043 010560 011501
 2044 010562 120001
 2045 010564 001767
 2046 010566 104005
 2047 010570 000406
 2048 010572 052714 000020
 2049 010576 104412
 2050 010600 005714
 2051 010602 100001
 2052 010604 104001
 2053 010606 000207
 2054
 2055
 2056
 2057
 2058 010610 105777 170324
 2059 010614 100416
 2060 010616 005003
 2061 010620 013701 001230
 2062 010624 122127 000101
 2063 010630 001402
 2064 010632 012703 000400
 2065 010636 105711
 2066 010640 001002
 2067 010642 012701 001224
 2068 010646 010137 001230
 2069 010652 000207

```

; WRITE ROUTINE FOR THE MANUAL OPERATIONS
;*****
WRTBLK:  MOV    @BLKLN,R1          ;PICKUP THE BLOCK SIZE
        MOVB   @WRITE!GO,@TACS   ;START A WRITE
10:     WAITXFER                   ;WAIT ON TRANSFER REQUEST
        BIT    @READY,@TACS      ;DID READY SET?
        BNE    30                 ;BR IF YES
        DEC    R1                 ;COUNT THIS REQUEST
        BLT    20                 ;BR IF TIME FOR ILBS
        MOVB   @PATTRN,@TADB     ;PUT DATA ON TAPE
        BR     10                 ;LOOP
20:     BIS    @ILBS,@TACS       ;WRITE CRC AND SHUT DOWN
        WAITREADY                  ;WAIT ON THE READY FLAG
30:     RTS    PC

;*****
; READ ROUTINE FOR THE MANUAL OPERATIONS
;*****
RDBLK:  MOV    @BLKLN,R2          ;PICKUP THE BLOCK SIZE
        MOV    @PATTRN,R0        ;USE THIS DATA PATTERN TO COMPARE TO
        MOVB   @READ!GO,@TACS   ;START A READ
10:     WAITXFER                   ;WAIT ON TRANSFER REQUEST
        BIT    @READY,@TACS      ;IS READY SET?
        BNE    30                 ;BR IF YES
        DEC    R2                 ;COUNT THIS REQUEST
        BLT    20                 ;BR IF TIME FOR ILBS
        MOV    @TADB,R1          ;READ THE DATA BUFFER
        CMPB   R0,R1             ;CHECK THE DATA
        BEQ    10                 ;BR IF OK
        ERROR  5                 ;BAD DATA
        BR     40                 ;GET OUT
20:     BIS    @ILBS,@TACS       ;READ ILBS
        WAITREADY                  ;WAIT ON READY
30:     TST    @TACS              ;CHECK FOR ERROR
        BPL    40                 ;BR IF NONE
        ERROR  1                 ;ERROR OCCURRED
40:     RTS    PC                ;RETURN

;*****
; ROUTINE TO CHANGE DRIVES
;*****
NXTDRV: TSTB   @SWR              ;IS SW07 ON A (1)?
        BMI    30                 ;BR IF YES
        CLR    DRIVE              ;SET DRIVE TO "A"
        MOV    @DRVPNT,R1         ;GET DRIVE POINTER
        CMPB   (R1)+,@'A         ;IS IT DRIVE "A"?
        BEQ    10                 ;BR IF YES
        MOV    @UNIT,DRIVE        ;SET DRIVE TO "B"
10:     TSTB   (R1)                ;LAST DRIVE BEEN SELECTED
        BNE    20                 ;BR IF NO
        MOV    @DRVKEY,R1         ;RESET DRIVE POINTER
20:     MOV    R1,@DRVPNT         ;SAVE DRIVE POINTER FOR NEXT TIME
30:     RTS    PC                ;GO BACK
    
```

```

2070 ;,.....
2071 ;
2072 ;SBTTL          ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABLITY
2073
2074 ;CALL:
2075 ;      MOV      @DRVKEY,R0
2076 ;      JSR      PC,@EXAM          ;R1 IS DESTROYED
2077 ;      NORMAL RETURN
2078 ;      ERROR RETURN
2079
2080 010654 013701 001206 EXAM:  MOV      @TACSL,R1          ;PICKUP THE "CONTROL & STATUS" REG. ADR.
2081 010660 005011          CLR      (R1)          ;DRIVE="A", FUNCTION="WPG"
2082 010662 122710 000101  CMPB    @'A',(R0)      ;EXAMINE DRIVE "A"?
2083 010666 001402          BEQ     10             ;BR IF YES
2084 010670 052711 000400  BIS     @UNIT,(R1)     ;SELECT DRIVE "B"
2085 010674 032711 000040 10:    BIT     @READY,(R1)  ;WAIT ON READY
2086 010700 001775          BEQ     10
2087 010702 005711          TST     (R1)          ;ANY ERROR?
2088 010704 100024          BPL     40             ;BR IF NO
2089 010706 032711 001000  BIT     @OFFLINE,(R1) ;ERROR DUE TO "OFF LINE"?
2090 010712 001017          BNE     30             ;BR IF YES
2091 010714 032711 010000  BIT     @WRTLOCK,(R1) ;ERROR DUE TO "WRITE LOCK"?
2092 010720 001411          BEQ     20             ;BR IF NO
2093 010722 122777 000201 170210 CMPB    @BIT07|BIT00,@SWR ;"READONLY" SELECTED? (RD1PAS)
2094 010730 001412          BEQ     40             ;BR IF YES
2095 010732 122777 000203 170200 CMPB    @BIT07|BIT01|BIT00,@SWR ;(RD2PAS)?
2096 010740 001406          BEQ     40             ;BR IF YES
2097 010742 000403          BR      30             ;TAKE THE ERROR EXIT
2098 010744 032711 020000 20:    BIT     @LEADER,(R1)  ;ERROR DUE TO "CLEAR LEADER"?
2099 010750 001002          BNE     40             ;BR IF YES
2100 010752 062716 000002 30:    ADD     @2,(SP)       ;TAKE ERROR RETURN
2101 010756 000207 40:    RTS     PC           ;RETURN
  
```

2102
 2103
 2104
 2105
 2106
 2107
 2108
 2109
 2110
 2111
 2112
 2113
 2114
 2115
 2116
 2117
 2118
 2119 010760 105767 170173
 2120 010764 100002
 2121 010766 000000
 2122 010770 000407
 2123 010772 010046
 2124 010774 017600 000002
 2125 011000 112046
 2126 011002 001005
 2127 011004 005726
 2128 011006 012600
 2129 011010 062716 000002
 2130 011014 000002
 2131 011016 122716 000011
 2132 011022 001430
 2133 011024 122716 000200
 2134 011030 001006
 2135 011032 005726
 2136 011034 104401
 2137 011036 001177
 2138 011040 105067 000130
 2139 011044 000755
 2140 011046 004767 000056
 2141 011052 126726 170100
 2142 011056 001350
 2143 011060 016746 170070
 2144
 2145 011064 105366 000001
 2146 011070 002770
 2147 011072 004767 000032
 2148 011076 105367 000072
 2149 011102 000770
 2150
 2151
 2152
 2153 011104 112716 000040
 2154 011110 004767 000014
 2155 011114 132767 000007 000052
 2156 011122 001372
 2157 011124 005726

.SBTTL TYPE ROUTINE

```

;*****
;ROUTINE TO TYPE ASCIZ MESSAGE, MESSAGE MUST TERMINATE WITH A 0 BYTE.
;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
;
;CALL:
;1) USING A TRAP INSTRUCTION
;      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;OR
;      TYPE
;      MESADR
;
8TYPE:  TSTB      8TPFLG      ;;IS THERE A TERMINAL?
        BPL       10         ;;BR IF YES
        HALT      ;;HALT HERE IF NO TERMINAL
        BR        30         ;;LEAVE
10:     MOV       R0, -(SP)   ;;SAVE R0
        MOV       02(SP), R0 ;;GET ADDRESS OF ASCIZ STRING
20:     MOVB      (R0)+, -(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
        BNE       40         ;;BR IF IT ISN'T THE TERMINATOR
        TST       (SP)+     ;;IF TERMINATOR POP IT OFF THE STACK
60:     MOV       (SP)+, R0   ;;RESTORE R0
30:     ADD       02, (SP)    ;;ADJUST RETURN PC
        RTI
40:     CMPB      0HT, (SP)   ;;BRANCH IF <HT>
        BEQ       00
        CMPB      0CRLF, (SP) ;;BRANCH IF NOT <CRLF>
        BNE       50
        TST       (SP)+     ;;POP <CR><LF> EQUIV
        TYPE      ;;TYPE A CR AND LF
        CLR      8CHARCNT   ;;CLEAR CHARACTER COUNT
        BR        20         ;;GET NEXT CHARACTER
50:     JSR       PC, 8TYPEC  ;;GO TYPE THIS CHARACTER
60:     CMPB      8FILLC, (SP)+ ;;IS IT TIME FOR FILLER CHARS.?
        BNE       20         ;;IF NO GO GET NEXT CHAR.
        MOV       8NULL, -(SP) ;;GET # OF FILLER CHARS. NEEDED
        ;;AND THE NULL CHAR.
70:     DECB     1(SP)       ;;DOES A NULL NEED TO BE TYPED?
        BLT       60         ;;BR IF NO--GO POP THE NULL OFF OF STACK
        JSR       PC, 8TYPEC  ;;GO TYPE A NULL
        DECB     8CHARCNT    ;;DO NOT COUNT AS A COUNT
        BR        70         ;;LOOP
;HORIZONTAL TAB PROCESSOR
80:     MOVB     0' , (SP)   ;;REPLACE TAB WITH SPACE
90:     JSR     PC, 8TYPEC   ;;TYPE A SPACE
        BITB     07, 8CHARCNT ;;BRANCH IF NOT AT
        BNE     90         ;;TAB STOP
        TST     (SP)+     ;;POP SPACE OFF STACK
  
```

```

2158 011126 000724          BR      28          ;;GET NEXT CHARACTER
2159 011130 105777 170014  STYPEC: TSTB 08TPS      ;;WAIT UNTIL PRINTER IS READY
2160 011134 100375          BPL      8TYPEC
2161 011136 116677 000002 170006  MOVB 2(SP),08TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2162 011144 122766 000015 000002  CMPB 0CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
2163 011152 001003          BNE      10          ;;BRANCH IF NO
2164 011154 105067 000014  CLRB 8CHARCNT      ;;YES--CLEAR CHARACTER COUNT
2165 011160 000406          BR      8TYPEX      ;;EXIT
2166 011162 122766 000012 000002 10:  CMPB 0LF,2(SP)      ;;IS CHARACTER A LINE FEED?
2167 011170 001402          BEQ      8TYPEX      ;;BRANCH IF YES
2168 011172 105227          INCB  (PC)+        ;;COUNT THE CHARACTER
2169 011174 000000          SCHARCNT: ,WORD 0   ;;CHARACTER COUNT STORAGE
2170 011176 000207          STYPEX: RTS      PC

2171
2172          .SBTTL  TTY INPUT ROUTINE
2173
2174          ;;*****
2175          .ENABL  LSB
2176
2177          ;;*****
2178          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2179          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2180          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2181          ;*WHEN OPERATING IN TTY FLAG MODE.
2182 011200 022767 000176 167732  SCKSWR: CMP      0SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
2183 011206 001074          BNE      150          ;;BRANCH IF NO
2184 011210 105777 167730  TSTB 08TKS          ;;CHAR THERE?
2185 011214 100071          BPL      150          ;;IF NO, DON'T WAIT AROUND
2186 011216 117746 167724  MOVB 08TKB,-(SP)      ;;SAVE THE CHAR
2187 011222 042716 177600  BIC  0'C177,(SP)      ;;STRIP-OFF THE ASCII
2188 011226 022726 000007  CMP  07,(SP)+        ;;IS IT A CONTROL G?
2189 011232 001062          BNE      150          ;;NO, RETURN TO USER
2190 011234 126727 167674 000001  CMPB 8AUTOB,01      ;;ARE WE RUNNING IN AUTO-MODE?
2191 011242 001456          BEQ      150          ;;BRANCH IF YES
2192
2193 011244 104401 011725          SGTSWR: TYPE  ,8CNTLG      ;;ECHO THE CONTROL-G (^G)
2194 011250 104401 011732          TYPE  ,8MSWR      ;;TYPE CURRENT CONTENTS
2195 011254 016746 166716  MOV  SWREG,-(SP)      ;;SAVE SWREG FOR TYPEOUT
2196 011260 104402          TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2197 011262 104401 011743          TYPE  ,8MNEW      ;;PROMPT FOR NEW SWR
2198 011266 005046          198:  CLR  -(SP)      ;;CLEAR COUNTER
2199 011270 005046          CLR  -(SP)      ;;THE NEW SWR
2200 011272 105777 167646          78:  TSTB 08TKS      ;;CHAR THERE?
2201 011276 100375          BPL      78          ;;IF NOT TRY AGAIN
2202
2203 011300 117746 167642  MOVB 08TKB,-(SP)      ;;PICK UP CHAR
2204 011304 042716 177600  BIC  0'C177,(SP)      ;;MAKE IT 7-BIT ASCII
2205
2206
2207
2208 011310 021627 000025          98:  CMP  (SP),025      ;;IS IT A CONTROL-U?
2209 011314 001005          BNE      100          ;;BRANCH IF NOT
2210 011316 104401 011720          TYPE  ,8CNTLU      ;;YES, ECHO CONTROL-U (^U)
2211 011322 062706 000006          208:  ADD  06,SP      ;;IGNORE PREVIOUS INPUT
2212 011326 000757          BR      198          ;;LET'S TRY IT AGAIN
2213

```

```

2214
2215 011330 021627 000015      100:  CMP      (SP),015      ;;IS IT A <CR>?
2216 011334 001022              BNE      160           ;;BRANCH IF NO
2217 011336 005766 000004              TST      4(SP)        ;;YES, IS IT THE FIRST CHART?
2218 011342 001403              BEQ      110           ;;BRANCH IF YES
2219 011344 016677 000002 167566      MOV      2(SP),08WR   ;;SAVE NEW SWR
2220 011352 062706 000006      110:  ADD      06,SP        ;;CLEAR UP STACK
2221 011356 104401 001177      140:  TYPE     ,0CRLF       ;;ECHO <CR> AND <LF>
2222 011362 126727 167547 000001      CMPB     0INTAG,01    ;;RE-ENABLE TTY KBD INTERRUPTS?
2223 011370 001003              BNE      150           ;;BRANCH IF NOT
2224 011372 012777 000100 167544      MOV      0100,00TKS  ;;RE-ENABLE TTY KBD INTERRUPTS
2225 011400 000002      150:  RTI                    ;;RETURN
2226 011402 004767 177522      160:  JSR      PC,0TYPEC    ;;ECHO CHAR
2227 011406 021627 000060      CMP      (SP),060    ;;CHAR < 0?
2228 011412 002420              BLT      100           ;;BRANCH IF YES
2229 011414 021627 000067      CMP      (SP),067    ;;CHAR > ??
2230 011420 003015              BGT      100           ;;BRANCH IF YES
2231 011422 042726 000060      BIC      060,(SP)+    ;;STRIP-OFF ASCII
2232 011426 005766 000002      TST      2(SP)        ;;IS THIS THE FIRST CHAR
2233 011432 001403              BEQ      170           ;;BRANCH IF YES
2234 011434 006316              ASL      (SP)         ;;NO, SHIFT PRESENT
2235 011436 006316              ASL      (SP)         ;; CHAR OVER TO MAKE
2236 011440 006316              ASL      (SP)         ;; ROOM FOR NEW ONE.
2237 011442 005266 000002      170:  INC      2(SP)        ;;KEEP COUNT OF CHAR
2238 011446 056616 177776      BIS      -2(SP),(SP)  ;;SET IN NEW CHAR
2239 011452 000707              BR       70           ;;GET THE NEXT ONE
2240 011454 104401 001176      100:  TYPE     ,0QUES       ;;TYPE ?<CR><LF>
2241 011460 000720              BR       200         ;;SIMULATE CONTROL-U
2242      .DSABL  LSB
2243
2244
2245      ;;*****
2246      ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2247      ;*CALL:
2248      ;*      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
2249      ;*      RETURN HERE   ;;CHARACTER IS ON THE STACK
2250      ;*                    ;;WITH PARITY BIT STRIPPED OFF
2251      ;
2252
2253 011462 011646      0RDCHR:  MOV      (SP),-(SP)  ;;PUSH DOWN THE PC
2254 011464 016666 000004 000002      MOV      4(SP),2(SP)  ;;SAVE THE PS
2255 011472 105777 167446      10:  TSTB     00TKS        ;;WAIT FOR
2256 011476 100375              BPL      10           ;;A CHARACTER
2257 011500 117766 167442 000004      MOVB     00TKB,4(SP)  ;;READ THE TTY
2258 011506 042766 177600 000004      BIC      0°C<177>,4(SP) ;;GET RID OF JUNK IF ANY
2259 011514 026627 000004 000023      CMP      4(SP),023   ;;IS IT A CONTROL-S?
2260 011522 001013              BNE      30           ;;BRANCH IF NO
2261 011524 105777 167414      20:  TSTB     00TKS        ;;WAIT FOR A CHARACTER
2262 011530 100375              BPL      20           ;;LOOP UNTIL ITS THERE
2263 011532 117746 167410      MOVB     00TKB,-(SP)  ;;GET CHARACTER
2264 011536 042716 177600      BIC      0°C177,(SP)  ;;MAKE IT 7-BIT ASCII
2265 011542 022627 000021      CMP      (SP)+,021    ;;IS IT A CONTROL-Q?
2266 011546 001366              BNE      20           ;;IF NOT DISCARD IT
2267 011550 000750              BR       10           ;;YES, RESUME
2268 011552 026627 000004 000140  30:  CMP      4(SP),0140  ;;IS IT UPPER CASE?
2269 011560 002407              BLT      40           ;;BRANCH IF YES
    
```

```

2270 011562 026627 000004 000175      CMP      4(SP),0175      ;;IS IT A SPECIAL CHART
2271 011570 003003                    BGT      40              ;;BRANCH IF YES
2272 011572 042766 000040 000004      BIC      040,4(SP)      ;;MAKE IT UPPER CASE
2273 011600 000002                    40:      RTI                ;;GO BACK TO USER
2274                                     ;;*****
2275                                     ;;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2276                                     ;;CALL:
2277                                     ;;
2278                                     ;;      RDLIN                ;;INPUT A STRING FROM THE TTY
2279                                     ;;      RETURN HERE        ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2280                                     ;;                          ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
2281 011602 010346                    SRDLIN: MOV      R3,-(SP)      ;;SAVE R3
2282 011604 012703 011710              10:      MOV      @TTYIN,R3        ;;GET ADDRESS
2283 011610 022703 011720              20:      CMP      @TTYIN+0,,R3    ;;BUFFER FULL?
2284 011614 101405                    BLOS     40              ;;BR IF YES
2285 011616 104407                    RDCHR   RDCHR           ;;GO READ ONE CHARACTER FROM THE TTY
2286 011620 112613                    MOVB    (SP)+,(R3)       ;;GET CHARACTER
2287 011622 122713 000177              100:     CMPB    @177,(R3)        ;;IS IT A RUBOUT
2288 011626 001003                    BNE     30              ;;SKIP IF NOT
2289 011630 104401 001176              40:      TYPE    ,@QUES         ;;TYPE A '?'
2290 011634 000763                    BR      10              ;;CLEAR THE BUFFER AND LOOP
2291 011636 111367 000044              30:      MOVB    (R3),90         ;;ECHO THE CHARACTER
2292 011642 104401 011706              TYPE    ,90
2293 011646 122723 000015              CMPB    @15,(R3)+       ;;CHECK FOR RETURN
2294 011652 001356                    BNE     20              ;;LOOP IF NOT RETURN
2295 011654 105063 177777              CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
2296 011660 104401 001200              TYPE    ,@LF           ;;TYPE A LINE FEED
2297 011664 012603                    MOV     (SP)+,R3        ;;RESTORE R3
2298 011666 011646                    MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2299 011670 016666 000004 000002      MOV     4(SP),2(SP)    ;;      FIRST ASCII CHARACTER ON IT
2300 011676 012766 011710 000004      MOV     @TTYIN,4(SP)
2301 011704 000002                    RTI
2302 011706 000                    90:      .BYTE  0                ;;RETURN
2303 011707 000                    .BYTE  0                ;;STORAGE FOR ASCII CHAR. TO TYPE
2304 011710 000010                    .BLKB  0                ;;TERMINATOR
2305 011720 052536 005015 000          @TTYIN: .ASCIZ  /"U/<15><12>  ;;RESERVE 8 BYTES FOR TTY INPUT
2306 011725 136 006507 000012        @CNTLU: .ASCIZ  /"G/<15><12>  ;;CONTROL "U"
2307 011732 005015 053523 020122      @MSWR:  .ASCIZ  <15><12>/SWR = /  ;;CONTROL "G"
2308 011740 020075 000
2309 011743 040 047040 053505      @MNEW:  .ASCIZ  / NEW = /
2310 011750 036440 000040
2311                                     .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2312
2313                                     ;;*****
2314                                     ;;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2315                                     ;;CHANGE IT TO BINARY.
2316                                     ;;CALL:
2317                                     ;;
2318                                     ;;      RDOCT                ;;READ AN OCTAL NUMBER
2319                                     ;;      RETURN HERE        ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2320                                     ;;                          ;;HIGH ORDER BITS ARE IN SHIOCT
2321 011754 011646                    SRDOCT: MOV     (SP),-(SP)  ;;PROVIDE SPACE FOR THE
2322 011756 016666 000004 000002      MOV     4(SP),2(SP)    ;;INPUT NUMBER
2323 011764 010046                    MOV     R0,-(SP)       ;;PUSH R0 ON STACK
2324 011766 010146                    MOV     R1,-(SP)       ;;PUSH R1 ON STACK
2325 011770 010246                    MOV     R2,-(SP)       ;;PUSH R2 ON STACK
    
```

```

2326 011772 104410          18:  RDLIN          ;;READ AN ASCII LINE
2327 011774 012600          MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
2328 011776 005001          CLR      R1           ;;CLEAR DATA WORD
2329 012000 005002          CLR      R2
2330 012002 112046          28:  MOVVB     (R0)+,-(SP)  ;;PICKUP THIS CHARACTER
2331 012004 001412          BEQ      38           ;;IF ZERO GET OUT
2332 012006 006301          ASL     R1           ;;*2
2333 012010 006102          ROL     R2
2334 012012 006301          ASL     R1           ;;*4
2335 012014 006102          ROL     R2
2336 012016 005301          ASL     R1           ;;*8
2337 012020 006102          ROL     R2
2338 012022 042716 177770  BIC     0^C7,(SP)    ;;STRIP THE ASCII JUNK
2339 012026 062601          ADD     (SP)+,R1    ;;ADD IN THIS DIGIT
2340 012030 000764          BR      28           ;;LOOP
2341 012032 005726          38:  TST      (SP)+      ;;CLEAN TERMINATOR FROM STACK
2342 012034 010166 000012  MOV     R1,12(SP)    ;;SAVE THE RESULT
2343 012040 010267 000010  MOV     R2,$HIOCT
2344 012044 012602          MOV     (SP)+,R2    ;;POP STACK INTO R2
2345 012046 012601          MOV     (SP)+,R1    ;;POP STACK INTO R1
2346 012050 012600          MOV     (SP)+,R0    ;;POP STACK INTO R0
2347 012052 000002          RTI
2348 012054 000000          $HIOCT: .WORD 0      ;;HIGH ORDER BITS GO HERE
2349          .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
2350
2351          ;;*****
2352          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2353          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
2354          ;*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2355          ;*CALL:
2356          ;*  MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
2357          ;*  TYPOS      ;;CALL FOR TYPEOUT
2358          ;*  .BYTE  N           ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2359          ;*  .BYTE  M           ;;M=1 OR 0
2360          ;*                               ;;1=TYPE LEADING ZEROS
2361          ;*                               ;;0=SUPPRESS LEADING ZEROS
2362          ;*
2363          ;*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2364          ;*STYPOS OR STYPOC
2365          ;*CALL:
2366          ;*  MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
2367          ;*  TYPON      ;;CALL FOR TYPEOUT
2368          ;*
2369          ;*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2370          ;*CALL:
2371          ;*  MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
2372          ;*  TYPOC      ;;CALL FOR TYPEOUT
2373
2374 012056 017646 000000  STYPOS: MOV     0(SP),-(SP)  ;;PICKUP THE MODE
2375 012062 116667 000001 000211  MOVVB   1(SP),$0FILL  ;;LOAD ZERO FILL SWITCH
2376 012070 112667 000207  MOVVB   (SP)+,$0MODE+1 ;;NUMBER OF DIGITS TO TYPE
2377 012074 062716 000002  ADD     $2,(SP)      ;;ADJUST RETURN ADDRESS
2378 012100 000406  BR      $TYPON
2379 012102 112767 000001 000171  STYPOC: MOVVB   $1,$0FILL  ;;SET THE ZERO FILL SWITCH
2380 012110 112767 000006 000165  MOVVB   $6,$0MODE+1  ;;SET FOR SIX(6) DIGITS
2381 012116 112767 000005 000154  STYPON: MOVVB   $5,$0CNT  ;;SET THE ITERATION COUNT
    
```

2302	012124	010346		MOV	R3,-(SP)	;;SAVE R3	
2303	012126	010446		MOV	R4,-(SP)	;;SAVE R4	
2304	012130	010546		MOV	R5,-(SP)	;;SAVE R5	
2305	012132	116704	000145	MOVB	8OMODE+1,R4	;;GET THE NUMBER OF DIGITS TO TYPE	
2306	012136	005404		NEG	R4		
2307	012140	062704	000006	ADD	86,R4	;;SUBTRACT IT FOR MAX. ALLOWED	
2308	012144	110467	000132	MOVB	R4,8OMODE	;;SAVE IT FOR USE	
2309	012150	116704	000125	MOVB	80FILL,R4	;;GET THE ZERO FILL SWITCH	
2390	012154	016605	000012	MOV	12(SP),R5	;;PICKUP THE INPUT NUMBER	
2391	012160	005003		CLR	R3	;;CLEAR THE OUTPUT WORD	
2392	012162	006105		ROL	R5	;;ROTATE MSB INTO "C"	
2393	012164	000404	16:	BR	38	;;GO DO MSB	
2394	012166	006105	26:	ROL	R5	;;FORM THIS DIGIT	
2395	012170	006105		ROL	R5		
2396	012172	006105		ROL	R5		
2397	012174	010503		MOV	R5,R3		
2398	012176	006103	36:	ROL	R3	;;GET LSB OF THIS DIGIT	
2399	012200	105367	000076	DECB	8OMODE	;;TYPE THIS DIGIT?	
2400	012204	100016		BPL	78	;;BR IF NO	
2401	012206	042703	177770	BIC	8177770,R3	;;GET RID OF JUNK	
2402	012212	001002		BNE	48	;;TEST FOR 0	
2403	012214	005704		TST	R4	;;SUPPRESS THIS 0?	
2404	012216	001403		BEQ	58	;;BR IF YES	
2405	012220	005204	46:	INC	R4	;;DON'T SUPPRESS ANYMORE 0'S	
2406	012222	052703	000060	BIS	8'0,R3	;;MAKE THIS DIGIT ASCII	
2407	012226	052703	000040	56:	BIS	8' ,R3	;;MAKE ASCII IF NOT ALREADY
2408	012232	110367	000040	MOVB	R3,88	;;SAVE FOR TYPING	
2409	012236	104401	012276	TYPE	,88	;;GO TYPE THIS DIGIT	
2410	012242	105367	000032	76:	DECB	8OCNT	;;COUNT BY 1
2411	012246	003347		BGT	28	;;BR IF MORE TO DO	
2412	012250	002402		BLT	68	;;BR IF DONE	
2413	012252	005204		INC	R4	;;INSURE LAST DIGIT ISN'T A BLANK	
2414	012254	000744		BR	28	;;GO DO THE LAST DIGIT	
2415	012256	012605	66:	MOV	(SP)+,R5	;;RESTORE R5	
2416	012260	012604		MOV	(SP)+,R4	;;RESTORE R4	
2417	012262	012603		MOV	(SP)+,R3	;;RESTORE R3	
2418	012264	016666	000002 000004	MOV	2(SP),4(SP)	;;SET THE STACK FOR RETURNING	
2419	012272	012616		MOV	(SP)+,(SP)		
2420	012274	000002		RTI		;;RETURN	
2421	012276	000	88:	.BYTE	0	;;STORAGE FOR ASCII DIGIT	
2422	012277	000		.BYTE	0	;;TERMINATOR FOR TYPE ROUTINE	
2423	012300	000	8OCNT:	.BYTE	0	;;OCTAL DIGIT COUNTER	
2424	012301	000	80FILL:	.BYTE	0	;;ZERO FILL SWITCH	
2425	012302	000000	8OMODE:	.WORD	0	;;NUMBER OF DIGITS TO TYPE	

2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469

012304 010046
012306 016600 000002
012312 005740
012314 111000
012316 006300
012320 016000 012340
012324 000200

012326 011646
012330 016666 000004 000002
012336 000002

012340 012326
012342 010760
012344 012102
012346 012056
012350 012116

012352 011250

012354 011200
012356 011462
012360 011602
012362 011754
012364 006630
012366 006734

.SBTTL TRAP DECODER

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

```
STRAP:  MOV    R0,-(SP)      ;;SAVE R0
        MOV    2(SP),R0    ;;GET TRAP ADDRESS
        TST    -(R0)       ;;BACKUP BY 2
        MOVB   (R0),R0     ;;GET RIGHT BYTE OF TRAP
        ASL    R0         ;;POSITION FOR INDEXING
        MOV    STRPAD(R0),R0 ;;INDEX TO TABLE
        RTS    R0         ;;GO TO ROUTINE
```

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

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

.SBTTL TRAP TABLE

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

```
; ROUTINE
; -----
STRPAD: .WORD  STRAP2
        $TYPE  ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
        $TYPOC ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOS ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPON ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)

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

        $CKSWR ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
        $RDCHR ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
        $RDLIN ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
        $RDOCT ;;CALL=RDOCT    TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
        WAIT.ON.READY ;;CALL=WAITREADY TRAP+12(104412) WAIT ON THE READY BIT TO
        WAIT.FOR.XFER ;;CALL=WAITXFER TRAP+13(104413) WAIT ON XFER REQ.
```

```

2470 .SBTTL POWER DOWN AND UP ROUTINES
2471
2472 ;;*****
2473 ;POWER DOWN ROUTINE
2474 012370 012737 012534 000024 $PWRDN: MOV $ILLUP,$PWRVEC ;;SET FOR FAST UP
2475 012376 012737 000340 000026 MOV $340,$PWRVEC+2 ;;PRIO:7
2476 012404 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
2477 012406 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
2478 012410 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
2479 012412 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
2480 012414 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
2481 012416 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
2482 012420 017746 166514 MOV $SWR,-(SP) ;;PUSH $SWR ON STACK
2483 012424 010667 000110 MOV $P,$SAVR6 ;;SAVE SP
2484 012430 012737 012442 000024 MOV $PWRUP,$PWRVEC ;;SET UP VECTOR
2485 012436 000000 HALT
2486 012440 000776 BR -2 ;;HANG UP
2487
2488 ;;*****
2489 ;POWER UP ROUTINE
2490 012442 012737 012534 000024 $PWRUP: MOV $ILLUP,$PWRVEC ;;SET FOR FAST DOWN
2491 012450 016706 000064 MOV $SAVR6,$P ;;GET SP
2492 012454 005067 000060 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
2493 012460 005267 000054 18: INC $SAVR6 ;;WAIT FOR THE INC
2494 012464 001375 BNE 18 ;;OF WORD
2495 012466 012677 166446 MOV (SP)+,$SWR ;;POP STACK INTO $SWR
2496 012472 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
2497 012474 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
2498 012476 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
2499 012500 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
2500 012502 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
2501 012504 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
2502 012506 012737 012370 000024 MOV $PWRDN,$PWRVEC ;;SET UP THE POWER DOWN VECTOR
2503 012514 012737 000340 000026 MOV $340,$PWRVEC+2 ;;PRIO:7
2504 012522 104401 TYPE ;;REPORT THE POWER FAILURE
2505 012524 012542 $PWRNG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
2506 012526 012716 MOV (PC)+,(SP) ;;RESTART AT PWRST
2507 012530 002172 $PWRAD: .WORD PWRST ;;RESTART ADDRESS
2508 012532 000002 RTI
2509 012534 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
2510 012536 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
2511 012540 000000 $SAVR6: 0 ;;PUT THE SP HERE
2512 012542 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
2513 012550 000122
2514 .EVEN
    
```

35

B6

2515	012552	042200	044522	042526	MSGDRV: .ASCIZ	<CRLF>"DRIVE(S)? "
2516	012560	051450	037451	000040		
2517	012566	005015	040524	051503	MSGASK: .ASCIZ	<15><12>/TAC8?/
2518	012574	000077				
2519	012576	042526	052103	051117	MSGVEC: .ASCIZ	/VECTOR?/
2520	012604	000077				
2521	012606	051120	047511	044522	MSGPRI: .ASCIZ	/PRIORITY?/
2522	012614	054524	000077			
2523	012620	040524	051503	000075	MTACS: .ASCIZ	/TACS=/
2524	012626	052040	042101	036502	MTADB: .ASCIZ	/ TADB=/
2525	012634	000				
2526	012635	040	042526	052103	MTAVEC: .ASCIZ	/ VECTOR=/
2527	012642	051117	000075			
2528	012646	050040	044522	051117	MTAPRI: .ASCIZ	/ PRIORITY=/
2529	012654	052111	036531	000		
2530	012661	015	047412	037513	MSGOK: .ASCIZ	<15><12>/OK?/
2531	012666	000				
2532	012667	131	051505	000200	MYES: .ASCIZ	/YES/<CRLF>
2533	012674	047520	042527	020122	MTUPNR: .ASCIZ	/POWER DOWN TU60/
2534	012702	047504	047127	052040		
2535	012710	033125	000060			
2536	012714	047520	042527	020122	MPNRUP: .ASCIZ	/POWER UP TU60/
2537	012722	050125	052040	033125		
2538	012730	000060				
2539	012732	042522	047515	042526	MOFFLN: .ASCIZ	/REMOVE CASSETTE FROM DRIVE UNDER TEST/
2540	012740	041440	051501	042523		
2541	012746	052124	020105	051106		
2542	012754	046517	042040	044522		
2543	012762	042526	052440	042116		
2544	012770	051105	052040	051505		
2545	012776	000124				
2546	013000	042522	046120	041501	MTAPE: .ASCIZ	/REPLACE CASSETTE/
2547	013006	020105	040503	051523		
2548	013014	052105	042524	000		
2549	013021	123	052105	053440	MWRTLK: .ASCIZ	/SET WRITE LOCK/
2550	013026	044522	042524	046040		
2551	013034	041517	000113			
2552	013040	042523	020124	051127	MLAST: .ASCIZ	/SET WRITE ENABLE/
2553	013046	052111	020105	047105		
2554	013054	041101	042514	000		
2555	013061	123	040524	052524	EM1: .ASCIZ	/STATUS PROBLEM/
2556	013066	020123	051120	041117		
2557	013074	042514	000115			
2558	013100	051042	040505	054504	EM2: .ASCIZ	/"READY" FAILED TO SET/
2559	013106	020042	040506	046111		
2560	013114	042105	052040	020117		
2561	013122	042523	000124			
2562	013126	052042	040522	051516	EM3: .ASCIZ	/"TRANSFER REQUEST" FAILED TO SET/
2563	013134	042506	020122	042522		
2564	013142	052521	051505	021124		
2565	013150	043040	044501	042514		
2566	013156	020104	047524	051440		
2567	013164	052105	000			
2568	013167	124	042510	053440	EM4: .ASCIZ	/THE WRONG FLAG SET/
2569	013174	047522	043516	043040		
2570	013202	040514	020107	042523		

2571	013210	000124							
2572	013212	040504	040524	050040	EM5:	.ASCIZ	/DATA PROBLEM/		
2573	013220	047522	046102	046505					
2574	013226	000							
2575	013227	120	020103	020040	DH1:	.ASCIZ	/PC TACS/		
2576	013234	020040	052040	041501					
2577	013242	000123							
2578	013244	041520	020040	020040	DH2:	.ASCIZ	/PC TACS WAIT ADDRESS/		
2579	013252	020040	040524	051503					
2580	013260	020040	020040	040527					
2581	013266	052111	040440	042104					
2582	013274	042522	051523	000					
2583	013301	120	020103	020040	DH5:	.ASCIZ	/PC TACS EXPECT RCV'D/		
2584	013306	020040	052040	041501					
2585	013314	020123	020040	042440					
2586	013322	050130	041505	020124					
2587	013330	051040	053103	042047					
2588	013336	000							
2589		013340				.EVEN			
2590	013340	001116	001162	000000	DT1:	.WORD	SERRPC,SREG0,0		
2591	013346	001116	001162	001202	DT2:	.WORD	SERRPC,SREG0,SAVPC,0		
2592	013354	000000							
2593	013356	001116	001162	001124	DT5:	.WORD	SERRPC,SREG0,SGDDAT,0		
2594	013364	001126	000000						
2595	013370	001116	001206	000000	DT201:	.WORD	SERRPC,TACSL,0		
2596									
2597	013376	001116	000000		DT202:	.WORD	SERRPC,0		
2598									
2599	013402	040524	030461	043040	EM201:	.ASCIZ	"TA11 FAILED TO RESPOND"		
2600	013410	044501	042514	020104					
2601	013416	047524	051040	051505					
2602	013424	047520	042116	000					
2603	013431	116	020117	051104	EM202:	.ASCIZ	"NO DRIVE AVAILABLE"		
2604	013436	053111	020105	053101					
2605	013444	044501	040514	046102					
2606	013452	000105							
2607	013454	041520	020040	020040	DH201:	.ASCIZ	/PC TACS/		
2608	013462	020040	040524	051503					
2609	013470	000							
2610	013471	120	000103		DH202:	.ASCIZ	/PC/		
2611		000001				.END			

ASKADR	007146	556	16920																
ASKDRV	007036	553	16600	1603															
ASKKEY	001232	3950	6430	673															
ASKQUE	007436	731	789	803	818	885	1392	17600											
BEGINX	001730	234	5500	506	588	636	638												
BEGIN1	001326	231	4690																
BEGIN2	001360	232	4760																
BEGIN3	001366	233	475	4700															
BEGIN4	001374	4800																	
BGNCHN	001400	472	474	477	479	4810													
BIT0	= 000001	1660																	
BIT00	= 000001	1560	166	200	2093	2095													
BIT01	= 000002	1550	165	207	2095														
BIT02	= 000004	1540	164	206															
BIT03	= 000010	1530	163	205															
BIT04	= 000020	1520	162	204															
BIT05	= 000040	1510	161	203															
BIT06	= 000100	1500	160	202															
BIT07	= 000200	1490	159	201	2093	2095													
BIT08	= 000400	1480	158	200	1470														
BIT09	= 001000	1470	157	199	1466	1550													
BIT1	= 000002	1650																	
BIT10	= 002000	1460	190	1534															
BIT11	= 004000	1450	197	1493															
BIT12	= 010000	1440	196																
BIT13	= 020000	1430	195	1541															
BIT14	= 040000	1420	194	1464															
BIT15	= 100000	1410	193	654															
BIT2	= 000004	1640																	
BIT3	= 000010	1630																	
BIT4	= 000020	1620																	
BIT5	= 000040	1610																	
BIT6	= 000100	1600																	
BIT7	= 000200	1590																	
BIT8	= 000400	1580																	
BIT9	= 001000	1570																	
BLKLIM	010462	19970	20110	2010	2035														
BPTVEC	= 000014	1730																	
BSBG	= 000010	1860	765	921	961	1021	1061	1113	1153	1213	1253								
BSFG	= 000006	1850	760	916	956	1016	1056	1100	1140	1200	1240	1974							
BSFGSD	010320	241	19710	1972	1979														
CHKADR	001752	552	555	5740															
CHKDRV	002032	6120																	
CKSWR	= 104406	1463	1526	1549	24640														
CR	= 000015	810	2162	2172															
CRCERR	= 040000	1940	997	1009															
CRLF	= 000200	820	530	2133	2172	2515	2532												
CURDRV	001234	3960	6660	667															
DDISP	= 177570	880	364	509															
DH1	013227	417	25750																
DH2	013244	423	429	435	25700														
DH201	013454	449	26070																
DH202	013471	454	26100																
DH5	013301	441	25030																
DISPLA	001142	3640	5090	5170	15070	15330													
DISPRE	000174	2280	517																

E5

E6

DRVKEY	001224	3930	394	470*	612	625	646	671	1661*	1666	1680	1684	2067	
DRVPT	001230	3940	626*	646*	657	672*	2061	2068*						
DSWR	= 177570	870	363	500										
DT1	013340	410	25900											
DT2	013346	424	430	436	25910									
DT201	013370	450	25950											
DT202	013376	455	25970											
DT5	013356	442	25930											
ENTVEC	= 000030	1760	492*	493*										
EM1	013061	416	25550											
EM2	013100	422	25500											
EM201	013402	440	25990											
EM202	013431	453	26030											
EM3	013126	420	25620											
EM4	013167	434	25600											
EM5	013212	440	25720											
ERROR	= 100000	1930												
ERRVEC	= 000004	1690	506	507*	510*	574*	500*	1469	1470*	1472*	1475*			
EXAM	010654	613	617	622	20000									
EXIT	007136	1670	16040											
FGAP	= 004000	1970	1009	1109										
FUNCTI	= 000016	2090												
FUNCO	= 000002	2070	209											
FUNC1	= 000004	2060	209											
FUNC2	= 000010	2050	209											
GNS	= ***** U	227	537	664	719	723	1776	1704	2457	2450	2459	2460	2462	2464
		2465	2466	2467	2460	2469								
GO	= 000001	2000	303	306	325	320	605	702	704	745	750	755	760	765
		770	775	700	837	845	856	865	869	892	900	991	1101	1103
		1107	1202	1204	1200	1293	1305	1300	1326	1320	1335	1337	1339	1343
		1350	1360	1370	1300	1397	1410	1013	1016	1045	1060	1099	1902	1927
		1933	1955	1950	1974	2019	2037							
		532	679	1909	2002	2000	24620							
GTSWR	= 104405													
HERE	001730	523	525	536	5300									
HGHTIM	006706	706	1603*	16170										
HT	= 000011	790	2131	2172										
ILBS	= 000020	2040	052	063	076	995	1105	1206	1312	1330	1341	1346	2027	2040
INT.EN	= 000100	2020												
IOTVEC	= 000020	1740	490*	491*										
ITEMS2	001306	4460												
LEADER	= 020000	1950	097	1009	1365	1303	1010	1049	1072	1904	1907	1931	1935	1960
		1976	2090											
LF	= 000012	800	2166	2172										
LOOP	007062	16670	1601											
LOOP1	000500	2910	292											
LOOP2	000600	3010	312											
LOOP3	000700	3230	334											
MANUAL	002152	632	6400											
MAXCNT	006732	654*	706*	707*	700*	709*	1603	16240	1631					
NLAST	013040	1393	25520											
NOFFLN	012732	804	25390											
NPWRUP	012714	790	25360											
MSGASK	012566	1693	25170											
MSGDRV	012552	1660	25150											
MSGID	001702	526	5300	645										
MSGOK	012661	1740	25300											

NSGPRI	012606	1716	25210											
NSGVEC	012576	1707	25190											
NTACS	012620	1720	25230											
NTADB	012626	1731	25240											
NTAPE	013000	019	25460											
NTAPRI	012646	1737	25200											
NTAVEC	012635	1734	25260											
NTUPWR	012674	732	25330											
NWRTLK	013021	006	25490											
NYES	012667	1749	25320											
NEXT	007116	1670	1674	16700										
NOTA	007074	1660	16710											
NOTB	007106	1672	16750											
NOTLGL	007130	1665	1676	16020	1605									
NXTDRV	010610	1011	1043	1066	1097	1925	1953	20500						
OFFLIN	001000	1990	737	795	009	010	024	1009	1400	2009				
OK	002164	641	6430											
PATTRN	010464	1990*	2004*	20120	2025	2036								
PIRQ	177772	060												
PIRQVE	000240	1000												
PR0	000000	1030												
PR1	000040	1040												
PR2	000100	1050												
PR3	000140	1060												
PR4	000200	1070												
PR5	000240	1080												
PR6	000300	1090												
PR7	000340	1100												
PS	177776	030	04	649*										
PSW	177776	040												
PWRST	002172	6450	2507											
PWRVEC	000024	1750	496*	497*	2474*	2475*	2404*	2490*	2502*	2503*				
RDBLK	010530	1071	1930	20350										
RDCHR	104407	1741	1766	2205	24650									
RDLIN	104410	1662	2326	24660										
RDOCT	104411	1694	1700	1717	24670									
RDSUB	007756	237	10610	1065	1073									
READ	000004	1040	320	755	911	951	991	1011	1051	1103	1143	1103	1203	1243
		1204	1300	1320	1339	1343	2037							
READY	000040	2030	304	311	326	333	606	1607	1637	1014	1046	1069	1900	1920
		1956	2021	2039	2005									
RESVEC	000010	1700												
REWIND	000016	1090	303	325	605	702	700	037	092	936	976	909	1036	1076
		1120	1160	1101	1220	1260	1202	1305	1326	1337	1350	1370	1397	1410
		1013	1045	1060	1099	1927	1955							
RGPBLK	010140	239	19200	1924	1932	1936								
SAVPC	001202	3040	1604*	1605*	1621	1632*	1633*	1649	2591					
SAVPS	001204	3050	1606*	1620	1634*	1640								
SETBUF	010352	1037	1091	19850										
SFBG	000014	1000	775	931	971	1031	1071	1123	1163	1223	1263	1200	1933	
SFFG	000012	1070	770	926	966	1026	1066	1110	1150	1107	1210	1250	1950	
SFFGSB	010234	240	19400	1952	1963									
STACK	001100	740	400	1006	1030	1061	1092	1920	1940					
START	002204	644	6470	1444										
STKLMT	177774	050												
SWR	001140	291	309	3630	406	500*	510	516*	530	677	1464	1470	1400	1406

		1493	1534	1541	1546	1550	1907	1991	1993	2000	2004	2006	2050	2093
SWREG	000176	2095	2102	2219*	2402	2495*	2000	2006	2102	2195				
SW0	= 000001	2290	516	530	677	1907								
SW00	= 000001	1300												
SW01	= 000002	1200	130											
SW02	= 000004	1270	137											
SW03	= 000010	1260	136											
SW04	= 000020	1250	135											
SW05	= 000040	1240	134											
SW06	= 000100	1230	133											
SW07	= 000200	1220	132											
SW08	= 000400	1210	131											
SW09	= 001000	1200	130											
SW1	= 000002	1190	129											
SW10	= 002000	1370												
SW11	= 004000	1100	1707											
SW12	= 010000	1170												
SW13	= 020000	1160												
SW14	= 040000	1150												
SW15	= 100000	1140												
SW2	= 000004	1130												
SW3	= 000010	1360												
SW4	= 000020	1350												
SW5	= 000040	1340												
SW6	= 000100	1330												
SW7	= 000200	1320												
SW8	= 000400	1310												
SW9	= 001000	1300												
TACSH	001210	1290												
TACSL	001206	3000	1702*	1703*										
		3070	576	650	1699*	1702	1729	1807	1839	1862	1893	1921	1949	2000
		2595												
TADBH	001214	3900	1705*	1706*										
TADBL	001212	3890	651	1701*	1705	1732	1800	1840	1863	1894	1922	1950		
TAPRIO	001222	3920	1727*	1730										
TAVEC	001216	3910	473	647*	640*	1713*	1715*	1735						
TBITVE=	000014	1710												
TDRV	002274	659	6610											
TIMERR=	002000	1900	1009											
TKVEC =	000060	1700												
TPVEC =	000064	1790												
TRAPVE=	000034	1770	494*	495*										
TRTVEC=	000014	1720												
TR.REQ=	000200	2010												
TST1	002422	603	6960											
TST10	003224	806	821	825	8300									
TST11	003422	8010												
TST12	003772	804	9040											
TST13	004332	906	10040											
TST14	004644	1006	11760											
TST15	005206	1170	12760											
TST16	005362	1279	13210											
TST17	005500	1323	13510											
TST2	002502	699	7130											
TST20	005564	1354	1366	13710										
TST21	005642	1374	1304	13090										

TST22	005716	800	1401	14060															
TST3	002646	716	7200																
TST4	002714	730	7430																
TST5	003062	744	782	7870															
TST6	003120	734	796	8010															
TST7	003164	811	8160																
TYPE =	104401	526	645	662	667	660	717	721	1434	1536	1544	1566	1575	1577					
		1500	1502	1506	1594	1660	1602	1693	1707	1716	1720	1731	1734	1737					
		1740	1747	1749	1763	1764	1774	1779	1782	1788	2136	2193	2194	2197					
		2210	2221	2240	2289	2292	2296	2409	24570	2504									
TYPERR	006512	1543	15660																
TYPOC =	104402	1591	1730	1733	1736	1739	2196	24500											
TYPON =	104404	24600																	
TYPOS =	104403	24590																	
UNIT =	000400	2000	660	2064	2084														
WAITRE =	104412	703	705	746	751	756	761	766	771	776	794	830	853	864					
		866	877	893	990	996	1102	1106	1100	1203	1207	1209	1292	1301					
		1306	1313	1315	1327	1331	1334	1336	1330	1342	1347	1357	1359	1361					
		1377	1379	1381	1398	1411	1817	1903	1934	1959	1975	2020	2049	24600					
WAITXF =	104413	846	851	857	862	870	875	992	994	1104	1205	1311	1329	1340					
		1344	2020	2030	24690														
WAIT.F	006734	16290	2469																
WAIT.O	006630	16010	2460																
WAIT1	006660	16070	1615	1610															
WAIT2	006700	16020	16140																
WAIT3	006714	1600	1612	16200															
WAIT4	006676	1610	16130																
WFG =	000000	1020	704	745	865	901	941	1001	1041	1093	1133	1193	1233	1335					
		1360	1399	1816	1902														
WFGSUB	007604	235	10060	1010	1021														
WGBL0P	010042	10920	1096	1910															
WGPBLK	010036	230	10910																
WLOOP	007674	10300	1042	1052															
WRITE =	000002	1030	306	750	845	856	869	906	946	1006	1046	1090	1130	1190					
		1230	1293	1300	2019														
WRTBLK	010466	1040	1906	20100															
WRTLOC =	010000	1960	902	907	912	917	922	927	932	937	1042	1047	1052	1057					
		1062	1067	1072	1077	1094	1099	1104	1109	1114	1119	1124	1129	1194					
		1199	1204	1209	1214	1219	1224	1229	1400	2091									
WRISUB	007670	236	10370																
%AUT0B	001134	3600	5340	2190	2311														
%BDADR	001122	3550																	
%BDDAT	001126	3570	15300	2593															
%BELL	001172	3790	1536	1562															
%CHARC	011174	21300	21400	2155	21640	21690													
%CKSWR	011200	21020	2464																
%CHTAG	001100	3430	483	484	492	498	499	500											
%CH1 =	000002	3750	3760	3770															
%CH2 =	000004	3750	3760	3770															
%CH3 =	000002	3730	375																
%CNTLG	011725	2193	23060																
%CNTLU	011720	2210	23050																
%CRLF	001177	3010	660	1544	1562	1566	1577	1582	1586	1763	1780	2137	2172	2221					
		2305																	
%DOAGN	006024	1430	1436	14420															
%ENDAD	006014	524	589	639	14300	1557													

SENDCT	005774	498	1432*															
SENDNG	006033	1434	1446*															
SENULL	006030	1445*																
SEOP	005740	1422*																
SEOPCT	005766	498*	1429*	1433														
SERFLG	001103	346*	1453	1482	1484	1490*	1511	1531*	1562									
SERMAX	001115	352*	501*	1484	1506*	1511												
SERROR	006320	492	1525*															
SERRPC	001116	353*	1530*	1539*	1540	1562	2590	2591	2593	2595	2597							
SERRTB	001236	411*	1573															
SERTTL	001112	350*	1537*	1562														
SESCAP	001170	370*	500*	744*	832*	884*	986*	1086*	1178*	1279*	1323*	1354*	1374*	1505*				
		1553	1555	1562														
SFILLC	001156	371*	2141	2172														
SFILLS	001155	370*	2172															
SGDADR	001120	354*																
SGDDAT	001124	356*	1529*	2593														
SGET42	006004	1435*																
SGTSWR	011250	2194*	2462															
SHD	000000	11																
SHIOCT	012054	2343*	2348*															
SICNT	001104	347*	1497*	1498	1500*	1510												
SILLUP	012534	2474	2490	2509*														
SINTAG	001135	361*	2222	2311														
SITEMB	001114	351*	1540*	1562	1568													
SLF	001200	382*	1562	2172	2296	2305												
SLPADR	001106	348*	502*	730*	788*	802*	817*	803*	985*	1085*	1177*	1278*	1353*	1373*				
		1391*	1488*	1503*	1508	1510	1552	1810*	1842*	1865*	1896*	1924*	1952*	1972*				
SLPERR	001110	349*	503*	1488	1504*	1510												
SMAIL	***** U	52*	53*	1503	1546	2125												
SMNEW	011743	2197	2309*															
SMSWR	011732	2194	2307*															
SMXCNT	006316	1501	1510*															
SNULL	001154	369*	2143	2172														
SNWTST	000001	693*	710*	725*	740*	784*	798*	813*	827*	878*	981*	1001*	1173*	1273*				
		1318*	1348*	1368*	1386*	1403*												
SOCNT	012300	2381*	2410*	2423*														
SOMODE	012302	2376*	2380*	2385	2388*	2399*	2425*											
SOVER	006302	1465	1481	1489	1499	1507*												
SPASS	001100	344*	652	698	715	1426*	1427*	1445	1495	1511								
SPOWER	012542	2505	2512*															
SPWRAD	012530	2507*																
SPWRDN	012370	496	2474*	2502														
SPWRNG	012524	2505*																
SPWRUP	012442	2484	2490*															
SQUES	001176	380*	1562	1682	1747	1779	2172	2240	2289	2305								
SRDCHR	011462	2253*	2465															
SRDDEC	***** U	2468																
SRDLIN	011602	2281*	2466															
SRDOCT	011754	2321*	2467															
SRDSZ	000010	2274*																
SREGAD	001160	373*																
SREG0	001162	375*	1527*	2590	2591	2593												
SREG1	001164	376*	1528*															
SRTHAD	006026	1444*																
SR2A	***** U	2468																

SSAVRE= ***** U	2468													
SSAVR6 012540	2483*	2491	2492*	2493*	25110									
SSCOPE 006046	490	14620												
SSETUP= 000137	4690	489	490	492	494	496	498	499	500	502	524	527	1424	
	1463	1526	1549	1557	2177	2311								
SSUP = 177777	4690													
SSVLAD 006254	1473	15020												
SSWR = 167400	10	11	62	63	64	65	66	67	68	377	378	379	499	
	500	502	503	697	714	729	744	780	802	817	831	882	985	
	1005	1177	1277	1322	1352	1372	1390	1407	1417	1425	1437	1443	1445	
	1454	1455	1456	1457	1458	1464	1476	1478	1479	1482	1483	1484	1491	
	1492	1493	1504	1507	1510	1517	1518	1519	1520	1521	1534	1541	1546	
	1550	1562	1787	2500										
SSWRNK= 000000	60	69	1450	1459	1480									
STIMES 001166	3770	499*	697*	714*	729*	831*	882*	1277*	1322*	1352*	1372*	1390*	1407*	
	1425*	1491*	1490	1501*	1510	1787*								
STKB 001146	3660	2175	2186	2203	2257	2263								
STKS 001144	3650	2175	2184	2200	2224*	2255	2261							
STN = 000023	10	11	693	697*	699	710	7140	716	725	7290	734	738	740	
	7440	782	784	7880	796	798	8020	806	811	813	8170	821	825	
	827	8310	870	8820	884	888	981	9850	986	1001	10050	1006	1173	
	11770	1178	1273	12770	1279	1310	13220	1323	1340	13520	1354	1366	1360	
	13720	1374	1384	1386	13900	1401	1403	14070						
STPB 001152	3680	2161*	2172											
STPFLG 001157	3720	2119	2172											
STPS 001150	3670	2159	2172											
STRAP 012304	494	24340												
STRAP2 012326	24450	2456												
STRP = 000014	24490	24580	24590	24600	24610	2462	24630	2464	24650	24660	24670	24680	24690	
	24700													
STRPAD 012340	2439	24560												
STSTNM 001102	3450	655*	1424*	1453	1480	1502*	1507	1511	1533	1562				
STTYIN 011710	2282	2283	2300	23040										
STYPBN= ***** U	2461													
STYPDS= ***** U	2461													
STYPE 010760	21190	2449	2457											
STYPEC 011130	2140	2147	2154	21590	2160	2226								
STYPEX 011176	2165	2167	21700											
STYPOC 012102	23790	2450												
STYPON 012116	2370	2381*	2460											
STYPOS 012056	23740	2459												
XTSTR 006060	14670													
SSGET4= 000000	14370													
SOFFILL 012301	2375*	2379*	2389	24240										
SOFCAT= ***** U	1464	1543												
	2230	2270	2890	2990	3210	3420	383	487	502	503	5300	6650	1445	
	1440	1510	1511	1562	15970	17770	17850	2172	2175	23040	2305	2311	2406	
	2510	25090												

ERRORS DETECTED: 0
 DEFAULT GLOBALS GENERATED: 0

*TACC, TACC/SOL/CRF; SYM=DZTACC, NEW

TA11 MANUAL INTERVENTION TEST MAINDEC-11-DETAC-C
DZTACC.NEW CROSS REFERENCE TABLE -- USER SYMBOLS

MACY11 27(732) 11-JUN-76 13:51 PAGE 65

SEQ 0077

RUN-TIME: 44 29 2 SECONDS
RUN-TIME RATIO: 171/77=2.2
CORE USED: 25K (49 PAGES)

5
L6