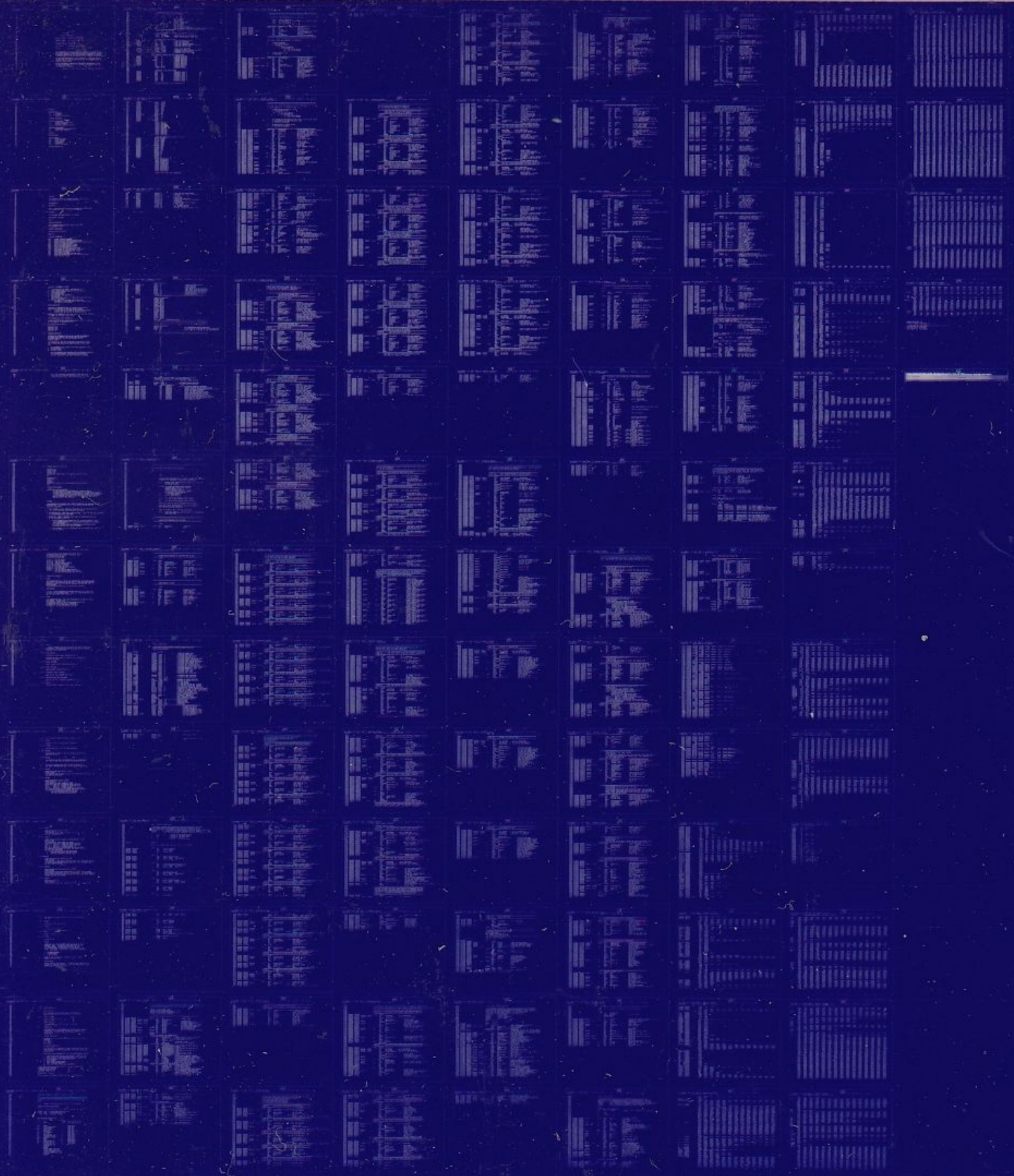


# TA11

LOGIC TEST 1  
MD-11-DZTAA-C

EP-DZTAA-C-DL-A  
COPYRIGHT © 1976  
FICHE 1 OF 1

NOV 1976  
**digital**  
MADE IN USA





.REM ::

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZTAA-C-D  
PRODUCT NAME: TAL1 LOGIC TEST (PART 1)  
DATE REVISED: 21 JUNE 76  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: JIM LACEY

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

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

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

COPYRIGHT (C) 1973,1976 BY DIGITAL EQUIPMEN CORPORATION

TAL1 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
C:\PAC.P11



108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128

- 1. ABSTRACT  
THIS PROGRAM CONTAINS A SERIES OF BASIC LOGIC TESTS THAT CHECK THE TA11 FOR PROPER OPERATION.
- 2. REQUIREMENTS
  - 2.1 EQUIPMENT  
PDP-11 COMPUTER WITH OR WITHOUT HARDWARE SWITCH REGISTER WITH CONSOLE TELETYPE, AND A TA11 CASSETTE
  - 2.2 STORAGE  
THIS PROGRAM REQUIRES APPROX. 4K STORAGE.
  - 2.3 PRELIMINARY PROGRAMS  
NONE
- 3. LOADING PROCEDURE  
USE STANDARD PROCEDURE FOR LOADING .ABS TAPES OR A CASSETTE TAPE.
- 4. STARTING PROCEDURE
  - 4.1 CONTROL SWITCH SETTINGS  
SEE 5.1.
  - 4.2 STARTING ADDRESSES
    - 200 NORMAL STARTING ADDRESS
    - 204 SELECT DRIVE(S) BEFORE STARTING TEST
    - 210 SELECT DRIVE(S) AND ADDRESSES BEFORE STARTING TEST
    - 214 SETUP FOR MANUAL LOOPING
    - 220 WRITE FILE GAP FROM BOT TO EOT
    - 224 WRITE CONTINUOUS BLOCKS OF DATA
    - 230 READ CONTINUOUS BLOCKS OF DATA
    - 234 WRITE FILE GAP AND A BLOCK OF DATA
    - 240 READ BLOCK OF DATA AND INTO A FILE GAP
    - 244 SPACE FWD FILE GAP FROM BOT TO EOT
    - 250 BACK SPACE FILE GAPS
    - 500 LOAD SWITCH REGISTER INTO THE TACS
    - 600 WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
    - 700 READ FROM BOT TO EOT

129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184

4.3 PROGRAM & OPERATOR ACTION

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

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

4.3.1 DRIVE SELECTION

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

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

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

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

4.3.1.1 DRIVE SELECTION EXAMPLES

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

4.3.2 ADDRESS SELECTION

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

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

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

AND THE OPERATOR MUST RESPOND WITH THE DESIRED PARAMETER OR A CARRIAGE RETURN (WHICH IMPLIES LEAVE AS IS). WHEN ALL PARAMETERS HAVE BEEN DEFINED THE PROGRAM

F01

MACY11 27(732) 25-SEP-76 11:04 PAGE 6

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAA.P11

185  
186  
187

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".

188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241

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.

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

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 "SLPADR" AND "SLPERR" AS IT IS  
BEING ENTERED.

THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS

5.2.2 TRAPCATCHER

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

5.2.3 ERROR

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

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



284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315

5.2.4 TRAP

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

5.2.4.1 TYPE (\$TYPE)

ROUTINE TO TYPE AN ASCIZ STRING ON THE TTY

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

5.2.4.2 RDCHR (\$RDCHR)

READ A SINGLE ASCII CHARACTER FROM THE TTY

5.2.4.3 RDLIN (\$RDLIN)

READ AN ASCII STRING FROM THE TTY

5.2.4.4 WAITREADY (WAIT.ON.READY)

WAIT ON THE "TA11 READY" BIT TO SET

5.2.4.5 WAITXFER (WAIT.FOR.XFER.REQ)

WAIT ON THE "TA11 TRANSFER REQUEST" BIT TO SET

5.2.4.6 \$TYPOC

CHANGE A BINARY NUMBER TO OCTAL ASCII AND TYPE IT.

316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361

5.2.5 THE FOLLOWING SUBROUTINES ARE CALLED BY A JSR

5.2.5.1 \$TYPEC

ROUTINE TO TYPE A SINGLE ASCII CHARACTER

5.2.5.2 TYPERR

THIS ROUTINE WILL TYPE THE ERROR MESSAGES

5.2.5.3 SELDRV

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

5.2.5.4 SELADR

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

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

5.2.6.1 WFGSUB

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

5.2.6.2 WRTSUB

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

362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407

5.2.6.3 RDSUB

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

5.2.6.4 WGPBLK

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

5.2.6.5 RGBLK

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

5.2.6.6 SFFGSB

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

5.2.6.7 BSFGSB

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

408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448

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 WR7BLK  
WRITES A BLOCK OF DATA

5.2.7.3 RDBLK  
READS A BLOCK OF DATA

5.2.7.4 NXTDRV  
CHANGE DRIVE

6. ERRORS

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

1. AN ERROR MESSAGE
2. A DATA HEADER
3. A DATA STRING

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

7. RESTRICTIONS

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

```

449
450
451      8.      MISCELLANEOUS
452
453      8.1     EXECUTION TIME
454
455             THE FIRST PASS TAKES APPROXIMATELY 45 SECONDS ALL
456             SUBSEQUENT PASSES TAKE APPROXIMATELY 100 SECONDS.
457
458      8.2     STACK PCINTER
459
460             STACK IS INITIALLY SET TO 1100.
461
462      8.3     PASS COUNT
463
464             A PROGRAM PASS THRU COUNT IS KEPT IN "$PASS".
465
466      8.4     ITERATIONS
467
468             THE FIRST PASS OF THE PROGRAM WILL AUTOMATICALLY PERFORM
469             ITERATIONS. ALL SUBSEQUENT PASSES WILL PERFORM FULL
470             (2000 DECIMAL UNLESS OTHERWISE SPECIFIED WITHIN A TEST),
471             ITERATIONS.
472
473      8.5     SPECIAL REGISTERS
474
475             REGISTER R3,R4 AND R5 ARE RESERVED FOR THE THE FOLLOWING
476             PURPOSES:
477             R3 = DRIVE
478             R4 = TACS
479             R5 = TADB
480
481      9.      PROGRAM DESCRIPTION
482
483             THIS PROGRAM IS A SEQUENCE OF SMALL INDEPENDENT TESTS THAT
484             CHECK THE TA11 FOR PROPER OPERATION WITH TAPE MOTION KEPT
485             AT A MINIMUM.
486
487             THE TESTS CAN BE GROUPED INTO THE FOLLOWING GENERAL GROUPS.
488
489             1.  VERIFY THAT THE "CONTROL AND STATUS" (TACS) AND THE
490             "DATA BUFFER" (TADB) REGISTERS RESPOND TO ADDRESSING.
491             2.  TEST THE "TACS" FOR PROPER OPERATION
492             3.  TEST "READY"
493             4.  TEST "CLEAR LEADER ERROR"
494             5.  TEST "TRANSFER REQUEST"
495             6.  TEST THE "TADB" FOR PROPER OPERATION.
496             %
497             .TITLE TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C
498             ;*COPYRIGHT (C) 1973,1976
499             ;*DIGITAL EQUIPMENT CORP.
500             ;*MAYNARD, MASS. 01754
501             ;*
502             ;*PROGRAM BY JIM LACEY
503             ;*
504             ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC

```



505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551

```
;*PACKAGE (MAINDEC-11-DZGAC-C1),MAR 24, 1976.*
;*****
;*****
;*****
;.REM!
```

GENERAL INFORMATION ABOUT THE TA11/TU60 CASSETTE

ADDRESS MNEMONIC DESCRIPTION  
-----

```
777501 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!

553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608

```

*****
.SBTTL OPERATIONAL SWITCH SETTINGS
*
* SWITCH USE
* -----
* 15 HALT ON ERROR
* 14 LOOP ON TEST
* 13 INHIBIT ERROR TYPEOUTS
* 11 INHIBIT ITERATIONS
* 10 BELL ON ERROR
* 9 LOOP ON ERROR
* 8 LOOP ON TEST IN SWR<7:0>
* 7 LOCK ON CURRENT DRIVE (ONLY VALID WITH MANUAL LOOPING)
*****

```

```

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

```

001100

```

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

```

000011  
000012  
000015  
000200  
177776  
177774  
177772  
177570  
177570

```

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

```

000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007

```

* PRIORITY LEVEL DEFINITIONS
PR0= 0 ;;PRIORITY LEVEL 0
PR1= 40 ;;PRIORITY LEVEL 1
PR2= 100 ;;PRIORITY LEVEL 2
PR3= 140 ;;PRIORITY LEVEL 3
PR4= 200 ;;PRIORITY LEVEL 4
PR5= 240 ;;PRIORITY LEVEL 5
PR6= 300 ;;PRIORITY LEVEL 6
PR7= 340 ;;PRIORITY LEVEL 7

```

000000  
000040  
000100  
000140  
000200  
000240  
000300  
000340

;\*SWITCH REGISTER\* SWITCH DEFINITIONS

609 100000  
 610 040000  
 611 020000  
 612 010000  
 613 004000  
 614 002000  
 615 001000  
 616 000400  
 617 000200  
 618 000100  
 619 000040  
 620 000020  
 621 000010  
 622 000004  
 623 000002  
 624 000001

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

625 100000  
 626 040000  
 627 020000  
 628 010000  
 629 004000  
 630 002000  
 631 001000  
 632 000400  
 633 000200  
 634 000100  
 635 000040  
 636 000020  
 637 000010  
 638 000004  
 639 000002  
 640 000001

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

.\*BASIC "CPU" TRAP VECTOR ADDRESSES

641  
 642  
 643  
 644  
 645  
 646  
 647  
 648  
 649  
 650  
 651  
 652  
 653  
 654  
 655  
 656  
 657  
 658  
 659  
 660  
 661  
 662  
 663  
 664

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAAC.P11 BASIC DEFINITIONS

665	000004	ERRVEC= 4
666	000010	RESVEC= 10
667	000014	TBITVEC=14
668	000014	TRTVEC= 14
669	000014	BPTVEC= 14
670	000020	IOTVEC= 20
671	000024	PWRVEC= 24
672	000030	EMTVEC= 30
673	000034	TRAPVEC=34
674	000060	TKVEC= 60
675	000064	TPVEC= 64
676	000240	PIRQVEC=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

```

```

677      ;TA11 FUNCTIONS
678      000000      WFG= 0      ;WRITE FILE GAP FUNCTION
679      000002      WRITE= 2      ;WRITE FUNCTION
680      000004      READ= 4      ;READ FUNCTION
681      000006      BSFG= 6      ;BACK SPACE FILE GAP FUNCTION
682      000010      BSBG= 10      ;BACK SPACE BLOCK GAP FUNCTION
683      000012      SFFG= 12      ;SPACE FWD FILE GAP FUNCTION
684      000014      SFBG= 14      ;SPACE FWD BLOCK GAP FUNCTION
685      000016      REWIND= 16      ;REWIND FUNCTION
686      ;;*****
687
688      ;TA11 BIT ASSIGNMENT
689      100000      ERROR= BIT15
690      040000      CRCERR= BIT14
691      020000      LEADER= BIT13
692      010000      WRTLOCK=BIT12
693      004000      FGAP= BIT11
694      002000      TIMERR= BIT10
695      001000      OFFLINE=BIT09
696      000400      UNIT= BIT08
697      000200      TR.REQ= BIT07
698      000100      INT.EN= BIT06
699      000040      READY= BIT05
700      000020      ILBS= BIT04
701      000010      FUNC2= BIT03
702      000004      FUNC1= BIT02
703      000002      FUNC0= BIT01
704      000001      GO= BIT00
705      FUNCTION=      FUNC2+FUNC1+FUNC0
706      ;////////////////////////////////////
707      ;////////////////////////////////////
708      ;////////////////////////////////////
709
710      ;SPECIAL REGISTERS
711      000003      DRIVE= %3      ;R3 CONTAINS THE DRIVE UNDER TEST
712      000004      TACS= %4      ;R4 IS USED AS A POINTER TO THE TACS REGISTER
713      000005      TADB= %5      ;R5 IS USED AS A POINTER TO THE TADB REGISTER.
714
715      ;////////////////////////////////////
716      ;////////////////////////////////////

```



```

717          .SBTTL TRAP CATCHER
718
719          000000
720          .=0
721          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
722          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
723          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
724          000174 000000
725          000176 000000
726          .=174
727          000200 000137 001356
728          000204 000137 001410
729          000210 000137 001416
730          000214 000137 001424
731          000220 000137 012622
732          000224 000137 012706
733          000230 000137 012774
734          000234 000137 013054
735          000240 000137 013156
736          000244 000137 013252
737          000250 000137 013336

          DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
          SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
          .SBTTL STARTING ADDRESS(ES)
          JMP      @#BEGIN1        ;;JUMP TO STARTING ADDRESS OF PROGRAM
          JMP      @#BEGIN2        ;SELECT DRIVE(S) BEFORE STARTING TEST
          JMP      @#BEGIN3        ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
          JMP      @#BEGIN4        ;SETUP FOR MANUAL LOOPING
          JMP      @#WFGSUB        ;WRITE FILE GAP FROM BOT TO EOT
          JMP      @#WRTSUB        ;WRITE CONTINUOUS BLOCKS OF DATA
          JMP      @#RDSUB        ;READ CONTINUOUS BLOCKS OF DATA
          JMP      @#WGPBLK        ;WRITE FILE GAP AND A BLOCK OF DATA
          JMP      @#RGPBLK        ;READ BLOCK OF DATA AND INTO A FILE GAP
          JMP      @#SFFGSB        ;SPACE FWD FILE GAP FROM BOT TO EOT
          JMP      @#RSFGSB        ;BACK SPACE FILE GAPS
    
```

738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790

```

;/////////////////////////////////////////////////////////////////
;/////////////////////////////////////////////////////////////////

```

;THE FOLLOWING ROUTINES CAN BE TOGGLED IN.

```

;/////////////////////////////////////////////////////////////////

```

.REM !

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

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

\*\* NOTE: IF USING SOFTWARE SWITCH REGISTER LOCATION SWR (=1140) MUST CONTAIN ADDRESS "SWREG" (=176).  
\*\*\*\*\* PUT VALUE INTO 176 \*\*\*\*\*  
\*\* REGISTERS 3,4,&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

```

;*****
;*****

```

.=500

000500 017714 000434  
000504 000775

```

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

```

WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT

```

791 ;*****
792 ;
793 ;WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
794 ;*****
795         000600
796         .=600
797 000600 000005 LOOP2: RESET ;CLEAR ALL FLAGS
798 000602 010314 MOV DRIVE,@TACS ;SELECT DRIVE
799 000604 112714 000017 MOVB #REWIND!GO,@TACS ;START A REWIND
800 000610 032714 000040 BIT #READY,@TACS ;WAIT ON REWIND TO FINISH
801 000614 001775 BEQ 1$
802 000616 112714 000003 MOVB #WRITE!GO,@TACS ;START A WRITE
803 000622 105714 2$: TSTB @TACS ;CHECK FOR TRANSFER REQUEST
804 000624 100003 BPL 3$ ;BR IF NOT SET
805 000626 017715 000306 MOV @SWR,@TADB ;SEND DATA TO TA11
806 000632 000773 BR 2$ ;LOOP
807 000634 032714 000040 BIT #READY,@TACS ;DID READY SET?
808 000640 001357 BNE LOOP2 ;START OVER IF YES
809 000642 000767 BR 2$ ;LOOP
810
811 ;*****
812 ;
813 ;READ FROM BOT TO EOT
814 ;*****
815         000700
816         .=700
817
818
819 000700 000005 LOOP3: RESET ;CLEAR ALL FLAGS
820 000702 010314 MOV DRIVE,@TACS ;SELECT DRIVE
821 000704 112714 000017 MOVB #REWIND!GO,@TACS ;START A REWIND
822 000710 032714 000040 BIT #READY,@TACS ;WAIT ON REWIND TO FINISH
823 000714 001775 BEQ 1$
824 000716 112714 000005 MOVB #READ!GO,@TACS ;START A READ
825 000722 105714 2$: TSTB @TACS ;CHECK TRANSFER REQ
826 000724 100002 BPL 3$ ;BR IF NOT SET
827 000726 011500 MOV @TADB,R0 ;PICKUP THE DATA
828 000730 000774 BR 2$ ;LOOP
829 000732 032714 000040 BI #READY,@TACS ;CHECK READY
830 000736 001360 BNE LOOP3 ;START OVER
831 000740 000770 BR 2$ ;LOOP

```

.SBTTL COMMON TAGS

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

832									
833									
834									
835									
836									
837									
838		001100							
839	001100			SCMTAG:					:: START OF COMMON TAGS
840	001100	000000		\$PASS:	.WORD	0			:: CONTAINS PASS COUNT
841	001102	000		\$TSTNM:	.BYTE	00			:: CONTAINS THE TEST NUMBER
842	001103	000		\$ERFLG:	.BYTE	00			:: CONTAINS ERROR FLAG
843	001104	000000		\$ICNT:	.WORD	00			:: CONTAINS SUBTEST ITERATION COUNT
844	001106	000000		\$LPADR:	.WORD	00			:: CONTAINS SCOPE LOOP ADDRESS
845	001110	000000		\$LPERR:	.WORD	00			:: CONTAINS SCOPE RETURN FOR ERRORS
846	001112	000000		\$ERTTL:	.WORD	00			:: CONTAINS TOTAL ERRORS DETECTED
847	001114	000		\$ITEMB:	.BYTE	00			:: CONTAINS ITEM CONTROL BYTE
848	001115	001		\$ERMAX:	.BYTE	1			:: CONTAINS MAX. ERRORS PER TEST
849	001116	000000		\$ERRPC:	.WORD	00			:: CONTAINS PC OF LAST ERROR INSTRUCTION
850	001120	000000		\$GDADR:	.WORD	00			:: CONTAINS ADDRESS OF 'GOOD' DATA
851	001122	000000		\$BDADR:	.WORD	00			:: CONTAINS ADDRESS OF 'BAD' DATA
852	001124	000000		\$GDDAT:	.WORD	00			:: CONTAINS 'GOOD' DATA
853	001126	000000		\$BDDAT:	.WORD	00			:: CONTAINS 'BAD' DATA
854	001130	000700			.WORD	00			:: RESERVED--NOT TO BE USED
855	001132	000000			.WORD	00			
856	001134	000		\$AUTOB:	.BYTE	00			:: AUTOMATIC MODE INDICATOR
857	001135	000		\$INTAG:	.BYTE	00			:: INTERRUPT MODE INDICATOR
858	001136	000000			.WORD	00			
859	001140	177570		\$SWR:	.WORD	DSWR			:: ADDRESS OF SWITCH REGISTER
860	001142	177570		\$DISPLAY:	.WORD	DDISP			:: ADDRESS OF DISPLAY REGISTER
861	001144	177560		\$TKS:	177560				:: TTY KBD STATUS
862	001146	177562		\$TKB:	177562				:: TTY KBD BUFFER
863	001150	177564		\$TPS:	177564				:: TTY PRINTER STATUS REG. ADDRESS
864	001152	177566		\$TPB:	177566				:: TTY PRINTER BUFFER REG. ADDRESS
865	001154	000		\$NULL:	.BYTE	0			:: CONTAINS NULL CHARACTER FOR FILLS
866	001155	002		\$FILLS:	.BYTE	2			:: CONTAINS # OF FILLER CHARACTERS REQUIRED
867	001156	012		\$FILLC:	.BYTE	12			:: INSERT FILL CHARS. AFTER A "LINE FEED"
868	001157	000		\$TPFLG:	.BYTE	0			:: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
869	001160	000000		\$REGAD:	.WORD	0			:: CONTAINS THE ADDRESS FROM WHICH (\$REGO) WAS OBTAINED
870									:: CONTAINS ((\$REGAD)+0)
871	001162	000000		\$REGO:	.WORD	0			:: CONTAINS ((\$REGAD)+2)
872	001164	000000			.WORD	0			:: MAX. NUMBER OF ITERATIONS
873	001166	000000		\$TIMES:	0				:: ESCAPE ON ERROR ADDRESS
874	001170	000000		\$ESCAPE:	0				:: CODE FOR BELL
875	001172	177607	000377	\$BELL:	.ASCIZ	<207><377><377>			:: QUESTION MARK
876	001176	077		\$QUES:	.ASCII	/?/			:: CARRIAGE RETURN
877	001177	015		\$CRLF:	.ASCII	<15>			:: LINE FEED
878	001200	000012		\$LF:	.ASCIZ	<12>			
879									*****
880	001202	000000		\$AVPC:	.WORD	0			:: STORAGE FOR THE PC
881	001204	000000		\$AVPS:	.WORD	0			:: STORAGE FOR THE PS
882	001206	177500		\$TACSL:	177500				:: LOW BYTE ADDRESS OF TACS
883	001210	177501		\$TACSH:	177501				:: HIGH BYTE ADDRESS OF TACS
884	001212	177502		\$TADBL:	177502				:: LOW BYTE ADDRESS OF TADB
885	001214	177503		\$TADBH:	177503				:: HIGH BYTE ADDRESS OF TADB
886	001216	000260	000262	\$TAVEC:	260,262				:: TAIL VECTOR ADDRESS
887	001222	000300		\$TAPRIO:	300				:: TAIL BR LEVEL 6

J02

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAA.P11 COMMON TAGS

MACY11 27(732) 25-SEP-76 11:04 PAGE 23

888 001224 000000 000000  
889 001230 001224  
890 001232 000000  
891 001234 000000

DRVKEY: 0,0  
DRVPNT: DRVKEY  
ASKKEY: 0  
CURDRV: 0

;DRIVE SELECT KEY:

;CURRENT DRIVE BEING TESTED



892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947

.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 $REGO
      0

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

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

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

;ITEM 5
      EM5      ;COUNT PATTERN FAILED
      DH5      ;PC TACS EXPECT RCV'D
      DT5      ;$ERRPC $REGO $GDDAT $BDDAT
      0

;ITEM 6
      EM6      ;DATA PROBLEM
      DH6      ;PC TADB
      DT6      ;$ERRPC $REG1
      0

;ITEM 7
      EM6      ;DATA PROBLEM
    
```

001236

001236 015712  
001240 016124  
001242 016272  
001244 000000

001246 015731  
001250 016141  
001252 016300  
001254 000000

001256 015757  
001260 016141  
001262 016300  
001264 000000

001266 016020  
001270 016141  
001272 016300  
001274 000000

001276 016043  
001300 016176  
001302 016310  
001304 000000

001306 016070  
001310 016234  
001312 016322  
001314 000000

001316 016070

948	001320	016176	DHS	;PC	TACS	EXPECT	RCV'D
949	001322	016310	DTS	;\$ERRPC	\$REGO	\$GDDAT	\$BDDAT
950	001324	000000	0				
951							
952			;ITEM	10			
953	001326	016105	EM10	;ADDRESS	FAILED		
954	001330	016251	DH10	;PC	ADDRESS		
955	001332	016330	DT10	;\$ERRPC	\$BDADR		
956	001334	000000	0				
957							
958	001336		ITEMS2:	;ITEMS	201-202		
959							
960	001336	016350	EM201	;TA11	FAILED TO RESPOND		
961	001340	016422	DH201	;PC	TACS		
962	001342	016336	JT201	;\$ERRPC	TACS		
963	001344	000000	0	;BOTH	NUMBERS ARE TYPED AS OCTAL NUMBERS		
964							
965	001346	016377	EM202	;NO	DRIVES AVAILABLE		
966	001350	016437	DH202	;PC			
967	001352	016344	DT202	;\$ERRPC			
968	001354	000000	0				
969							

M02

```

970 ;////////////////////////////////////
971 ;////////////////////////////////////
972 ;*****
973
974 ;BEGIN1 IS FOR NORMAL START
975 ;BEGIN2 IS FOR DRIVE SELECTION
976 ;BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
977 ;BEGIN4 IS FOR MANUAL OPERATION
978
979 ;*****
980
981 001356 005005 BEGIN1: CLR R5 ;NORMAL START
982 001360 012737 041101 001224 MOV #AB, @DRVKEY
983 001366 122737 000005 000041 CMPB #5, @#41 ;CASSETTE DDP?
984 001374 001015 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
985 001376 022737 000260 001216 CMP #260, @TAVEC ;STANDARD VECTOR?
986 001404 001011 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
987 001406 000403 BR BEGIN3 ;GET DRIVES AND ADDRESSES
988 001410 012705 000001 BEGIN2: MOV #1, R5 ;ASK FOR DRIVES FLAG
989 001414 000405 BR BGNCMN ;BEGIN COMMON CODE
990 001416 012705 000002 BEGIN3: MOV #2, R5 ;ASK FOR DRIVES AND ADDRESSES
991 001422 000402 BR BGNCMN
992 001424 012705 000003 BEGIN4: MOV #3, R5
993 001430 BGNCMN:
994 .SBTTL INITIALIZE THE COMMON TAGS
995 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
996 001430 012706 001100 MOV #CMTAG, R6 ;;FIRST LOCATION TO BE CLEARED
997 001434 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
998 001436 022706 001140 CMP #SWR, R6 ;;DONE?
999 001374 001374 BNE -.6 ;;LOOP BACK IF NO
1000 001436 001374 001100 MOV #STACK, SP ;;SETUP THE STACK POINTER
1001 ;;INITIALIZE A FEW VECTORS
1002 001450 012737 000020 MOV #SCOPE, @IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1003 001456 012737 000022 MOV #340, @IOTVEC+2 ;;LEVEL 7
1004 001464 012737 011504 MOV #ERROR, @EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1005 001472 012737 000340 MOV #340, @EMTVEC+2 ;;LEVEL 7
1006 001500 012737 015322 MOV #TRAP, @TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1007 001506 012737 000340 MOV -340, @TRAPVEC+2 ;;LEVEL 7
1008 001514 012737 015406 MOV #SPWRDN, @PWRVEC ;;POWER FAILURE VECTOR
1009 001522 012737 000340 MOV #340, @PWRVEC+2 ;;LEVEL 7
1010 001530 016767 007424 MOV SENDCT, SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
1011 001536 005067 177424 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
1012 001542 005067 177422 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1013 001546 112767 000001 177341 MOVB #1, $ERMAX ;;ALLOW ONE ERROR PER TEST
1014 001554 012767 001554 177324 MOV #, $LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1015 001562 012767 001562 177320 MOV #, $LPERR ;;SETUP THE ERROR LOOP ADDRESS
1016 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1017 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1018 001570 013746 000004 MOV @ERRVEC, -(SP) ;;SAVE ERROR VECTOR
1019 001574 012737 001630 000004 MOV #64$, @ERRVEC ;;SET UP ERROR VECTOR
1020 001602 012767 177570 177330 MOV #DSWR, SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
1021 001610 012767 177570 177324 MOV #DDISP, DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1022 001616 022777 177777 177314 CMP #-1, @SWR ;;TRY TO REFERENCE HARDWARE SWR
1023 001624 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1024 ;;AND THE HARDWARE SWR IS NOT = -1
1025 001626 000403 BR 65$ ;;BRANCH IF NO TIMEOUT

```

```

1026 001630 012716 001636      64$:  MOV    #65$, (SP)      ;;SET UP FOR TRAP RETURN
1027 001634 000002                RTI
1028 001636 012767 000176 177274 65$:  MOV    #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1029 001644 012767 000174 177270  MOV    #DISPREG,DISPLAY
1030 001652 012637 000004      66$:  MOV    (SP)+, @#ERRVEC  ;;RESTORE ERROR VECTOR
1031
1032
1033      .SBTTL  TYPE PROGRAM NAME
      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1034 001656 005227 177777      INC    #-1                ;;FIRST TIME?
1035 001662 001036                BNE    HERE                ;;BRANCH IF NO
1036 001664 022737 011200 000042  CMP    #SENDAD, @#42      ;;ACT-11?
1037 001672 001432                BEQ    HERE                ;;BRANCH IF YES
1038 001674 104401 001732      TYPE    MSGID              ;;TYPE ASCIZ STRING
1039      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1040 001700 005737 000042      TST    @#42                ;;ARE WE RUNNING UNDER XXDP/ACT?
1041 001704 001006                BNE    67$                 ;;BRANCH IF YES
1042 001706 026727 177226 000176  CMP    SWR, #SWREG        ;;SOFTWARE SWITCH REG SELECTED?
1043 001714 001005                BNE    68$                 ;;BRANCH IF NO
1044 001716 104405                GTSWR                       ;;GET SOFT-SWR SETTINGS
1045 001720 000403                BR
1046 001722 112767 000001 177204 67$:  MOVB   #1, $AUTOB        ;;SET AUTO-MODE INDICATOR
1047 001730
1048 001730 000413                BR    HERE                ;;GET OVER THE ASCIZ
1049      ;;MSGID:  .ASCIZ <CRLF>/MAINDEC-11-DZTAA-C/<CRLF>
1050 001760      HERE:
  
```

```

1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062 001760 010504
1063 001762 005305
1064 001764 002406
1065 001766 004737 012222
1066 001772 005305
1067 001774 002402
1068 001776 004737 012332
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086 002002 012737 002020 000004
1087 002010 005300
1088 002012 005777 177170
1089 002016 000402
1090 002020 005200 15:
1091 002022 022626
1092 002024 012737 000006 000004 25:
1093 002032 005700
1094 002034 001412
1095 002036 104201
1096 002040 012705 000002
1097 002044 005704
1098 002046 001344
1099 002050 013700 000042
1100 002054 001741
1101 002056 000137 011200
1102 002062

```

```

*****
*****
: THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
:
: R5=3  MANUAL OPERATIONS
: R5=2  ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
: R5=1  ASK FOR DRIVE(S)
: R5=0  DON'T ASK FOR ANYTHING
*****
BEGINX: MOV  R5,R4          ; COPY R5
        DEC  R5            ; ASK FOR DRIVES?
        BLT  CHKADR       ; BR IF NO
        JSR  PC,@ASKDRV   ; GO GET DRIVES TO BE TESTED
        DEC  R5            ; ASK FOR ADDRESSES?
        BLT  CHKADR       ; BR IF NO
        JSR  PC,@ASKADR   ; GO GET TAIL ADDRESSES
*****
: CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
: I.  TIMEOUT OCCURRED
:     A. TYPE ERROR MESSAGE
:     B. EXAMINE R4
:         1. R4>0 GOTO BEGINX
:         2. R4=0 EXAMINE (42)
:             A. (42)=0 GOTO BEGINX
:             B. (42)>0 GOTO SENDAD
: II. TIMEOUT DIDN'T OCCUR
:     A. CONTINUE
*****
CHKADR: MOV  #15,@ERRVEC   ; IN CASE OF TIMEOUTS
        CLR  R0            ; USE AS A SWITCH
        TST  @TACSL       ; SEE IF TAIL RESPONDS
        BR   25           ; BR IF NO TIMEOUT
        INC  R0            ; COME HERE ON TIMEOUT
        CMP  (SP)+,(SP)+  ; CLEANUP THE STACK
        MOV  #ERRVEC+2,@ERRVEC ; RESTORE TIMEOUT VECTOR
        TST  R0           ; DID A TIMEOUT OCCUR?
        BEQ  35           ; BR IF NO
        ERROR 201         ; TAIL FAILED TO RESPOND
        MOV  #2,R5        ; DRIVES & ADDRESSES
        TST  R4           ; OPERATOR INPUTS?
        BNE  BEGINX       ; BR IF YES
        MOV  @#42,R0      ; GET MONITOR RETURN ADDRESS
        BEQ  BEGINX       ; BR IF NO MONITOR
        JMP  @SENDAD      ; GO TO END
35:

```

```

1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1119
1119
1120
1121
1122
1123
1124 002062 012700 001224
1125 002066 004737 013672
1126 002072 000410
1127 002074 116010 000001
1128 002100 001412
1129 002102 004737 013672
1130 002106 000407
1131 002110 005010
1132 002112 000405
1133 002114 005200
1134 002116 004737 013672
1135 002122 000401
1136 002124 105010
1137 002126 012700 001224
1138 002132 010037 001230
1139 002136 121060 000001
1140 002142 001002
1141 002144 105060 000001
1142 002150 005710
1143 002152 001401
1144 002154 000412
1145 002156 104202
1146 002160 012705 000002
1147 002164 005704
1148 002166 001274
1149 002170 013700 000042
1150 002174 001671
1151 002176 000137 011200
1152 002202 020427 000003
1153 002206 001002
1154 002210 016704 175563
1155 002214 010437 001232
1156 002220 000405
1157 002222 104401 001732
1158 002226 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     #DRVKEY,RO      ;PICKUP ADDRESS OF ASCII DRIVE KEY
        JSR     PC,#EXAM      ;GO EXAMINE FIRST DRIVE
        BR      1$           ;OK TO TEST---GO CHECK NEXT
        MOVB   1(RO),(RO)     ;REPLACE 1ST WITH 2ND
        BEQ    2$           ;BR IF NO 2ND DRIVE SELECTED
        JSR     PC,#EXAM      ;GO EXAMINE DRIVE
        BR      2$           ;OK TO TEST
        CLR    (RO)         ;CLEAR DRIVE CODES
        BR      2$
1$: INC   RO                ;POINT TO 2ND
        JSR     PC,#EXAM      ;GO EXAMINE DRIVE
        BR      2$           ;OK TO TEST
        CLRB   (RO)         ;CLEAR 2ND
        BR      2$           ;RESET ADDRESS POINTERS
2$: MOV   #DRVKEY,RO
        MOV   RO,#DRVPT
        CMPB  (RO),1(RO)     ;1ST = 2ND?
        BNE   3$           ;BR IF NO
        CLRB  1(RO)         ;YES---CLEAR 2ND
        TST   (RO)         ;ANY DRIVES?
        BEQ   5$           ;BR IF NO
        BR   MANUAL
5$: ERROR 202              ;NO DRIVES AVAILABLE
        MOV   #2,R5         ;DRIVES & ADDRESS
        TST   R4           ;OPERATOR INPUTS?
        BNE   BEGINX       ;BR IF YES
        MOV   #42,RO        ;GET MONITOR RETURN ADDRESS
        BEQ   BEGINX       ;NO MONITOR
        JMP   #SENDAD      ;GO TO END
MANUAL: CMP   R4,#3
        BNE   OK
        MOV   -1,R4
OK:     MOV   R4,#ASKKEY
        BR   START
PWRST: TYPE  MSGID         ;POWER FAIL RESTART
        MOV   #DRVKEY,#DRVPT

```

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
 DZTRAC.P11 GET VALUE FOR SOFTWARE SWITCH REGISTER

```

1159 002234 013777 001220 176754 START: MOV      @TAVEC+2,@TAVEC      ;SETUP TA11 TRAP VECTOR
1160 002242 005077 176752          CLR      @TAVEC+2
1161 002246 012737 000340 177776          MOV      #340,@#PS      ;LOCKOUT ALL I/O INT
1162 002254 013704 001206          MOV      @TACSL,TACS    ;SETUP TACS
1163 002260 013705 001212          MOV      @TADBL,TADB    ;SETUP TADB
1164 002264 005737 001100          TST     @#SPASS        ;IF FIRST PASS SETLP FOR EXTRA LONG WAIT LOOPS
1165 002270 001024          BNE     1$             ;OTHERWISE USE OLD VALUES
1166 002272 012737 077777 012116          MOV      @CBIT15,@#MAXCNT
1167 002300 005737 000042          TST     @#42          ;UNDER MONITOR CONTROL?
1168 002304 001016          BNE     1$             ;SKIP TYPEOUT IF YES
1169 002306 104401 002314          TYPE    ,65$          ;;TYPE ASCIZ STRING
1170 002312 000413          BR      64$          ;;GET OVER THE ASCIZ
1171          ;;65$: .ASCIZ <15><12>/'DO A MANUAL REWIND/'
1172          64$:
1173 002342 005037 001102          1$: CLR      @#STSTNM    ;ZERO THE TEST NUMBER
1174 002346 005003          CLR      DRIVE        ;SET DRIVE TO "A"
1175 002350 013701 001230          MOV      @#DRVPT,R1    ;GET DRIVE POINTER
1176 002354 121127 000101          CMPB    (R1),#'A       ;IS IT DRIVE "A"?
1177 002360 001402          BEQ     TDRV          ;BR IF YES
1178 002362 012703 000400          MOV      @UNIT,DRIVE   ;SET DRIVE TO "B"
1179 002366          TDRV:
1180 002366 104401 002374          TYPE    ,65$          ;;TYPE ASCIZ STRING
1181 002372 000411          BR      64$          ;;GET OVER THE ASCIZ
1182          ;;65$: .ASCIZ <15><12>*TESTING DRIVE *
1183          64$:
1184 002416          MOVB    (R1)+,CURDRV   ;SETUP TO TYPE CURRENT DRIVE
1185 002422 104401 001234          TYPE    ,CURDRV
1186 002426 104401 001177          TYPE    ,SCRLF        ;TYPE A CR & LF
1187 002432 105711          I$TB    (R1)          ;LAST DRIVE BEEN SELECTED
1188 002434 001002          BNE     1$             ;BR IF NO
1189 002436 012701 001224          MOV      @#DRVKEY,R1   ;RESET DRIVE POINTER
1190 002442 010137 001230          1$: MOV      R1,@#DRVPT ;SAVE DRIVE POINTER FOR NEXT TIME
1191 002446 005737 001232          TST     @#ASKKEY      ;GO START TESTING IF NO MANUAL
1192 002452 002007          BGE     2$             ;OPERATIONS REQUESTED
1193 002454 005000          CLR     R0
1194 002456 000000          HALT
1195 002460 022767 000176 176452          CMP     @SWREG,SWR     ;GIVE CONTROL TO THE OPERATOR
1196 002466 001001          BNE     20$          ;SOFTWARE SWITCH REG.?
1197 002470 104405          GTSWR    20$          ;NO - LET HIM GO
1198          ;GET NEW SWR VALUE
1199          20$:
1200 002472 005737 000042          ;THIS CODE IS FOR ACT11 & DOP
1201 002476 001406          2$: TST     @#42          ;IS THERE A MONITOR?
1202 002500 010314          BEQ     TST1          ;GO START TESTING IF NO
1203 002502 112714 000017          MOV      DRIVE,@TACS   ;IF YES SELECT DRIVE
1204 002506 032714 000040          MOVB    @REWIND!GO,@TACS ;SEND TAPE TO BOT
1205 002512 001775          3$: BIT     @READY,@TACS ;WAIT ON READY
          BEQ     3$          ;FALL THRU IF READY=1
    
```

```

1206 : ////////////////////////////////////////////////////////////////////
1207 : ////////////////////////////////////////////////////////////////////
1208 : THE FOLLOWING TESTS WILL CHECK THE TA11
1209 : STATUS (TACS) AND DATA BUFFER (TADB) REGISTERS
1210 : TO INSURE THAT THEY RESPOND TO ADDRESSING.
1211 : ////////////////////////////////////////////////////////////////////
1212 :
1213 ; *****
1214 ; TA11 REGISTER ADDRESS TEST
1215 ; INSURE THAT TACS RESPONDS TO ADDRESSING
1216 : *****
1217 ; *TEST 1 TEST IF TACS EXIST
1218 : *****
1219 ; TST1: SCOPE
1220 002514 000004 MOV #1,STIMES ;;DO 1 ITERATION
1221 002516 012767 000001 176442 MOV #STACK,SP ;SETUP THE STACK POINTER
1222 002524 012706 001100 MOV #15,@#ERRVEC ;SETUP FOR BUS TIMEOUT
1223 002530 012737 002542 000004 CLR @TACS ;TACS ARE YOU THERE?
1224 002536 005014 BR 2$ ;YES -- NO TRAP
1225 002540 000420 1$: MOV #ERRVEC+2,@#ERRVEC ;RESTORE TRAP CATCHER
1226 002542 012737 000006 000004 CMP (SP)+,(SP)+ ;CLEAN UP THE STACK
1227 002550 022626 MOV @#EMTVEC,-(SP) ;SETUP FOR THIS ERROR CALL
1228 002552 013746 000030 MOV #ERROR1,@#EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB
1229 002556 012737 011516 000030 MOV #ERROR1,@#EMTVEC
1230 002564 010437 001122 MOV TACS,@#SBOADR ;GET THE FAILING ADDRESS
1231 002570 104010 ERROR 10 ;ERROR -- TIME OUT OCCURRED
1232 : WHEN ADDRESSING THE TACS REGISTER
1233 : RESTORE THE ERROR CALLS VECTOR
1234 : GO TO END OF PROGRAM
1235 2$: MOV #ERRVEC+2,@#ERRVEC ;RESTORE TRAP CATCHER
1236 : *****
1237 ; TA11 REGISTER ADDRESS TEST
1238 ; INSURE THAT TADB RESPONDS TO ADDRESSING
1239 : *****
1240 ; *TEST 2 TEST IF TADB EXIST
1241 : *****
1242 ; TST2: SCOPE
1243 002610 000004 MOV #1,STIMES ;;DO 1 ITERATION
1244 002612 012767 000001 176346 MOV #STACK,SP ;SETUP THE STACK POINTER
1245 002620 012706 001100 MOV #15,@#ERRVEC ;SETUP FOR BUS TIMEOUT
1246 002624 012737 002636 000004 CLR @TADB ;TADB ARE YOU THERE?
1247 002632 005015 BR 2$ ;YES -- NO TRAP
1248 002634 000420 1$: MOV #ERRVEC+2,@#ERRVEC ;RESTORE TRAP CATCHER
1249 002636 012737 000006 000004 CMP (SP)+,(SP)+ ;CLEAN UP THE STACK
1250 002644 022626 MOV @#EMTVEC,-(SP) ;SETUP FOR THIS ERROR CALL
1251 002646 013746 000030 MOV #ERROR1,@#EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB
1252 002652 012737 011516 000030 MOV #ERROR1,@#EMTVEC
1253 002660 010537 001122 MOV TADB,@#SBOADR ;GET THE FAILING ADDRESS
1254 002664 104010 ERROR 10 ;ERROR -- TIME OUT OCCURRED
1255 : WHEN ADDRESSING THE TADB REGISTER
1256 : RESTORE THE ERROR CALLS VECTOR
1257 2$: MOV #ERRVEC+2,@#ERRVEC ;RESTORE TRAP CATCHER

```



```

1258
1259 ;*****
1260 ;TA11 REGISTER ADDRESS TEST
1261 ;INSURE THAT TACS RESPONDS TO ADDRESSING
1262 ;*****
1263 ;*TEST 3 TEST FOR LOW BYTE OF TACS
1264 ;*****
1265 002704 000004 TST3: SCOPE
1266 002706 012767 000001 176252 MOV #1,STIMES ;;DO 1 ITERATION
1267 002714 012706 001100 MOV #STACK,SP ;SETUP THE STACK POINTER
1268 002720 012737 002732 000004 MOV #IS,@ERRVEC ;SETUP FOR BUS TIMEOUT
1269 002726 105014 CLRB @TACS ;TACS ARE YOU THERE?
1270 002730 000420 BR 2$ ;YES -- NO TRAP
1271 002732 012737 000006 000004 1$: MOV #ERRVEC+2,@ERRVEC ;RESTORE TRAP CATCHER
1272 002740 022626 CMP (SP)+,(SP)+ ;CLEAN UP THE STACK
1273 002742 013746 000030 MOV @EMTVEC,-(SP) ;SETUP FOR THIS ERROR CALL
1274 002746 012737 011516 000030 MOV #ERROR1,@EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB
1275 002754 010437 001122 MOV TACS,@$BDAOR ;GET THE FAILING ADDRESS
1276 002760 104010 ERROR 10 ;ERROR -- TIME OUT OCCURRED
1277 ;WHEN ADDRESSING THE TACS REGISTER
1278 002762 012637 000030 MOV (SP)+,@EMTVEC ;RESTORE THE ERROR CALLS VECTOR
1279 002766 000137 011124 JMP @SEOP ;GO TO END OF PROGRAM
1280 002772 012737 000006 000004 2$: MOV #ERRVEC+2,@ERRVEC ;RESTORE TRAP CATCHER
1281
1282 ;*****
1283 ;TA11 REGISTER ADDRESS TEST
1284 ;INSURE THAT TADB RESPONDS TO ADDRESSING
1285 ;*****
1286 ;*TEST 4 TEST FOR LOW BYTE OF TADB
1287 ;*****
1288 003000 000004 TST4: SCOPE
1289 003002 012767 000001 176156 MOV #1,STIMES ;;DO 1 ITERATION
1290 003010 012706 001100 MOV #STACK,SP ;SETUP THE STACK POINTER
1291 003014 012737 003026 000004 MOV #IS,@ERRVEC ;SETUP FOR BUS TIMEOUT
1292 003022 105015 CLRB @TADB ;TADB ARE YOU THERE?
1293 003024 000420 BR 2$ ;YES -- NO TRAP
1294 003026 012737 000006 000004 1$: MOV #ERRVEC+2,@ERRVEC ;RESTORE TRAP CATCHER
1295 003034 022626 CMP (SP)+,(SP)+ ;CLEAN UP THE STACK
1296 003036 013746 000030 MOV @EMTVEC,-(SP) ;SETUP FOR THIS ERROR CALL
1297 003042 012737 011516 000030 MOV #ERROR1,@EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB
1298 003050 010537 001122 MOV TADB,@$BDAOR ;GET THE FAILING ADDRESS
1299 003054 104010 ERROR 10 ;ERROR -- TIME OUT OCCURRED
1300 ;WHEN ADDRESSING THE TADB REGISTER
1301 003056 012637 000030 MOV (SP)+,@EMTVEC ;RESTORE THE ERROR CALLS VECTOR
1302 003062 000137 011124 JMP @SEOP ;GO TO END OF PROGRAM
1303 003066 012737 000006 000004 2$: MOV #ERRVEC+2,@ERRVEC ;RESTORE TRAP CATCHER
1304
1305 ;*****
1306 ;TA11 REGISTER ADDRESS TEST
1307 ;INSURE THAT TACSH RESPONDS TO ADDRESSING
1308 ;*****
1309 ;*TEST 5 TEST FOR HIGH BYTE OF TACS
1310 ;*****
1311 003074 000004 TST5: SCOPE
1312 003076 012767 000001 176062 MOV #1,STIMES ;;DO 1 ITERATION
1313 003104 012706 001100 MOV #STACK,SP ;SETUP THE STACK POINTER

```

T11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAAC.P11 T5 TEST FOR HIGH BYTE OF TACS

```

1314 003110 012737 003124 000004      MOV      #15, @#ERRVEC      ;SETUP FOR BUS TIMEOUT
1315 003116 105077 176066              CLRB     @TACSH             ;TACSH ARE YOU THERE?
1316 003122 000421                      BR       2$                ;YES -- NO TRAP
1317 003124 012737 000006 000004 1$:  MOV      #ERRVEC+2, @#ERRVEC ;RESTORE TRAP CATCHER
1318 003132 022626                      CMP      (SP)+, (SP)+      ;CLEAN UP THE STACK
1319 003134 013746 000030              MOV      @#EMTVEC, -(SP)   ;SETUP FOR THIS ERROR CALL
1320 003140 012737 011516 000030      MOV      #ERROR1, @#EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB
1321 003146 016737 176036 001122      MOV      TACSH, @#SBDADR   ;GET THE FAILING ADDRESS
1322 003154 104010                      ERROR    10                ;ERROR -- TIME OUT OCCURRED
1323                                     ;WHEN ADDRESSING THE TACSH REGISTER
1324 003156 012637 000030              MOV      (SP)+, @#EMTVEC   ;RESTORE THE ERROR CALLS VECTOR
1325 003162 000137 011124              JMP      @#SEOP            ;GO TO END OF PROGRAM
1326 003166 012737 000006 000004 2$:  MOV      #ERRVEC+2, @#ERRVEC ;RESTORE TRAP CATCHER
1327
1328                                     ;*****
1329                                     ;T11 REGISTER ADDRESS TEST
1330                                     ;INSURE THAT TADBH RESPONDS TO ADDRESSING
1331                                     ;*****
1332 *TEST 6 TEST FOR HIGH BYTE OF TADB
1333 ;*****
1334 003174 000004      TST6:  SCOPE
1335 003176 012767 000001 175762      MOV      #1, $TIMES      ;;DO 1 ITERATION
1336 003204 012706 001100              MOV      #STACK, SP      ;SETUP THE STACK POINTER
1337 003210 012737 003224 000004      MOV      #15, @#ERRVEC   ;SETUP FOR BUS TIMEOUT
1338 003216 105077 175772              CLRB     @TADBH          ;TADBH ARE YOU THERE?
1339 003222 000421                      BR       2$                ;YES -- NO TRAP
1340 003224 012737 000006 000004 1$:  MOV      #ERRVEC+2, @#ERRVEC ;RESTORE TRAP CATCHER
1341 003232 022626                      CMP      (SP)+, (SP)+      ;CLEAN UP THE STACK
1342 003234 013746 000030              MOV      @#EMTVEC, -(SP)   ;SETUP FOR THIS ERROR CALL
1343 003240 012737 011516 000030      MOV      #ERROR1, @#EMTVEC ;SKIP OVER THE SAVE OF TACS AND TADB
1344 003246 016737 175742 001122      MOV      TADBH, @#SBDADR   ;GET THE FAILING ADDRESS
1345 003254 104010                      ERROR    10                ;ERROR -- TIME OUT OCCURRED
1346                                     ;WHEN ADDRESSING THE TADBH REGISTER
1347 003256 012637 000030              MOV      (SP)+, @#EMTVEC   ;RESTORE THE ERROR CALLS VECTOR
1348 003262 000137 011124              JMP      @#SEOP            ;GO TO END OF PROGRAM
1349 003266 012737 000006 000004 2$:  MOV      #ERRVEC+2, @#ERRVEC ;RESTORE TRAP CATCHER
1350

```

```

1351
1352
1353
1354
1355
1356
1357
1358
1359 003274 000004
1360 003276 012767 003306 175602
1361 003304 000005
1362 003306 032714 040000
1363 003312 001401
1364 003314 104001
1365
1366
1367
1368 003316 000004
1369 003320 012767 003330 175560
1370 003326 000005
1371 003330 032714 004000
1372 003334 001401
1373 003336 104001
1374
1375
1376
1377 003340 000004
1378 003342 012767 003352 175536
1379 003350 000005
1380 003352 032714 002000
1381 003356 001401
1382 003360 104001
1383
1384
1385
1386 003362 000004
1387 003364 012767 003374 175514
1388 003372 000005
1389 003374 032714 000400
1390 003400 001401
1391 003402 104001
1392
1393
1394
1395 003404 000004
1396 003406 012767 003416 175472
1397 003414 000005
1398 003416 032714 000200
1399 003422 001401
1400 003424 104001
1401
1402
1403
1404 003426 000004
1405 003430 012767 003440 175450
1406 003436 000005

```

```

: ////////////////////////////////////////////////////////////////////
: ////////////////////////////////////////////////////////////////////
: THE FOLLOWING TESTS WILL CHECK THE INITIAL STATE OF ALL BITS
: IN THE TACS THAT ARE NOT DRIVE DEPENDENT.
: ////////////////////////////////////////////////////////////////////
: *****
: *TEST 7 TEST INITIAL CONDITION OF TACS BIT14 = 0
: *****
†ST7: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET
1$: BIT #BIT14,@TACS ;IS BIT14 OF TACS = 0?
BEQ TST10 ;;BR IF YES
ERROR 1 ;ERROR -- BIT14 OF TACS DIDN'T INITIALIZE PROPER
: *****
: *TEST 10 TEST INITIAL CONDITION OF TACS BIT11 = 0
: *****
†ST10: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET
1$: BIT #BIT11,@TACS ;IS BIT11 OF TACS = 0?
BEQ TST11 ;;BR IF YES
ERROR 1 ;ERROR -- BIT11 OF TACS DIDN'T INITIALIZE PROPER
: *****
: *TEST 11 TEST INITIAL CONDITION OF TACS BIT10 = 0
: *****
†ST11: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET
1$: BIT #BIT10,@TACS ;IS BIT10 OF TACS = 0?
BEQ TST12 ;;BR IF YES
ERROR 1 ;ERROR -- BIT10 OF TACS DIDN'T INITIALIZE PROPER
: *****
: *TEST 12 TEST INITIAL CONDITION OF TACS BIT08 = 0
: *****
†ST12: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET
1$: BIT #BIT08,@TACS ;IS BIT08 OF TACS = 0?
BEQ TST13 ;;BR IF YES
ERROR 1 ;ERROR -- BIT08 OF TACS DIDN'T INITIALIZE PROPER
: *****
: *TEST 13 TEST INITIAL CONDITION OF TACS BIT07 = 0
: *****
†ST13: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET
1$: BIT #BIT07,@TACS ;IS BIT07 OF TACS = 0?
BEQ TST14 ;;BR IF YES
ERROR 1 ;ERROR -- BIT07 OF TACS DIDN'T INITIALIZE PROPER
: *****
: *TEST 14 TEST INITIAL CONDITION OF TACS BIT06 = 0
: *****
†ST14: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET

```

```

1407 003440 032714 000100 15: BIT #BIT06,@TACS ;IS BIT06 OF TACS = 0?
1408 003444 001401 BEQ TST15 ;;BR IF YES
1409 003446 104001 ERROR 1 ;ERROR -- BIT06 OF TACS DIDN'T INITIALIZE PROPER
1410 ;*****
1411 ;*TEST 15 TEST INITIAL CONDITION OF TACS BIT04 = 0
1412 ;*****
1413 003450 000004 TST15: SCOPE
1414 003452 012767 003462 175426 MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
1415 003460 000005 RESET
1416 003462 032714 000020 15: BIT #BIT04,@TACS ;IS BIT04 OF TACS = 0?
1417 003466 001401 BEQ TST16 ;;BR IF YES
1418 003470 104001 ERROR 1 ;ERROR -- BIT04 OF TACS DIDN'T INITIALIZE PROPER
1419 ;*****
1420 ;*TEST 16 TEST INITIAL CONDITION OF TACS BIT03 = 0
1421 ;*****
1422 003472 000004 TST16: SCOPE
1423 003474 012767 003504 175404 MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
1424 003502 000005 RESET
1425 003504 032714 000010 15: BIT #BIT03,@TACS ;IS BIT03 OF TACS = 0?
1426 003510 001401 BEQ TST17 ;;BR IF YES
1427 003512 104001 ERROR 1 ;ERROR -- BIT03 OF TACS DIDN'T INITIALIZE PROPER
1428 ;*****
1429 ;*TEST 17 TEST INITIAL CONDITION OF TACS BIT02 = 0
1430 ;*****
1431 003514 000004 TST17: SCOPE
1432 003516 012767 003526 175362 MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
1433 003524 000005 RESET
1434 003526 032714 000004 15: BIT #BIT02,@TACS ;IS BIT02 OF TACS = 0?
1435 003532 001401 BEQ TST20 ;;BR IF YES
1436 003534 104001 ERROR 1 ;ERROR -- BIT02 OF TACS DIDN'T INITIALIZE PROPER
1437 ;*****
1438 ;*TEST 20 TEST INITIAL CONDITION OF TACS BIT01 = 0
1439 ;*****
1440 003536 000004 TST20: SCOPE
1441 003540 012767 003550 175340 MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
1442 003546 000005 RESET
1443 003550 032714 000002 15: BIT #BIT01,@TACS ;IS BIT01 OF TACS = 0?
1444 003554 001401 BEQ TST21 ;;BR IF YES
1445 003556 104001 ERROR 1 ;ERROR -- BIT01 OF TACS DIDN'T INITIALIZE PROPER
1446 ;*****
1447 ;*TEST 21 TEST INITIAL CONDITION OF TACS BIT00 = 0
1448 ;*****
1449 003560 000004 TST21: SCOPE
1450 003562 012767 003572 175316 MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
1451 003570 000005 RESET
1452 003572 032714 000001 15: BIT #BIT00,@TACS ;IS BIT00 OF TACS = 0?
1453 003576 001401 BEQ TST22 ;;BR IF YES
1454 003600 104001 ERROR 1 ;ERROR -- BIT00 OF TACS DIDN'T INITIALIZE PROPER
1455

```

```

1456 :////////////////////////////////////////////////////////////////////
1457 :////////////////////////////////////////////////////////////////////
1458 : THE FOLLOWING TESTS VALIDATES EACH READ/WRITE
1459 : BIT OF THE TACS FOR SETTING, CLEARING,
1460 : AND INITIALIZING
1461 :////////////////////////////////////////////////////////////////////
1462 :*****
1463 :*TEST 22 TEST BIT08 OF TACS MAY BE "SET" AND "CLEARED"
1464 :*****
1465 003602 000004
1466 003604 052714 000400
1467 003610 032714 000400
1468 003614 001001
1469 003616 104001
1470
1471 003620 042714 000400
1472 003624 032714 000400
1473 003630 001401
1474 003632 104001
1475
1476 :*****
1477 :*TEST 23 TEST BIT08 OF TACS INITIALIZE TO ZERO
1478 :*****
1478 003634 000004
1479 003636 012767 000005 175322
1480 003644 012714 000400
1481 003650 032714 000400
1482 003654 001001
1483 003656 104001
1484
1485 003660 000005
1486 003662 032714 000400
1487 003666 001401
1488 003670 104001
1489
1490 :*****
1491 :*TEST 24 TEST BIT06 OF TACS MAY BE "SET" AND "CLEARED"
1492 :*****
1492 003672 000004
1493 003674 052714 000100
1494 003700 032714 000100
1495 003704 001001
1496 003706 104001
1497
1498 003710 042714 000100
1499 003714 032714 000100
1500 003720 001401
1501 003722 104001
1502
1503 :*****
1504 :*TEST 25 TEST BIT06 OF TACS INITIALIZE TO ZERO
1505 :*****
1505 003724 000004
1506 003726 012767 000005 175232
1507 003734 012714 000100
1508 003740 032714 000100
1509 003744 001001
1510 003746 104001
1511

```

K03

```

1512 003750 000005      2$:  RESET                ;CLEAR THE WORLD
1513 003752 032714 000100      BIT      #BIT06,@TACS      ;DID BIT06 CLEAR?
1514 003756 001401      BEQ      TST26            ;;BR IF YES
1515 003760 104001      ERROR    1                ;ERROR -- RESET DIDN'T CLEAR BIT06
1516                                     ;*****
1517 ;*TEST 26      TEST BIT04 OF TACS MAY BE "SET" AND "CLEARED"
1518 ;*****
1519 003762 000004      †TST26:  SCOPE
1520 003764 052714 000020      BIS      #BIT04,@TACS      ;SET BIT04 OF TACS
1521 003770 032714 000020      BIT      #BIT04,@TACS      ;DID BIT04 SET?
1522 003774 001001      BNE     1$                ;BR IF YES
1523 003776 104001      ERROR    1                ;BIT04 FAILED TO SET
1524
1525 004000 042714 000020      1$:    BIC      #BIT04,@TACS      ;NOW CLEAR IT
1526 004004 032714 000020      BIT      #BIT04,@TACS      ;DID BIT04 CLEAR?
1527 004010 001401      BEQ      TST27            ;;BR IF YES
1528 004012 104001      ERROR    1                ;ERROR -- BIT04 OF TACS FAILED TO CLEAR
1529
1530 ;*TEST 27      TEST BIT04 OF TACS INITIALIZE TO ZERO
1531 ;*****
1532 004014 000004      †TST27:  SCOPE
1533 004016 012767 000005 175142      MOV      #5,$TIMES        ;;DO 5 ITERATIONS
1534 004024 012714 000020      1$:    MOV      #BIT04,@TACS      ;SET BIT04 OF TACS
1535 004030 032714 000020      BIT      #BIT04,@TACS      ;DID BIT04 SET?
1536 004034 001001      BNE     2$                ;BR IF YES
1537 004036 104001      ERROR    1                ;BIT04 FAILED TO SET
1538
1539 004040 000005      2$:    RESET                ;CLEAR THE WORLD
1540 004042 032714 000020      BIT      #BIT04,@TACS      ;DID BIT04 CLEAR?
1541 004046 001401      BEQ      TST30            ;;BR IF YES
1542 004050 104001      ERROR    1                ;ERROR -- RESET DIDN'T CLEAR BIT04
1543
1544 ;*TEST 30      TEST BIT03 OF TACS MAY BE "SET" AND "CLEARED"
1545 ;*****
1546 004052 000004      †TST30:  SCOPE
1547 004054 052714 000010      BIS      #BIT03,@TACS      ;SET BIT03 OF TACS
1548 004060 032714 000010      BIT      #BIT03,@TACS      ;DID BIT03 SET?
1549 004064 001001      BNE     1$                ;BR IF YES
1550 004066 104001      ERROR    1                ;BIT03 FAILED TO SET
1551
1552 004070 042714 000010      1$:    BIC      #BIT03,@TACS      ;NOW CLEAR IT
1553 004074 032714 000010      BIT      #BIT03,@TACS      ;DID BIT03 CLEAR?
1554 004100 001401      BEQ      TST31            ;;BR IF YES
1555 004102 104001      ERROR    1                ;ERROR -- BIT03 OF TACS FAILED TO CLEAR
1556
1557 ;*TEST 31      TEST BIT03 OF TACS INITIALIZE TO ZERO
1558 ;*****
1559 004104 000004      †TST31:  SCOPE
1560 004106 012767 000005 175052      MOV      #5,$TIMES        ;;DO 5 ITERATIONS
1561 004114 012714 000010      1$:    MOV      #BIT03,@TACS      ;SET BIT03 OF TACS
1562 004120 032714 000010      BIT      #BIT03,@TACS      ;DID BIT03 SET?
1563 004124 001001      BNE     2$                ;BR IF YES
1564 004126 104001      ERROR    1                ;BIT03 FAILED TO SET
1565
1566 004130 000005      2$:    RESET                ;CLEAR THE WORLD
1567 004132 032714 000010      BIT      #BIT03,@TACS      ;DID BIT03 CLEAR?

```

```

1568 004136 001401          BEQ     TST32          ;;BR IF YES
1569 004140 104001          ERROR    1          ;ERROR -- RESET DIDN'T CLEAR BIT03
1570                                     ;*****
1571                                     ;*TEST 32      TEST BIT02 OF TACS MAY BE "SET" AND "CLEARED"
1572                                     ;*****
1573 004142 000004          †TST32: SCOPE
1574 004144 052714 000004      BIS     #BIT02,@TACS      ;SET BIT02 OF TACS
1575 004150 032714 000004      BIT     #BIT02,@TACS      ;DID BIT02 SET?
1576 004154 001001          BNE     1$              ;BR IF YES
1577 004156 104001          ERROR    1          ;BIT02 FAILED TO SET
1578
1579 004160 042714 000004      1$:    BIC     #BIT02,@TACS      ;NOW CLEAR IT
1580 004164 032714 000004      BIT     #BIT02,@TACS      ;DID BIT02 CLEAR?
1581 004170 001401          BEQ     TST33          ;;BR IF YES
1582 004172 104001          ERROR    1          ;ERROR -- BIT02 OF TACS FAILED TO CLEAR
1583                                     ;*****
1584                                     ;*TEST 33      TEST BIT02 OF TACS INITIALIZE TO ZERO
1585                                     ;*****
1586 004174 000004          †TST33: SCOPE
1587 004176 012767 000005 174762  MOV     #5,$TIMES        ;;DO 5 ITERATIONS
1588 004204 012714 000004      1$:    MOV     #BIT02,@TACS      ;SET BIT02 OF TACS
1589 004210 032714 000004      BIT     #BIT02,@TACS      ;DID BIT02 SET?
1590 004214 001001          BNE     2$              ;BR IF YES
1591 004216 104001          ERROR    1          ;BIT02 FAILED TO SET
1592
1593 004220 000005          2$:    RESET          ;CLEAR THE WORLD
1594 004222 032714 000004      BIT     #BIT02,@TACS      ;DID BIT02 CLEAR?
1595 004226 001401          BEQ     TST34          ;;BR IF YES
1596 004230 104001          ERROR    1          ;ERROR -- RESET DIDN'T CLEAR BIT02
1597                                     ;*****
1598                                     ;*TEST 34      TEST BIT01 OF TACS MAY BE "SET" AND "CLEARED"
1599                                     ;*****
1600 004232 000004          †TST34: SCOPE
1601 004234 052714 000002      BIS     #BIT01,@TACS      ;SET BIT01 OF TACS
1602 004240 032714 000002      BIT     #BIT01,@TACS      ;DID BIT01 SET?
1603 004244 001001          BNE     1$              ;BR IF YES
1604 004246 104001          ERROR    1          ;BIT01 FAILED TO SET
1605
1606 004250 042714 000002      1$:    BIC     #BIT01,@TACS      ;NOW CLEAR IT
1607 004254 032714 000002      BIT     #BIT01,@TACS      ;DID BIT01 CLEAR?
1608 004260 001401          BEQ     TST35          ;;BR IF YES
1609 004262 104001          ERROR    1          ;ERROR -- BIT01 OF TACS FAILED TO CLEAR
1610                                     ;*****
1611                                     ;*TEST 35      TEST BIT01 OF TACS INITIALIZE TO ZERO
1612                                     ;*****
1613 004264 000004          †TST35: SCOPE
1614 004266 012767 000005 174672  MOV     #5,$TIMES        ;;DO 5 ITERATIONS
1615 004274 012714 000002      1$:    MOV     #BIT01,@TACS      ;SET BIT01 OF TACS
1616 004300 032714 000002      BIT     #BIT01,@TACS      ;DID BIT01 SET?
1617 004304 001001          BNE     2$              ;BR IF YES
1618 004306 104001          ERROR    1          ;BIT01 FAILED TO SET
1619
1620 004310 000005          2$:    RESET          ;CLEAR THE WORLD
1621 004312 032714 000002      BIT     #BIT01,@TACS      ;DID BIT01 CLEAR?
1622 004316 001401          BEQ     TST36          ;;BR IF YES
1623 004320 104001          ERROR    1          ;ERROR -- RESET DIDN'T CLEAR BIT01

```

```

1624
1625
1626
1627 004322 000004
1628 004324 012767 004370 174636
1629 004332 005014
1630 004334 005000
1631 004336 012702 000002
1632 004342 011401
1633 004344 042701 177741
1634 004350 020100
1635 004352 001401
1636 004354 104005
1637 004356 060200
1638 004360 060214
1639 004362 032700 000036
1640 004366 001365
1641

```

```

*****
*TEST 36 COUNT PATTERN TO TACS<04:01>
*****
TST36: SCOPE
MOV #TST37,$ESCAPE ;;ESCAPE TO TEST 37 ON ERROR
CLR @TACS ;SETUP TACS FOR COUNTING
CLR R0 ;SETUP R0 FOR COUNTING
MOV #2,R2 ;SET THE INCREMENT
1$: MOV @TACS,R1 ;GET TACS
BIC #T36,R1 ;CLEAR THE JUNK
CMP R1,R0 ;COUNT OK?
BEQ 2$ ;BR IF YES
ERROR 5 ;ERROR--BAD COUNT
2$: ADD R2,R0 ;UPDATE R0
ADD R2,@TACS ;COUNT TACS
BIT #36,R0 ;READY?
BNE 1$ ;NO--GO DO TEST COUNT PATTERN

```



```

1642 ;////////////////////////////////////
1643 ;////////////////////////////////////
1644 ; THE FOLLOWING TESTS WILL CHECK THE INITIAL STATE OF ALL BITS
1645 ; IN THE TACS THAT ARE DRIVE AND FUNCTION DEPENDENT.
1646 ; NOTE: FOR ALL OF THESE TEST TO PASS
1647 ; THE DRIVE UNDERGOING TEST MUST BE
1648 ; ON CLEAR LEADER AND WRITE ENABLED.
1649 ;////////////////////////////////////
1650 ;*****
1651 ;*TEST 37 TEST INITIAL CONDITION OF TACS BIT05 = 1
1652 ;*****
1653 004370 000004 TST37: SCOPE
1654 004372 012767 004404 174506 MOV #1$, $LPADR ;;SET SCOPE LOOP ADDRESS
1655 004400 000005 RESET
1656 004402 010314 MOV DRIVE, @TACS ;SELECT DRIVE
1657 004404 032714 000040 1$: BIT #BIT05, @TACS ;IS BIT05 OF TACS = 1?
1658 004410 001001 BNE TST40 ;;BR IF YES
1659 004412 104001 ERROR 1 ;ERROR -- BIT05 OF TACS DIDN'T INITIALIZE PROPER
1660 ;*****
1661 ;*TEST 40 TEST INITIAL CONDITION OF TACS BIT09 = 0
1662 ;*****
1663 004414 000004 TST40: SCOPE
1664 004416 012767 004430 174462 MOV #1$, $LPADR ;;SET SCOPE LOOP ADDRESS
1665 004424 000005 RESET
1666 004426 010314 MOV DRIVE, @TACS ;SELECT DRIVE
1667 004430 032714 001000 1$: BIT #BIT09, @TACS ;IS BIT09 OF TACS = 0?
1668 004434 001401 BEQ TST41 ;;BR IF YES
1669 004436 104001 ERROR 1 ;ERROR -- BIT09 OF TACS DIDN'T INITIALIZE PROPER
1670 ;*****
1671 ;*TEST 41 TEST INITIAL CONDITION OF TACS BIT12 = 0
1672 ;*****
1673 004440 000004 TST41: SCOPE
1674 004442 012767 004460 174436 MOV #1$, $LPADR ;;SET SCOPE LOOP ADDRESS
1675 004450 000005 RESET
1676 004452 010314 MOV DRIVE, @TACS ;SELECT DRIVE
1677 004454 052714 000002 BIS #WRITE, @TACS ;LOAD WRITE FUNCTION
1678 004460 032714 010000 1$: BIT #BIT12, @TACS ;IS BIT12 OF TACS = 0?
1679 004464 001401 BEQ TST42 ;;BR IF YES
1680 004466 104001 ERROR 1 ;ERROR -- BIT12 OF TACS DIDN'T INITIALIZE PROPER
1681 ;*****
1682 ;*TEST 42 TEST INITIAL CONDITION OF TACS BIT13 = 1
1683 ;*****
1684 004470 000004 TST42: SCOPE
1685 004472 012767 004504 174406 MOV #1$, $LPADR ;;SET SCOPE LOOP ADDRESS
1686 004500 000005 RESET
1687 004502 010314 MOV DRIVE, @TACS ;SELECT DRIVE
1688 004504 032714 020000 1$: BIT #BIT13, @TACS ;IS BIT13 OF TACS = 1?
1689 004510 001001 BNE TST43 ;;BR IF YES
1690 004512 104001 ERROR 1 ;ERROR -- BIT13 OF TACS DIDN'T INITIALIZE PROPER
1691 ;*****
1692 ;*TEST 43 TEST INITIAL CONDITION OF TACS BIT15 = 1
1693 ;*****
1694 004514 000004 TST43: SCOPE
1695 004516 012767 004530 174362 MOV #1$, $LPADR ;;SET SCOPE LOOP ADDRESS
1696 004524 000005 RESET
1697 004526 010314 MOV DRIVE, @TACS ;SELECT DRIVE

```

804

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C

MACY11 27(732) 25-SEP-76 11:04 PAGE 41

DZTAA.C.P11

T43

TEST INITIAL CONDITION OF TACS BIT15 = 1

1698	004530	032714	100000
1699	004534	001001	
1700	004536	104001	

:S:

BIT	#BIT15,2TACS
BNE	TST44
ERROR	1

```

::BR IF :IS BIT15 OF TACS = 1?
        YES
        :FRPR -- BIT15 OF TACS DIDN'T INITIALIZE PROPER

```

```

1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712 004540 000004
1713 004542 012767 004552 174336
1714 004550 000005
1715 004552 011400
1716 004554 010001
1717 004556 005101
1718 004560 042700 077777
1719 004564 042701 077777
1720 004570 010114
1721 004572 011401
1722 004574 042701 077777
1723 004600 020001
1724 004602 001401
1725 004604 104001
1726
1727
1728
1729
1730
1731
1732 004606 000004
1733 004610 012767 004620 174270
1734 004616 000005
1735 004620 011400
1736 004622 010001
1737 004624 005101
1738 004626 042700 137777
1739 004632 042701 137777
1740 004636 010114
1741 004640 011401
1742 004642 042701 137777
1743 004646 020001
1744 004650 001401
1745 004652 104001
1746
1747
1748
1749
1750
1751
1752 004654 000004
1753 004656 012767 004666 174222
1754 004664 000005
1755 004666 011400
1756 004670 010001

```

```

////////////////////////////////////
////////////////////////////////////
THE FOLLOWING TESTS CHECKS EACH READ ONLY
BIT OF THE TACS FOR READ ONLY CAPABILITY
////////////////////////////////////
;*****
;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT15.
;THEN CHECK IT TO SEE IF IT CHANGED.
;*****
;TEST 44 TEST TACS BIT15 IS READ ONLY
;*****
TST44: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET ;;INITIALIZE ALL BITS
15: MOV @TACS,RO ;;GET TACS FOR COMPARING
MOV RO,R1 ;;PUT STATUS IN R1
COM R1 ;;COMPLEMENT R1
BIC #1<BIT15>,RO ;;GET RID OF UNDESIREC BITS
BIC #1<BIT15>,R1
MOV R1,@TACS ;;TRY TO CHANGE BIT15
MOV @TACS,R1 ;;READ IT BACK
BIC #1<BIT15>,R1 ;;ONLY LOOK AT BIT15
CMP RO,R1 ;;DID IT CHANGE?
BEQ TST45 ;;BR IF NO
ERROR 1 ;ERROR -- BIT15 CHANGED
;*****
;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT14.
;THEN CHECK IT TO SEE IF IT CHANGED.
;*****
;TEST 45 TEST TACS BIT14 IS READ ONLY
;*****
TST45: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET ;;INITIALIZE ALL BITS
15: MOV @TACS,RO ;;GET TACS FOR COMPARING
MOV RO,R1 ;;PUT STATUS IN R1
COM R1 ;;COMPLEMENT R1
BIC #1<BIT14>,RO ;;GET RID OF UNDESIREC BITS
BIC #1<BIT14>,R1
MOV R1,@TACS ;;TRY TO CHANGE BIT14
MOV @TACS,R1 ;;READ IT BACK
BIC #1<BIT14>,R1 ;;ONLY LOOK AT BIT14
CMP RO,R1 ;;DID IT CHANGE?
BEQ TST46 ;;BR IF NO
ERROR 1 ;ERROR -- BIT14 CHANGED
;*****
;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT13.
;THEN CHECK IT TO SEE IF IT CHANGED.
;*****
;TEST 46 TEST TACS BIT13 IS READ ONLY
;*****
TST46: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET ;;INITIALIZE ALL BITS
15: MOV @TACS,RO ;;GET TACS FOR COMPARING
MOV RO,R1 ;;PUT STATUS IN R1

```

```

1757 004672 005101          CUM      R1          ;COMPLEMENT R1
1758 004674 042700 157777  BIC      #1C<BIT13>,R0 ;GET RID OF UNDESIRE
1759 004700 042701 157777  BIC      #1C<BIT13>,R1
1760 004704 010114          MOV      R1,@TACS    ;TRY TO CHANGE BIT13
1761 004706 011401          MOV      @TACS,R1    ;READ IT BACK
1762 004710 042701 157777  BIC      #1C<BIT13>,R1 ;ONLY LOOK AT BIT13
1763 004714 020001          CMP      R0,R1      ;DID IT CHANGE?
1764 004716 001401          BEQ      TST47      ;;BR IF NO
1765 004720 104001          ERROR    1          ;ERROR -- BIT13 CHANG
;*****
;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT12.
;THEN CHECK IT TO SEE IF IT CHANGED.
;*****
;TEST 47      TEST TACS BIT12 IS READ ONLY
;*****
TST47: SCOPE
1772 004722 000004          MOV      #1$,$LPADR  ;;SET SCOPE LOOP ADDRESS
1773 004724 012767 004734 174154 RESET          ;INITIALIZE ALL BITS
1774 004732 000005          MOV      @TACS,R0   ;GET TACS FOR COMPARI
1775 004734 011400          MOV      R0,R1      ;PUT STATUS IN R1
1776 004736 010001          COM      R1          ;COMPLEMENT R1
1777 004740 005101          BIC      #1C<BIT12>,R0 ;GET RID OF UNDESIRE
1778 004742 042700 167777  BIC      #1C<BIT12>,R1
1779 004746 042701 167777  MOV      R1,@TACS    ;TRY TO CHANGE BIT12
1780 004752 010114          MOV      @TACS,R1    ;READ IT BACK
1781 004754 011401          BIC      #1C<BIT12>,R1 ;ONLY LOOK AT BIT12
1782 004756 042701 167777  CMP      R0,R1      ;DID IT CHANGE?
1783 004762 020001          BEQ      TST50      ;;BR IF NO
1784 004764 001401          ERROR    1          ;ERROR -- BIT12 CHANG
1785 004766 104001
;*****
;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT11.
;THEN CHECK IT TO SEE IF IT CHANGED.
;*****
;TEST 50      TEST TACS BIT11 IS READ ONLY
;*****
TST50: SCOPE
1792 004770 000004          MOV      #1$,$LPADR  ;;SET SCOPE LOOP ADDRESS
1793 004772 012767 005002 174106 RESET          ;INITIALIZE ALL BITS
1794 005000 000005          MOV      @TACS,R0   ;GET TACS FOR COMPARI
1795 005002 011400          MOV      R0,R1      ;PUT STATUS IN R1
1796 005004 010001          COM      R1          ;COMPLEMENT R1
1797 005006 005101          BIC      #1C<BIT11>,R0 ;GET RID OF UNDESIRE
1798 005010 042700 173777  BIC      #1C<BIT11>,R1
1799 005014 042701 173777  MOV      R1,@TACS    ;TRY TO CHANGE BIT11
1800 005020 010114          MOV      @TACS,R1    ;READ IT BACK
1801 005022 011401          BIC      #1C<BIT11>,R1 ;ONLY LOOK AT BIT11
1802 005024 042701 173777  CMP      R0,R1      ;DID IT CHANGE?
1803 005030 020001          BEQ      TST51      ;;BR IF NO
1804 005032 001401          ERROR    1          ;ERROR -- BIT11 CHANG
1805 005034 104001
;*****
;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT10.
;THEN CHECK IT TO SEE IF IT CHANGED.
;*****
;TEST 51      TEST TACS BIT10 IS READ ONLY
;*****
TST51: SCOPE
1812 005036 000004

```



TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAAC.P11 T54 TEST TACS BITOS IS READ ONLY

```

1869          ;*****
1870          ;*TEST 54      TEST TACS BITOS IS READ ONLY
1871          ;*****
1872 005220 000004          †T54: SCOPE
1873 005222 012767 005232 173656      MOV      #15,$LPADR      ;;SET SCOPE LOOP ADDRESS
1874 005230 000005          RESET          ;INITIALIZE ALL BITS
1875 005232 011400          1$:  MOV      @TACS,R0      ;GET TACS FOR COMPARING
1876 005234 010001          MOV      R0,R1      ;PUT STATUS IN R1
1877 005236 005101          COM      R1      ;COMPLEMENT R1
1878 005240 042700 177737          BIC      #@C<BITOS>,R0 ;GET RID OF UNDESIRED BITS
1879 005244 042701 177737          BIC      #@C<BITOS>,R1
1880 005250 010114          MOV      R1,@TACS      ;TRY TO CHANGE BITOS
1881 005252 011401          MOV      @TACS,R1      ;READ IT BACK
1882 005254 042701 177737          BIC      #@C<BITOS>,R1 ;ONLY LOOK AT BITOS
1883 005260 020001          CMP      R0,R1      ;DID IT CHANGE?
1884 005262 001401          BEQ     T55          ;;BR IF NO
1885 005264 104001          ERROR  1          ;ERROR -- BITOS CHANGED
1886

```

1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942

005266 000004  
005270 012767 005300 173610  
005276 000005  
005300 012714 000004  
005304 105077 173700  
005310 132714 000004  
005314 001001  
005316 104001  
  
005320 000004  
005322 012767 005332 173556  
005330 000005  
005332 012714 000400  
005336 105014  
005340 132777 000001 173642  
005346 001001  
005350 104001  
  
005352 000004  
005354 012767 005412 173606  
005362 005014  
005364 032714 000536  
005370 001401  
005372 104001  
005374 112777 000136 173606  
005402 032714 000136  
005406 001401  
005410 104001  
  
005412 000004  
005414 012767 005450 173546  
005422 005014  
005424 032714 000536  
005430 001401  
005432 104001  
005434 112714 000400  
005440 032714 000400  
005444 001401  
005446 104001

```

////////////////////////////////////
////////////////////////////////////
THE FOLLOWING TESTS WILL CHECK FOR DUAL AND/OR FAULTY SELECTION
OF THE TAIL ADDRESSES
////////////////////////////////////
*****
*TEST 55 TEST HIGH BYTE SELECTION OF TACS
*****
TST55: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET
1$: MOV #BIT02,@TACS ;SET TACS BIT02
CLR @TACSH ;CLEAR THE HIGH BYTE
BITB #BIT02,@TACS ;IS BIT02 STILL SET?
BNE TST56 ;;BR IF YES
ERROR 1 ;CLRB TO HIGH BYTE OF TACS CLEARED LOW BYTE
*****
*TEST 56 TEST LOW BYTE SELECTION OF TACS
*****
TST56: SCOPE
MOV #15,$LPADR ;;SET SCOPE LOOP ADDRESS
RESET
1$: MOV #BIT08,@TACS ;SET BIT08 OF TACS
CLR @TACS ;CLEAR THE LOW BYTE OF TACS
BITB #BIT00,@TACSH ;IS BIT08 OF TACS STILL SET?
BNE TST57 ;;BR IF YES
ERROR 1 ;CLRB TO LOW BYTE OF TACS CLEARED THE HIGH BYTE
*****
*TEST 57 TEST "OUT LOW" SIGNAL
*****
TST57: SCOPE
MOV #TST60,$ESCAPE ;;ESCAPE TO TEST 60 ON ERROR
CLR @TACS ;ZERO THE TAIL STATUS
BIT #UNIT!INT.EN!ILBS!FUNCTION,@TACS
BEQ 1$
ERROR 1 ;TACS DIDN'T CLEAR PROPERLY
1$: MOVB #INT.EN!ILBS!FUNCTION,@TACSH ;LOAD HIGH BYTE OF TACS
BIT #INT.EN!ILBS!FUNCTION,@TACS ;CHECK LOW BYTE
BEQ TST60 ;;BR IF LOW BYTE IS OK
ERROR 1 ;LOW BYTE IMPROPERLY SELECTED
*****
*TEST 60 TEST "OUT HIGH" SIGNAL
*****
TST60: SCOPE
MOV #TST61,$ESCAPE ;;ESCAPE TO TEST 61 ON ERROR
CLR @TACS ;ZERO THE TAIL STATUS
BIT #UNIT!INT.EN!ILBS!FUNCTION,@TACS
BEQ 1$
ERROR 1 ;TACS DIDN'T CLEAR PROPERLY
1$: MOVB #BIT08,@TACS ;LOAD LOW BYTE OF TACS
BIT #BIT08,@TACS ;DID BIT08 GET SET?
BEQ TST61 ;;BR IF NO
ERROR 1 ;HIGH BYTE IMPROPERLY SELECTED
*****
*TEST 61 TEST "SELECT 00" ISN'T STUCK
*****

```

```

1943 005450 000004 TST61: SCOPE
1944 005452 012767 005506 173510 MOV #TST62,$ESCAPE ;;ESCAPE TO TEST 62 ON ERROR
1945 005460 005014 CLR @TACS ;CLEAR TA11 STATUS
1946 005462 032714 000536 BIT #UNIT!INT.EN!ILBS!FUNCTION,@TACS
1947 005466 001401 BEQ 1$
1948 005470 104001 ERROR 1 ;TACS DIDN'T CLEAR PROPERLY
1949 005472 012715 177776 1$: MOV #1C1,@TADB ;LOAD THE DATA BUFFER
1950 005476 032714 000536 BIT #UNIT!INT.EN!ILBS!FUNCTION,@TACS
1951 005502 001401 BEQ TST62 ;;BR IF TACS DIDN'T CHANGE
1952 005504 104001 ERROR 1 ;TACS CHANGED WHEN LOADING TADB

```

```

1953
1954
1955 :*****
1956 :THE FOLLOWING TEST INSURES THAT THE STATE OF THE "ERROR" INDICATOR IS
1957 :PROPER FOR ALL FUNCTIONS WHEN THE TAPE IS AT "CLEAR LEADER"
1958 :*****
1959 :*TEST 62 TEST "ERROR" FOR "CLEAR LEADER"
1960 :*****

```

```

1961 005506 000004 TST62: SCOPE
1962 005510 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1963 005512 032714 020000 BIT #LEADER,@TACS ;CHECK FOR CLEAR LEADER
1964 005516 001001 BNE 1$ ;BR IF ON CLEAR LEADER
1965 005520 104001 ERROR 1 ;NOT ON CLEAR LEADER
1966 005522 112714 000000 1$: MOVB #WFG,@TACS ;CHECK "ERROR" WITH "WFG"
1967 005526 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1968 005530 100401 BMI 2$ ;BR IF "ERROR" = 1
1969 005532 104001 ERROR 1 ;"ERROR" NOT = 1
1970 005534 112714 000002 2$: MOVB #WRITE,@TACS ;CHECK "ERROR" WITH "WRITE"
1971 005540 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1972 005542 100401 BMI 3$ ;BR IF "ERROR" = 1
1973 005544 104001 ERROR 1 ;"ERROR" NOT = 1
1974 005546 112714 000004 3$: MOVB #READ,@TACS ;CHECK "ERROR" WITH "READ"
1975 005552 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1976 005554 100401 BMI 4$ ;BR IF "ERROR" = 1
1977 005556 104001 ERROR 1 ;"ERROR" NOT = 1
1978 005560 112714 000006 4$: MOVB #BSFG,@TACS ;CHECK "ERROR" WITH "BSFG"
1979 005564 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1980 005566 100401 BMI 5$ ;BR IF "ERROR" = 1
1981 005570 104001 ERROR 1 ;"ERROR" NOT = 1
1982 005572 112714 000010 5$: MOVB #BSBG,@TACS ;CHECK "ERROR" WITH "BSBG"
1983 005576 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1984 005600 100401 BMI 6$ ;BR IF "ERROR" = 1
1985 005602 104001 ERROR 1 ;"ERROR" NOT = 1
1986 005604 112714 000012 6$: MOVB #SFFG,@TACS ;CHECK "ERROR" WITH "SFFG"
1987 005610 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1988 005612 100401 BMI 7$ ;BR IF "ERROR" = 1
1989 005614 104001 ERROR 1 ;"ERROR" NOT = 1
1990 005616 112714 000014 7$: MOVB #SFBG,@TACS ;CHECK "ERROR" WITH "SFBG"
1991 005622 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1992 005624 100401 BMI 8$ ;BR IF "ERROR" = 1
1993 005626 104001 ERROR 1 ;"ERROR" NOT = 1
1994 005630 112714 000016 8$: MOVB #REWIND,@TACS ;CHECK "ERROR" WITH "REWIND"
1995 005634 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1996 005636 100001 BPL TST63 ;;BR IF "ERROR" = 0
1997 005640 104001 ERROR 1 ;"ERROR" NOT = 0

```



1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053

005642 000004  
005644 012767 000005 173314  
005652 012767 005670 173226  
005660 000005  
005662 012767 005742 173300  
005670 010314  
005672 005001  
005674 012700 000144  
005700 112714 000017  
005704 032714 00004C  
005710 001010  
005712 005201  
005714 001373  
005716 005300  
005720 001371  
005722 032714 000040  
005726 001001  
005730 104001  
005732 032714 020000  
005736 001001  
005740 104001  
  
005742 000004  
005744 012767 000005 173214  
005752 012767 005770 173126  
005760 000005  
005762 012767 006042 173200  
005770 010314  
005772 005001  
005774 012700 000144  
006000 112714 000001  
006004 032714 000040  
006010 001010  
006012 005201  
006014 001373  
006016 005300  
006020 001371  
006022 032714 000040  
006026 001001  
006030 104001  
006032 032714 020000  
006036 001401  
006040 104001

```

: ///////////////////////////////////////////////////////////////////
: ///////////////////////////////////////////////////////////////////
: THE FOLLOWING TESTS WILL CHECK BASIC OPERATIONS OF THE
: "REWIND" AND "WRITE FILE GAP" FUNCTIONS.
: ///////////////////////////////////////////////////////////////////
: *****
: *TEST 63 TEST THAT "READY" SETS WHEN "REWIND" IS COMPLETED
: *****
TST63: SCOPE
      MOV #5,$TIMES ;;DO 5 ITERATIONS
      MOV #3,$SLPADR ;;SET SCOPE LOOP ADDRESS
      RESET
      MOV #TST64,$ESCAPE ;;ESCAPE TO TEST 64 ON ERROR
3$: MOV DRIVE,$TACS ;SELECT DRIVE
      CLR R1 ;WASTE SOME TIME
      MOV #100.,R0
      MOVB #REWIND!GO,$TACS ;START A REWIND
1$: BIT #READY,$TACS ;SAMPLE READY
      BNE 2$ ;GET OUT IF READY IS SET
      INC R1
      BNE 1$
      DEC R0
      BNE 1$
      BIT #READY,$TACS ;IS "READY" BIT SET?
      BNE 2$ ;BR IF YES
      ERROR 1 ;ERROR--"READY" BIT DIDN'T SET
2$: BIT #LEADER,$TACS ;CHECK FOR CLEAR LEADER
      BNE TST64 ;;BR IF CLEAR LEADER NE 0
      ERROR 1 ;NOT ON CLEAR LEADER
      ;AFTER DOING A REWIND
: *****
: *TEST 64 TEST THAT "READY" SETS WHEN "WRITE FILE GAP" IS COMPLETED
: *****
TST64: SCOPE
      MOV #5,$TIMES ;;DO 5 ITERATIONS
      MOV #3,$SLPADR ;;SET SCOPE LOOP ADDRESS
      RESET
      MOV #TST65,$ESCAPE ;;ESCAPE TO TEST 65 ON ERROR
3$: MOV DRIVE,$TACS ;SELECT DRIVE
      CLR R1 ;WASTE SOME TIME
      MOV #100.,R0
      MOVB #WFG!GO,$TACS ;START A WFG
1$: BIT #READY,$TACS ;SAMPLE READY
      BNE 2$ ;GET OUT IF READY IS SET
      INC R1
      BNE 1$
      DEC R0
      BNE 1$
      BIT #READY,$TACS ;IS "READY" BIT SET?
      BNE 2$ ;BR IF YES
      ERROR 1 ;ERROR--"READY" BIT DIDN'T SET
2$: BIT #LEADER,$TACS ;CHECK FOR CLEAR LEADER
      BEQ TST65 ;;BR IF CLEAR LEADER EQ 0
      ERROR 1 ;ON CLEAR LEADER
      ;AFTER DOING A WRITE FILE GAP
: *****

```

```

2054 ;*TEST 65 TEST "READY" CLEARS WHEN STARTING A "REWIND" FUNCTION
2055 ;*****
2056 TST65: SCOPE
2057 MOV #5,$TIMES ;;DO 5 ITERATIONS
2058 RESET
2059 MOV #1,R1 ;SET TIMER
2060 MOV DRIVE,$TACS ;SELECT DRIVE
2061 WAITREADY ;WAIT FOR "READY"
2062 MOVB #REWIND!GO,$TACS ;START A "REWIND"
2063 1$: BIT #READY,$TACS ;CHECK "READY" BIT
2064 BEQ TST66 ;;BR IF "READY" = 0
2065 ASL R1 ;IS TIME UP?
2066 BNE 1$ ;BR IF NO
2067 ERROR 1 ;"READY" FAILED TO CLEAR
2068 ;*****
2069 ;*TEST 66 TEST "READY" CLEARS WHEN STARTING A "WRITE FILE GAP" FUNCTION
2070 ;*****
2071 TST66: SCOPE
2072 MOV #5,$TIMES ;;DO 5 ITERATIONS
2073 RESET
2074 MOV #1,R1 ;SET TIMER
2075 MOV DRIVE,$TACS ;SELECT DRIVE
2076 WAITREADY ;WAIT FOR "READY"
2077 MOVB #WFG!GO,$TACS ;START A "WRITE FILE GAP"
2078 1$: BIT #READY,$TACS ;CHECK "READY" BIT
2079 BEQ TST67 ;;BR IF "READY" = 0
2080 ASL R1 ;IS TIME UP?
2081 BNE 1$ ;BR IF NO
2082 ERROR 1 ;"READY" FAILED TO CLEAR
2083 ;*****
2084 ;*TEST 67 TEST THAT "RESET" CANNOT ABORT A "REWIND"
2085 ;*****
2086 TST67: SCOPE
2087 MOV #1,$TIMES ;;DO 1 ITERATION
2088 MOV #5,$$,$ESCAPE ;;ESCAPE TO 5$ ON ERROR
2089 RESET
2090 MOV DRIVE,$TACS ;SELECT DRIVE
2091 WAITREADY ;GO WAIT FOR "READY" TO SET
2092 MOVB #REWIND!GO,$TACS ;LOAD REWIND AND GO
2093 CLR 3$
2094 2$: BIT #READY,$TACS ;WAIT FOR READY TO CLEAR
2095 BEQ 4$
2096 INCB (PC)+
2097 3$: 0
2098 BNE 2$
2099 ERROR 1 ;"READY" FAILED TO CLEAR
2100 4$: RESET ;TRY TO STOP REWIND
2101 BIS DRIVE,$TACS ;SELECT DRIVE
2102 BIT #READY,$TACS ;IS READY SET?
2103 BEQ 5$ ;BR IF NO
2104 ERROR 1 ;"READY" WAS SET
2105 5$: WAITREADY ;WAIT ON READY TO SET
2106 ;*****
2107 ;THIS TEST WILL START A "WRITE FILE GAP" AND WAIT FOR "READY" TO CLEAR
2108 ;AS SOON AS "READY" CLEARS A "RESET" IS PERFORMED (WHICH SHOULD
2109 ;CLEAR THE FUNCTION AND SET READY) THEN "READY" IS EXAMINED

```

K04

```

2110 ;TO INSURE THAT IT IS SET
2111 ;*****
2112 ;*TEST 70 TEST "RESET" WILL ABORT A "WRITE FILE GAP"
2113 ;*****
2114 TST70: SCOPE
2115 006236 000004 MOV #5,$TIMES ;;DO 5 ITERATIONS
2116 006240 012767 000005 172720 MOV #1,$SLPADR ;;SET SCOPE LOOP ADDRESS
2117 006246 012767 006264 172632 MOV #TST71,$ESCAPE ;;ESCAPE TO TEST 71 ON ERROR
2118 006254 012767 006332 172706 RESET
2119 006264 010314 1$: MOV DRIVE,@TACS ;SELECT DRIVE
2120 006266 104412 WAITREADY ;WAIT ON READY
2121 006270 112714 000001 MOVB #WFG!GO,@TACS ;START A WRITE FILE GAP FUNCTION
2122 006274 005067 000010 CLR 3$
2123 006300 032714 000040 2$: BIT #READY,@TACS ;WAIT FOR READY TO CLEAR
2124 006304 001404 BEQ 4$ ;BR IF "READY" = 0
2125 006306 105227 INCB (PC)+ ;WASTE SOME TIME
2126 006310 000000 3$: 0
2127 006312 001372 BNE 2$
2128 006314 104001 ERROR 1 ;"READY" FAILED TO CLEAR
2129 006316 000005 4$: RESET ;KILL WRITE FILE GAP
2130 006320 010314 MOV DRIVE,@TACS
2131 006322 032714 000040 BIT #READY,@TACS
2132 006326 001001 BNE TST71 ;;BR IF "RESET" SET "READY"
2133 006330 104001 ERROR 1 ;"READY" DIDN'T SET
2134 ;*****
2135 ;*TEST 71 TEST "WFG" AND "REWIND" FOR NO ERRORS
2136 ;*****
2137 TST71: SCOPE
2138 006334 012767 000005 172624 MOV #5,$TIMES ;;DO 5 ITERATIONS
2139 006342 012767 006362 172536 MOV #1,$SLPADR ;;SET SCOPE LOOP ADDRESS
2140 006350 012767 006424 172612 MOV #TST72,$ESCAPE ;;ESCAPE TO TEST 72 ON ERROR
2141 006356 000005 RESET
2142 006360 010314 MOV DRIVE,@TACS
2143 006362 112714 000017 1$: MOVB #REWIND!GO,@TACS ;GO TO "CLEAR LEADER"
2144 006366 104412 WAITREADY ;GO WAIT FOR "READY" TO SET
2145 006370 005714 TST @TACS ;IS "ERROR" SET?
2146 006372 100001 BPL 2$ ;BR IF NO
2147 006374 104001 ERROR 1 ;"ERROR" = 1 AFTER "REWIND"
2148 006376 032714 020000 2$: BIT #LEADER,@TACS ;IS "CLEAR LEADER" = 1?
2149 006402 001001 BNE 3$ ;BR IF YES
2150 006404 104001 ERROR 1 ;NOT ON "CLEAR LEADER" AFTER "REWIND"
2151 006406 112714 000001 3$: MOVB #WFG!GO,@TACS ;GET OFF OF "CLEAR LEADER"
2152 006412 104412 WAITREADY ;GO WAIT FOR "READY" TO SET
2153 006414 032714 120000 BIT #ERROR!LEADER,@TACS ;SHOULD BE OFF OF "CLEAR LEADER"
2154 ;WITH NO "ERROR"
2155 006420 001401 BEQ .+4 ;BR IF "WFG" MOVED TAPE WITH NO "ERROR"
2156 006422 104001 ERROR 1 ;"WFG" FAILED
2157 ;*****
2158 ;THIS ISN'T A REAL TEST BUT A SMALL ROUTINE TO DETERMINE THE MAX.
2159 ;TIME FOR THE WAIT LOOPS (WAIT FOR "READY" AND "TRANSFER REQUEST")
2160 ;*****
2161 ;*TEST 72 ROUTINE TO DETERMINE TIME OF WAIT LOOPS
2162 ;*****
2163 TST72: SCOPE
2164 006424 000004 MOV #1,$TIMES ;;DO 1 ITERATION
2165 006426 012767 000001 172532

```

2166	006434	005737	001100	TST	@#SPASS		: IS THIS THE FIRST PASS?
2167	006440	001021		BNE	TST73	::BR IF NO	
2168	006442	000005		RESET			
2169	006444	010314		MOV	DRIVE,@TACS		: SELECT THE DRIVE
2170	006446	112714	000017	MOVB	#REWIND!GO,@TACS		: START A REWIND
2171	006452	104412		WAITREADY			: WAIT FOR READY
2172	006454	112714	000001	MOVB	#WFG!GO,@TACS		: WRITE A FILE GAP
2173	006460	104412		WAITREADY			: WAIT ON READY
2174	006462	163737	012072	SUB	@#HGHTIM,@#MAXCNT		: GET THE TIME IT TOOK
2175	006470	005237	012116	INC	@#MAXCNT		: MAKE IT BIGGER
2176	006474	006137	012116	ROL	@#MAXCNT		
2177	006500	006137	012116	ROL	@#MAXCNT		

```

2178
2179
2180
2181
2182
2183
2184
2185 006504 000004
2186 006506 012767 000005 172452
2187 006514 012767 006600 172446
2188 006522 000005
2189 006524 010314
2190 006526 112714 000017
2191 006532 104412
2192 006534 032714 020000
2193 006540 001001
2194 006542 104001
2195 006544 005214
2196 006546 005067 000010
2197 006552 032714 000040
2198 006556 001404
2199 006560 105227
2200 006562 000000
2201 006564 001372
2202 006566 104001
2203 006570 032714 020000
2204 006574 001401
2205 006576 104001
2206 006600 005067 172364
2207 006604 104412
2208
2209
2210
2211 006606 000004
2212 006610 012767 000005 172350
2213 006616 012767 006646 172262
2214 006624 012767 006722 172336
2215 006632 000005
2216 006634 010314
2217 006636 104412
2218 006640 112714 000017
2219 006644 104412
2220 006646 012700 000001
2221 006652 010314
2222 006654 112714 000001
2223 006660 032714 000040
2224 006664 001402
2225 006666 006300
2226 006670 100373
2227 006672 000005
2228 006674 010314
2229 006676 112714 000001
2230 006702 104412
2231 006704 005714
2232 006706 100401
2233 006710 104001

```

```

;////////////////////////////////////
;////////////////////////////////////
;THE FOLLOWING TESTS ARE USED TO CHECK VARIOUS OPERATIONS WHEN AT "CLEAR LEADER"
;////////////////////////////////////
;////////////////////////////////////
;*****
;*TEST 73 TEST THAT "CLEAR LEADER" GOES FALSE ON "REWIND"
;*****
†ST73: SCOPE
MOV #5,$TIMES ;;DO 5 ITERATIONS
MOV #6,$$ESCAPE ;;ESCAPE TO 6$ ON ERROR
RESET
MOV DRIVE,@TACS ;SELECT DRIVE
MOVB #REWIND!GO,@TACS ;GO TO CLEAR LEADER
WAITREADY ;GO WAIT FOR "READY" TO SET
BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
BNE 2$ ;BR IF YES
ERROR 1 ;ERROR--NOT AT CLEAR LEADER
2$: INC @TACS ;START ANOTHER REWIND
CLR 4$
3$: BIT #READY,@TACS ;WAIT ON "READY" TO CLEAR
BEQ 5$
INCB (PC)+
4$: 0
BNE 3$
ERROR 1 ;"READY" FAILED TO CLEAR
5$: BIT #LEADER,@TACS ;DID "CLEAR LEADER" GO FALSE?
BEQ 6$ ;BR IF YES
ERROR 1 ;CLEAR LEADER DIDN'T CLEAR
6$: CLR $$ESCAPE
WAITREADY
;*****
;*TEST 74 TEST FOR "CLEAR LEADER ERROR" FOR "WRITE FILE GAP FUNCTION"
;*****
†ST74: SCOPE
MOV #5,$TIMES ;;DO 5 ITERATIONS
MOV #1,$$LPADR ;;SET SCOPE LOOP ADDRESS
MOV #1ST73,$$ESCAPE ;;ESCAPE TO TEST 75 ON ERROR
RESET
MOV DRIVE,@TACS ;SELECT DRIVE
WAITREADY ;MAKE SURE TAB1 IS READY
MOVB #REWIND+GO,@TACS ;LOAD REWIND AND GO
WAITREADY ;GO WAIT FOR "READY" TO SET
1$: MOV #1,R0
MOV DRIVE,@TACS
MOVB #WFG!GO,@TACS ;START A WRITE FILE GAP FUNCTION
3$: BIT #READY,@TACS ;WAIT ON READY TO CLEAR
BEQ 4$
ASL R0
BPL 3$
4$: RESET
MOV DRIVE,@TACS ;SELECT DRIVE
MOVB #WFG!GO,@TACS ;START A WRITE FILE GAP FUNCTION
WAITREADY ;GO WAIT FOR "READY" TO SET
TST @TACS ;DID ERROR BIT SET?
BMI 2$ ;BR IF YES
ERROR 1 ;"ERROR" BIT (BIT15) FAILED TO SET

```

```

2234 006712 032714 020000      25:   BIT    #LEADER,@TACS      ;HOW ABOUT CLEAR LEADER BIT
2235 006716 001001              BNE    TST75                ;;BR IF SET
2236 006720 104001,              ERROR  1                    ;"CLEAR LEADER" NOT SET
2237                                     ;*****
2238                                     ;*TEST 75      TEST "READY" CLEARS WHEN STARTING A "WRITE" FUNCTION
2239                                     ;*****
2240 006722 000004      TST75: SCOPE
2241 006724 012767 000005 172234   MOV    #5,$TIMES          ;;DO 5 ITERATIONS
2242 006732 000005      RESET
2243 006734 012701 000001      MOV    #1,R1              ;SET TIMER
2244 006740 010314      MOV    DRIVE,@TACS        ;SELECT DRIVE
2245 006742 104412      WAITREADY                ;WAIT FOR "READY"
2246 006744 112714 000003      MOV    #WRITE!GO,@TACS    ;START A "WRITE"
2247 006750 032714 000040      1$:   BIT    #READY,@TACS    ;CHECK "READY" BIT
2248 006754 001403      BEQ    TST76              ;;BR IF "READY" = 0
2249 006756 006301      ASL   R1                  ;IS TIME UP?
2250 006760 001373      BNE   1$                 ;BR IF NO
2251 006762 104001      ERROR  1                    ;"READY" FAILED TO CLEAR
2252                                     ;****
2253                                     ;*TEST 76      TEST "READY" CLEARS WHEN STARTING A "READ" FUNCTION
2254                                     ;*****
2255 006764 000004      TST76: SCOPE
2256 006766 012767 000005 172172   MOV    #5,$TIMES          ;;DO 5 ITERATIONS
2257 006774 000005      RESET
2258 006776 012701 000001      MOV    #1,R1              ;SET TIMER
2259 007002 010314      MOV    DRIVE,@TACS        ;SELECT DRIVE
2260 007004 104412      WAITREADY                ;WAIT FOR "READY"
2261 007006 112714 000005      MOV    #READ!GO,@TACS     ;START A "READ"
2262 007012 032714 000040      1$:   BIT    #READY,@TACS    ;CHECK "READY" BIT
2263 007016 001403      BEQ    TST77              ;;BR IF "READY" = 0
2264 007020 006301      ASL   R1                  ;IS TIME UP?
2265 007022 001373      BNE   1$                 ;BR IF NO
2266 007024 104001      ERROR  1                    ;"READY" FAILED TO CLEAR
2267                                     ;*****
2268                                     ;*TEST 77      TEST "READY" CLEARS WHEN STARTING A "BACK SPACE FILE GAP" FUNCTION
2269                                     ;*****
2270 007026 000004      TST77: SCOPE
2271 007030 012767 000005 172130   MOV    #5,$TIMES          ;;DO 5 ITERATIONS
2272 007036 000005      RESET
2273 007040 012701 000001      MOV    #1,R1              ;SET TIMER
2274 007044 010314      MOV    DRIVE,@TACS        ;SELECT DRIVE
2275 007046 104412      WAITREADY                ;WAIT FOR "READY"
2276 007050 112714 000007      MOV    #BSFG!GO,@TACS     ;START A "BACK SPACE FILE GAP"
2277 007054 032714 000040      1$:   BIT    #READY,@TACS    ;CHECK "READY" BIT
2278 007060 001403      BEQ    TST100             ;;BR IF "READY" = 0
2279 007062 006301      ASL   R1                  ;IS TIME UP?
2280 007064 001373      BNE   1$                 ;BR IF NO
2281 007066 104001      ERROR  1                    ;"READY" FAILED TO CLEAR
2282                                     ;*****
2283                                     ;*TEST 100     TEST "READY" CLEARS WHEN STARTING A "BACK SPACE BLOCK GAP" FUNCTION
2284                                     ;*****
2285 007070 000004      TST100: SCOPE
2286 007072 012767 000005 172066   MOV    #5,$TIMES          ;;DO 5 ITERATIONS
2287 007100 000005      RESET
2288 007102 012701 000001      MOV    #1,R1              ;SET TIMER
2289 007106 010314      MOV    DRIVE,@TACS        ;SELECT DRIVE

```

```

2290 007110 104412          WAITREADY          ;WAIT FOR "READY"
2291 007112 112714 000011  MOVB #BSBG!GO,@TACS ;START A "BACK SPACE BLOCK GAP"
2292 007116 032714 000040  1$: BIT #READY,@TACS ;CHECK "READY" BIT
2293 007122 001403          BEQ TST101          ;;BR IF "READY" = 0
2294 007124 006301          ASL R1              ;IS TIME UP?
2295 007126 001373          BNE 1$             ;BR IF NO
2296 007130 104001          ERROR 1           ;"READY" FAILED TO CLEAR
2297
2298          ;*****
2299          ;*TEST 101 TEST "READY" CLEARS WHEN STARTING A "SPACE FORWARD FILE GAP" FUNCTION
2300          ;*****
2300 007132 000004  TST101: SCOPE
2301 007134 012767 000005 172024  MOV #5,$TIMES      ;;DO 5 ITERATIONS
2302 007142 000005          RESET
2303 007144 012701 000001  MOV #1,R1          ;SET TIMER
2304 007150 010314          MOV DRIVE,@TACS   ;SELECT DRIVE
2305 007152 104412          WAITREADY        ;WAIT FOR "READY"
2306 007154 112714 000013  MOVB #SFFG!GO,@TACS ;START A "SPACE FORWARD FILE GAP"
2307 007160 032714 000040  1$: BIT #READY,@TACS ;CHECK "READY" BIT
2308 007164 001403          BEQ TST102        ;;BR IF "READY" = 0
2309 007166 006301          ASL R1              ;IS TIME UP?
2310 007170 001373          BNE 1$             ;BR IF NO
2311 007172 104001          ERROR 1           ;"READY" FAILED TO CLEAR
2312
2313          ;*****
2314          ;*TEST 102 TEST "READY" CLEARS WHEN STARTING A "SPACE FORWARD BLOCK GAP" FUNCTION
2315          ;*****
2315 007174 000004  TST102: SCOPE
2316 007176 012767 000005 171762  MOV #5,$TIMES      ;;DO 5 ITERATIONS
2317 007204 000005          RESET
2318 007206 012701 000001  MOV #1,R1          ;SET TIMER
2319 007212 010314          MOV DRIVE,@TACS   ;SELECT DRIVE
2320 007214 104412          WAITREADY        ;WAIT FOR "READY"
2321 007216 112714 000015  MOVB #SFBG!GO,@TACS ;START A "SPACE FORWARD BLOCK GAP"
2322 007222 032714 000040  1$: BIT #READY,@TACS ;CHECK "READY" BIT
2323 007226 001403          BEQ TST103        ;;BR IF "READY" = 0
2324 007230 006301          ASL R1              ;IS TIME UP?
2325 007232 001373          BNE 1$             ;BR IF NO
2326 007234 104001          ERROR 1           ;"READY" FAILED TO CLEAR
2327
2328          ;*****
2329          ;*TEST 103 TEST FOR "CLEAR LEADER ERROR" FOR "WRITE FUNCTION"
2330          ;*****
2330 007236 000004  TST103: SCOPE
2331 007240 012767 000005 171720  MOV #5,$TIMES      ;;DO 5 ITERATIONS
2332 007246 012767 007276 171632  MOV #1$,SLPADR    ;;SET SCOPE LOOP ADDRESS
2333 007254 012767 007370 171706  MOV #TST104,$ESCAPE ;;ESCAPE TO TEST 104 ON ERROR
2334 007262 000005          RESET
2335 007264 010314          MOV DRIVE,@TACS   ;SELECT DRIVE
2336 007266 104412          WAITREADY        ;MAKE SURE TA11 IS READY
2337 007270 112714 000017  MOVB #REWIND+GO,@TACS ;LOAD REWIND AND GO
2338 007274 104412          WAITREADY        ;GO WAIT FOR "READY" TO SET
2339 007276 012707 000001  1$: MOV #1,R0
2340 007302 010314          MOV DRIVE,@TACS
2341 007304 112714 000003  MOVB #WRITE!GO,@TACS ;START A WRITE FUNCTION
2342 007310 032714 000040  3$: BIT #READY,@TACS ;WAIT ON READY TO CLEAR
2343 007314 001402          BEQ 4$
2344 007316 006300          ASL R0
2345 007320 001373          BPL 3$

```

```

2346 007322 000005          45:  RESET
2347 007324 010314          MOV    DRIVE,@TACS          ;SELECT DRIVE
2348 007326 112714 000003  MOVB   #WRITE!GO,@TACS     ;START A WRITE FUNCTION
2349 007332 012700 000001  MOV    #1,RO                ;WAIT A WHILE FOR XFER REQ.
2350 007336 105714          TSTB   @TACS                ;XFER REQ = 1?
2351 007340 100402          BMI    6$                   ;BR IF YES
2352 007342 005200          INC    RO                    ;WAITED LONG ENOUGH?
2353 007344 001374          BNE    5$                   ;BR IF NO
2354 007346 105715          TSTB   @TADB                ;CLEAR TRANSFER REQ.
2355 007350 104412          WAITREADY                   ;GO WAIT FOR "READY" TO SET
2356 007352 005714          TST    @TACS                ;DID ERROR BIT SET?
2357 007354 100401          BMI    2$                   ;BR IF YES
2358 007356 104001          ERROR 1                    ;"ERROR" BIT (BIT15) FAILED TO SET
2359 007360 032714 020000  25:  BIT    #LEADER,@TACS       ;HOW ABOUT CLEAR LEADER BIT
2360 007364 001001          BNE    TST104                ;;BR IF SET
2361 007366 104001          ERROR 1                    ;"CLEAR LEADER" NOT SET
2362
2363 ;*****
2364 ;*TEST 104 TEST FOR "CLEAR LEADER ERROR" FOR "READ FUNCTION"
2365 ;*****
2365 007370 000004          TST104: SCOPE
2366 007372 012767 000005 171566  MOV    #5,$TIMES            ;;DO 5 ITERATIONS
2367 007400 012767 007430 171500  MOV    #1,$SLPADR          ;;SET SCOPE LOOP ADDRESS
2368 007406 012767 007504 171554  MOV    #TST105,$ESCAPE    ;;ESCAPE TO TEST 105 ON ERROR
2369 007414 000005          RESET
2370 007416 010314          MOV    DRIVE,@TACS          ;SELECT DRIVE
2371 007420 104412          WAITREADY                   ;MAKE SURE TA11 IS READY
2372 007422 112714 000017  MOVB   #REWIND+GO,@TACS    ;LOAD REWIND AND GO
2373 007426 104412          WAITREADY                   ;GO WAIT FOR "READY" TO SET
2374 007430 012700 000001  15:  MOV    #1,RO
2375 007434 010314          MOV    DRIVE,@TACS
2376 007436 112714 000005  MOVB   #READ!GO,@TACS     ;START A READ FUNCTION
2377 007442 032714 000040  35:  BIT    #READY,@TACS       ;WAIT ON READY TO CLEAR
2378 007446 001402          BEQ    4$
2379 007450 006300          ASL    RO
2380 007452 100373          BPL    3$
2381 007454 000005          45:  RESET
2382 007456 010314          MOV    DRIVE,@TACS          ;SELECT DRIVE
2383 007460 112714 000005  MOVB   #READ!GO,@TACS     ;START A READ FUNCTION
2384 007464 104412          WAITREADY                   ;GO WAIT FOR "READY" TO SET
2385 007466 005714          TST    @TACS                ;DID ERROR BIT SET?
2386 007470 100401          BMI    2$                   ;BR IF YES
2387 007472 104001          ERROR 1                    ;"ERROR" BIT (BIT15) FAILED TO SET
2388 007474 032714 020000  25:  BIT    #LEADER,@TACS       ;HOW ABOUT CLEAR LEADER BIT
2389 007500 001001          BNE    TST105                ;;BR IF SET
2390 007502 104001          ERROR 1                    ;"CLEAR LEADER" NOT SET
2391
2392 ;*****
2393 ;*TEST 105 TEST FOR "CLEAR LEADER ERROR" FOR "BACK SPACE FILE GAP FUNCTION"
2394 ;*****
2394 007504 000004          TST105: SCOPE
2395 007506 012767 000005 171452  MOV    #5,$TIMES            ;;DO 5 ITERATIONS
2396 007514 012767 007544 171364  MOV    #1,$SLPADR          ;;SET SCOPE LOOP ADDRESS
2397 007522 012767 007620 171440  MOV    #TST106,$ESCAPE    ;;ESCAPE TO TEST 106 ON ERROR
2398 007530 000005          RESET
2399 007532 010314          MOV    DRIVE,@TACS          ;SELECT DRIVE
2400 007534 104412          WAITREADY                   ;MAKE SURE TA11 IS READY
2401 007536 112714 000017  MOVB   #REWIND+GO,@TACS    ;LOAD REWIND AND GO

```



```

02 007542 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
03 007544 012700 000001 1S:  MOV      #1,RO
04 007550 010314          MOV      DRIVE,@TACS
05 007552 112714 000007          MOV      #BSFG!GO,@TACS          ;START A BACK SPACE FILE GAP FUNCTION
06 007556 032714 000040 3S:  BIT      #READY,@TACS          ;WAIT ON READY TO CLEAR
07 007562 001402          BEQ      4S
08 007564 006300          ASL      RO
09 007566 100373          BPL      3S
10 007570 000005 4S:  RESET
11 007572 010314          MOV      DRIVE,@TACS          ;SELECT DRIVE
12 007574 112714 000007          MOV      #BSFG!GO,@TACS          ;START A BACK SPACE FILE GAP FUNCTION
13 007600 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
14 007602 005714          TST      @TACS          ;DID ERROR BIT SET?
15 007604 100401          BMI      2S          ;BR IF YES
16 007606 104001          ERROR    1          ;"ERROR" BIT (BIT15) FAILED TO SET
17 007610 032714 020000 2S:  BIT      #LEADER,@TACS          ;HOW ABOUT CLEAR LEADER BIT
18 007614 001001          BNE      TST106          ;;BR IF SET
19 007616 104001          ERROR    1          ;"CLEAR LEADER" NOT SET
20
21 *****
22 *TEST 106 TEST FOR "CLEAR LEADER ERROR" FOR "BACK SPACE BLOCK GAP FUNCTION"
23 *****
24 TST106: SCOPE
25 007620 000004          MOV      #5,STIMES          ;;DO 5 ITERATIONS
26 007622 012767 000005 171336          MOV      #1$, $LPADR          ;;SET SCOPE LOOP ADDRESS
27 007630 012767 007660 171250          MOV      #TST107,$ESCAPE          ;;ESCAPE TO TEST 107 ON ERROR
28 007636 012767 007734 171324          RESET
29 007644 000005          MOV      DRIVE,@TACS          ;SELECT DRIVE
30 007646 010314          WAITREADY          ;MAKE SURE TAIL IS READY
31 007650 104412          MOV      #REWIND+GO,@TACS          ;LOAD REWIND AND GO
32 007652 112714 000017          WAITREADY          ;GO WAIT FOR "READY" TO SET
33 007660 012700 000001 1S:  MOV      #1,RO
34 007664 010314          MOV      DRIVE,@TACS
35 007666 112714 000011          MOV      #BSBG!GO,@TACS          ;START A BACK SPACE BLOCK GAP FUNCTION
36 007672 032714 000040 3S:  BIT      #READY,@TACS          ;WAIT ON READY TO CLEAR
37 007676 001402          BEQ      4S
38 007700 006300          ASL      RO
39 007702 100373          BPL      3S
40 007704 000005 4S:  RESET
41 007706 010314          MOV      DRIVE,@TACS          ;SELECT DRIVE
42 007710 112714 000011          MOV      #BSBG!GO,@TACS          ;START A BACK SPACE BLOCK GAP FUNCTION
43 007714 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
44 007716 005714          TST      @TACS          ;DID ERROR BIT SET?
45 007720 100401          BMI      2S          ;BR IF YES
46 007722 104001          ERROR    1          ;"ERROR" BIT (BIT15) FAILED TO SET
47 007724 032714 020000 2S:  BIT      #LEADER,@TACS          ;HOW ABOUT CLEAR LEADER BIT
48 007730 001001          BNE      TST107          ;;BR IF SET
49 007732 104001          ERROR    1          ;"CLEAR LEADER" NOT SET
50 *****
51 *TEST 107 TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD FILE GAP FUNCTION"
52 *****
53 TST107: SCOPE
54 007734 000004          MOV      #5,STIMES          ;;DO 5 ITERATIONS
55 007736 012767 000005 171222          MOV      #1$, $LPADR          ;;SET SCOPE LOOP ADDRESS
56 007744 012767 007774 171134          MOV      #TST110,$ESCAPE          ;;ESCAPE TO TEST 110 ON ERROR
57 007752 012767 010050 171210          RESET
58 007760 000005          MOV      DRIVE,@TACS          ;SELECT DRIVE
59 007762 010314

```

E05

TAII BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTRAC.P11 T107

MACY11 27(732) 25-SEP-76 11:04 PAGE 57

TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD FILE GAP FUNCTION"

```

2458 007764 104412          WAITREADY          ;MAKE SURE TAI1 IS READY
2459 007766 112714 000017  MOVB #REWIND+GO,@TACS ;LOAD REWIND AND GO
2460 007772 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
2461 007774 012700 000001  15:  MOV #1,R0
2462 010000 010314          MOV DRIVE,@TACS
2463 010002 112714 000013  MOVB #SFFG!GO,@TACS ;START A SPACE FWD FILE GAP FUNCTION
2464 010006 032714 000040  35:  BIT #READY,@TACS ;WAIT ON READY TO CLEAR
2465 010012 001402          BEQ 45
2466 010014 006300          ASL R0
2467 010016 100373          BPL 35
2468 010020 000005          45:  RESET
2469 010022 010314          MOV DRIVE,@TACS ;SELECT DRIVE
2470 010024 112714 000013  MOVB #SFFG!GO,@TACS ;START A SPACE FWD FILE GAP FUNCTION
2471 010030 104412          WAITREADY          ;GO WAIT FOR "READY" TO SET
2472 010032 005714          TST @TACS ;DID ERROR BIT SET?
2473 010034 100401          BMI 25 ;BR IF YES
2474 010036 104001          ERROR 1 ;"ERROR" BIT (BIT15) FAILED TO SET
2475 010040 032714 020000  25:  BIT #LEADER,@TACS ;HOW ABOUT CLEAR LEADER BIT
2476 010044 001001          BNE TST110 ;;BR IF SET
2477 010046 104001          ERROR 1 ;"CLEAR LEADER" NOT SET
*****
;TEST 110 TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD BLOCK GAP FUNCTION"
*****
TST110: SCOPE
2481 010050 000004          MOV #5,$TIMES ;;DO 5 ITERATIONS
2482 010052 012767 000005 171106  MOV #1,$SLPADR ;;SET SCOPE LOOP ADDRESS
2483 010060 012767 010110 171020  MOV #TST111,$ESCAPE ;;ESCAPE TO TEST 111 ON ERROR
2484 010066 012767 010164 171074  RESET
2485 010074 000005          MOV DRIVE,@TACS ;SELECT DRIVE
2486 010076 010314          WAITREADY          ;MAKE SURE TAI1 IS READY
2487 010100 104412          MOVB #REWIND+GO,@TACS ;LOAD REWIND AND GO
2488 010102 112714 000017  WAITREADY          ;GO WAIT FOR "READY" TO SET
2489 010106 104412          15:  MOV #1,R0
2490 010110 012700 000001  MOV DRIVE,@TACS
2491 010114 010314          MOVB #SFBG!GO,@TACS ;START A SPACE FWD BLOCK GAP FUNCTION
2492 010116 112714 000015  35:  BIT #READY,@TACS ;WAIT ON READY TO CLEAR
2493 010122 032714 000040  BEQ 45
2494 010126 001402          ASL R0
2495 010130 006300          BPL 35
2496 010132 100373          45:  RESET
2497 010134 000005          MOV DRIVE,@TACS ;SELECT DRIVE
2498 010136 010314          MOVB #SFBG!GO,@TACS ;START A SPACE FWD BLOCK GAP FUNCTION
2499 010140 112714 000015  WAITREADY          ;GO WAIT FOR "READY" TO SET
2500 010144 104412          TST @TACS ;DID ERROR BIT SET?
2501 010146 005714          BMI 25 ;BR IF YES
2502 010150 104001          ERROR 1 ;"ERROR" BIT (BIT15) FAILED TO SET
2503 010152 104001          BIT #LEADER,@TACS ;HOW ABOUT CLEAR LEADER BIT
2504 010154 032714 020000  25:  BNE TST111 ;;BR IF SET
2505 010160 001001          ERROR 1 ;"CLEAR LEADER" NOT SET
*****
;TEST 111 TEST "GO" BIT IS WRITE ONLY
*****
TST111: SCOPE
2510 010164 000004          MOV #5,$TIMES ;;DO 5 ITERATIONS
2511 010166 012767 000005 170772  MOV #1,$SLPADR ;;SET SCOPE LOOP ADDRESS
2512 010174 012767 010206 170704  RESET
2513 010202 000005

```

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTRAC.P11 T111 TEST "GO" BIT IS WRITE ONLY

2514	010204	010314			MUV	DRIVE, @TACS		; LOAD THE DRIVE
2515	010206	052714	000001	1\$:	BIS	#1, @TACS		; SET "GO" BIT
2516	010212	032714	000001		BIT	#1, @TACS		; TRY TO READ IT
2517	010216	001401			BEQ	2\$		; NO ERROR IF "GO" BIT=0
2518	010220	104001			ERROR	1		; GO BIT WASN'T 0
2519	010222	104412		2\$:	WAITREADY			; WAIT ON "READY"

```

2520 ////////////////////////////////////////////////////////////////////
2521 ////////////////////////////////////////////////////////////////////
2522 ////////////////////////////////////////////////////////////////////
2523 // THE FOLLOWING TESTS CHECKS FOR PROPER OPERATION OF "TRANSFER REQUEST"
2524 // AND "ILBS" FOR BOTH WRITING AND READING
2525 ////////////////////////////////////////////////////////////////////
2526 //*****
2527 // *TEST 112 TEST TRANSFER REQUEST & ILBS FOR "WRITE"
2528 //*****
2529 // *ST112: SCOPE
2530 010224 000004          MOV      #10, $TIMES      ;; DO 10. ITERATIONS
2531 010226 012767 000012 170732  MOV      #8$, $LPADR      ;; SET SCOPE LOOP ADDRESS
2532 010234 012767 010276 170644  MOV      #TST113, $ESCAPE ;; ESCAPE TO TEST 113 ON ERROR
2533 010242 012767 010374 170720  RESET
2534 010250 000005          MOV      DRIVE, @TACS      ; SELECT DRIVE
2535 010252 010314          WAITREADY      ; INSURE DRIVE IS READY
2536 010254 104412          MOV      #REWIND+GO, @TACS ; GO TO BOT
2537 010256 112714 000017  WAITREADY      ; GO WAIT FOR "READY" TO SET
2538 010258 104412          MOV      #WFG+GO, @TACS    ; GET OFF OF CLEAR LEADER
2539 010262 104412          WAITREADY      ; GO WAIT FOR "READY" TO SET
2540 010264 112714 000001
2541 010270 104412          RESET
2542 010272 000005          MOV      DRIVE, @TACS
2543 010274 010314          MOV      #WRITE, @TACS    ; LOAD "WRITE" FUNCTION
2544 010276 112714 000002 8$:  MOV      #1, RO
2545 010278 012700 000001 1$:  MOV      #1, RO
2546 010302 012700 000001  ASL      RO              ; KILL A LITTLE TIME
2547 010306 006300 1$:  BNE      1$
2548 010310 001376          TSTB     @TACS
2549 010312 105714          BPL      2$
2550 010314 100001          ERROR    1              ; INSURE XFER REQ=0
2551 010316 104001          INC      @TACS          ; XFER REQ. SET BEFORE "GO"
2552 010320 005214 2$:  MOV      #1, RO          ; SET "GO"
2553 010322 012700 000001 3$:  ASL      RO
2554 010324 006300          BNE      3$
2555 010326 001376          TSTB     @TACS
2556 010330 001376          BMI      4$
2557 010332 105714          ERROR    1              ; CHECK THAT XFER EQ=1
2558 010334 100401          CLRB    @TADB          ; XFER REQ FAILED TO SET
2559 010336 104001          TSTB     @TACS          ; KNOCK DOWN XFER REQ.
2560 010340 105015 4$:  BPL      5$
2561 010342 105714          ERROR    1
2562 010344 100001          WAITXFER      ; CHECK THAT "XFER REQ" IS CLEAR
2563 010346 104001          BIS      #ILBS, @TACS   ; XFER REQ FAILED TO CLEAR
2564 010350 104413 5$:  TSTB     @TACS          ; GO WAIT ON "TRANSFER REQUEST" TO SET
2565 010352 052714 000020  BPL      6$
2566 010354 105714          ERROR    1              ; SET ILBS--WRITE CRC
2567 010356 105714          TSTB     @TACS          ; DID "XFER REQ" CLEAR?
2568 010360 100001          ERROR    1              ; BR IF YES
2569 010362 104001          WAITREADY      ; "ILBS" DIDN'T CLEAR "XFER REQ"
2570 010364 104412          TST      @TACS          ; GO WAIT FOR "READY" TO SET
2571 010366 005714 6$:  BPL      TST113        ; ANY ERROR?
2572 010370 100001          ERROR    1              ; BR IF NO
2573 010372 104001          ERROR    1              ; AN ERROR OCCURRED
2574 //*****
2575 // *TEST 113 TEST "TRANSFER REQ" AND "ILBS" FOR "READ"
2576 //*****
2577 // *ST113: SCOPE
2578 010374 000004          MOV      #10, $TIMES      ;; DO 10. ITERATIONS
2579 010376 012767 000012 170562  MOV      #4$, $LPADR      ;; SET SCOPE LOOP ADDRESS
2580 010404 012767 010472 170474  MOV      #TST114, $ESCAPE ;; ESCAPE TO TEST 114 ON ERROR
2581 010412 012767 010534 170550

```

```

2576 010420 000005          RESET
2577 010422 010314          MOV      DRIVE,@TACS          ;SELECT DRIVE
2578 010424 104412          WAITREADY                    ;CHECK READY
2579 010426 052714 000017  BIS      #REWIND+GO,@TACS
2580 010432 104412          WAITREADY                    ;GO WAIT FOR "READY" TO SET
2581 010434 112714 000001  MOVB    #WFG!GO,@TACS        ;GET OFF OF "CLEAR LEADER"
2582 010440 104412          WAITREADY                    ;GO WAIT ON "READY"
2583 010442 112714 000003  MOVB    #WRITE!GO,@TACS     ;START A WRITE
2584 010446 104413          WAITXFER
2585 010450 112715 000377  MOVB    #377,@TADB          ;WRITE ON TAPE
2586 010454 104413          WAITXFER
2587 010456 112715 000377  MOVB    #377,@TADB          ;WRITE ON TAPE
2588 010462 104413          WAITXFER
2589 010464 052714 000020  BIS      #ILBS,@TACS
2590 010470 104412          WAITREADY
2591 010472 112714 000017  4S:    MOVB    #REWIND!GO,@TACS
2592 010476 104412          WAITREADY
2593 010500 112714 000005  MOVB    #READ!GO,@TACS     ;START A READ
2594 010504 104413          WAITXFER                    ;GO WAIT ON "TRANSFER REQUEST" TO SET
2595 010506 005715          TST     @TADB                ;CLEAR XFER REQ
2596 010510 105714          TSTB   @TACS                 ;DID XFER REQ CLEAR?
2597 010512 100001          BPL     2S                    ;BR IF YES
2598 010514 104001          ERROR  1                      ;XFER REQ FAILED TO CLEAR
2599 010516 104413 2S:    WAITXFER                    ;GO WAIT ON "TRANSFER REQUEST" TO SET
2600 010520 052714 000020  BIS      #ILBS,@TACS        ;START ILBS--CLEARS XFER REQ
2601 010524 105714          TSTB   @TACS                 ;DID XFER REQ CLEAR?
2602 010526 100001          BPL     3S                    ;BR IF YES
2603 010530 104001          ERROR  1                      ;XFER REQ FAILED TO CLEAR
2604 010532 104412 3S:    WAITREADY                    ;GO WAIT FOR "READY" TO SET
2605          ;*****
2606          ;*TEST 114      TEST THAT TADB (READ BUFFER) INITIALIZES TO 0'S
2607          ;*****
2608 010534 000004          †ST114: SCOPE
2609 010536 012767 010546 170342  MOV     #15,$LPADR          ;;SET SCOPE LOOP ADDRESS
2610 010544 000005          RESET
2611 010546 011500 1S:    MOV     @TADB,RO            ;PUT TADB IN RO
2612 010550 001401          BEQ    TST115                ;;CHECK FOR ALL ZEROS
2613 010552 104006          ERROR  6                      ;TADB NOT ALL ZEROS
2614          ;*****
2615          ;*TEST 115      TEST TADB (WRITE BUFFER) FOR WRITE ONLY
2616          ;*****
2617 010554 000004          †ST115: SCOPE
2618 010556 012715 177777  MOV     #-1,@TADB           ;LOAD TADB WITH 1'S
2619 010562 011500          MOV     @TADB,RO            ;PUT TADB IN RO
2620 010564 001401          BEQ    TST116                ;;BR IF IT DIDN'T CHANGE
2621 010566 104006          ERROR  6                      ;ERROR

```

```

2622
2623
2624
2625 010570 000004
2626 010572 012767 000012 170366
2627 010600 012767 010634 170300
2628 010606 012767 010666 170354
2629 010614 000005
2630 010616 010314
2631 010620 112714 000017
2632 010624 104412
2633 010626 112714 000001
2634 010632 104412
2635 010634 012700 000001
2636 010640 112714 000003
2637 010644 104413
2638 010646 110015
2639 010650 104413
2640 010652 011501
2641 010654 120001
2642 010656 001401
2643 010660 104007
2644 010662 106300
2645 010664 103370
2646 010666 152714 000020
2647 010672 104412

;*****
;TEST 116      FLOAT A "1" THROUGH TADB
;*****
↑ST116: SCOPE
MOV      #10,$TIMES      ;;DO 10. ITERATIONS
MOV      #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
MOV      #4$,$ESCAPE     ;;ESCAPE TO 4$ ON ERROR
RESET
MOV      DRIVE,@TACS
MOVB     #REWIND!GO,@TACS ;POSITION THE TAPE
WAITREADY ;GO WAIT FOR "READY" TO SET
MOVB     #WFG!GO,@TACS   ;GET OFF OF "CLEAR LEADER"
WAITREADY ;GO WAIT FOR "READY" TO SET
1$: MOV   #1,RO           ;INITIALIZE THE PATTERN
MOVB     #WRITE!GO,@TACS ;START A WRITE FUNCTION
WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
2$: MOVB  RO,@TADB       ;LOAD TADB WRITE BUFFER
WAITXFER ;WAIT ON "TRANSFER REQUEST"
MOV      @TADB,R1        ;READ THE DATA BUFFER
CMPB     RO,R1           ;CHECK THE DATA
BEQ      3$              ;BR IF DATA IS GOOD
ERROR    7               ;ERROR---TADB NOT EQUAL RO
3$: ASLB  RO              ;NEXT PATTERN
BCC      2$              ;BR IF MORE TO DO
4$: BISH  #ILBS,@TACS    ;WRITE CRC
WAITREADY ;GO WAIT FOR "READY" TO SET
    
```

```

2648
2649
2650
2651 010674 000004
2652 010676 012767 0C0012 170262
2653 010704 012767 010740 170174
2654 010712 012767 010772 170250
2655 010720 000005
2656 010722 010314
2657 010724 112714 000017
2658 010730 104412
2659 010732 112714 000001
2660 010736 104412
2661 010740 012700 177776 1$:
2662 010744 112714 000003
2663 010750 104413
2664 010752 110015 2$:
2665 010754 104413
2666 010756 011501
2667 010760 120001
2668 010762 001401
2669 010764 104007
2670 010766 106300 3$:
2671 010770 103770
2672 010772 152714 000020 4$:
2673 010776 104412

```

```

*****
*TEST 117    FLOAT A "0" THROUGH TABD
*****
TST117: SCOPE
MOV    #10,$TIMES    ;;DO 10. ITERATIONS
MOV    #1$,$LPADR    ;;SET SCOPE LOOP ADDRESS
MOV    #4$,$ESCAPE   ;;ESCAPE TO 4$ ON ERROR
RESET
MOV    DRIVE,@TACS
MOVB   #REWIND!GO,@TACS    ;POSITION THE TAPE
WAITREADY    ;GO WAIT FOR "READY" TO SET
MOVB   #WFG!GO,@TACS    ;GET OFF OF "CLEAR LEADER"
WAITREADY    ;GO WAIT FOR "READY" TO SET
MOV    #-2,RO    ;INITIALIZE THE PATTERN
MOVB   #WRITE!GO,@TACS    ;START A WRITE FUNCTION
WAITXFER    ;GO WAIT ON "TRANSFER REQUEST" TO SET
MOV    RO,@TABD    ;LOAD INTO WRITE BUFFER
WAITXFER    ;WAIT ON "TRANSFER REQUEST"
MOV    @TABD,R1    ;READ THE DATA BUFFER
CMPB   RO,R1    ;CHECK THE DATA
BEQ    3$    ;BR IF DATA IS GOOD
ERROR  7    ;ERROR---TABD NOT EQUAL RO
ASLB   RO    ;NEXT PATTERN
BCS    2$    ;BR IF MORE TO DO
BISB   #ILBS,@TACS    ;WRITE CRC
WAITREADY    ;GO WAIT FOR "READY" TO SET

```

K05

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAA.P11 T120 COUNT PATTERN TO TADB

MACY11 27(732) 25-SEP-76 11:04 PAGE 63

2674					;;*****	
2675					;;*TEST 120	COUNT PATTERN TO TADB
2676					;;*****	
2677	G11000	000004			†ST120: SCOPE	
2678	011002	012767	000012	170156	MOV #10, \$TIMES	;;DO 10. ITERATIONS
2679	011010	012767	011044	170070	MOV #1\$, \$LPADR	;;SET SCOPE LOOP ADDRESS
2680	011016	012767	011074	170144	MOV #4\$, \$ESCAPE	;;ESCAPE TO 4\$ ON ERROR
2681	011024	000005			RESET	
2682	011026	010314			MOV DRIVE, @TACS	
2683	011030	112714	000017		MOVB #REWIND!GO, @TACS	; POSITION THE TAPE
2684	011034	104412			WAITREADY	; GO WAIT FOR "READY" TO SET
2685	011036	112714	000001		MOVB #WFG!GO, @TACS	; GET OFF OF "CLEAR LEADER"
2686	011042	104412			WAITREADY	; GO WAIT FOR "READY" TO SET
2687	011044	005000			CLR RO	; INITIALIZE THE COUNT PATTERN
2688	011046	112714	000003		MOVB #WRITE!GO, @TACS	; START A WRITE FUNCTION
2689	011052	104413			WAITXFER	; GO WAIT ON "TRANSFER REQUEST" TO SET
2690	011054	110015			MOVB RO, @TADB	; LOAD TADB WRITE BUFFER
2691	011056	104413			WAITXFER	; WAIT ON "TRANSFER REQUEST"
2692	011060	011501			MOV @TADB, R1	; READ THE DATA BUFFER
2693	011062	120001			CMPB RO, R1	; CHECK THE DATA
2694	011064	001401			BEQ 3\$	; BR IF DATA IS GOOD
2695	011066	104005			ERROR 5	; ERROR---TADB NOT EQUAL RO
2696	011070	105200			3\$: INCB RO	; UPDATE THE PATTERN
2697	011072	001370			BNE 2\$	; LOOP IF NON-ZERO
2698	011074	152714	000020		4\$: BISB #ILBS, @TACS	; WRITE CRC
2699	011100	104412			WAITREADY	; GO WAIT FOR "READY" TO SET



TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAAC.P11 T121 END OF TEST CODE

2700					
2701					
2702					
2703	011102	000004			
2704	011104	012767	000001	170054	
2705	011112	000005			
2706	011114	010314			
2707	011116	112714	000017		
2708	011122	104412			
2709					
2710					
2711					
2712					
2713					
2714					
2715					
2716					
2717					
2718					
2719	011124				
2720	011124	000004			
2721	011126	005067	167750		
2722	011132	005067	170030		
2723	011136	005267	167736		
2724	011142	042767	100000	167730	
2725	011150	005327			
2726	011152	000001			
2727	011154	003015			
2728	011156	012737			
2729	011160	000001			
2730	011162	011152			
2731	011164	104401	011217		
2732	011170	013700	000042		
2733	011174	001405			
2734	011176	000005			
2735	011200	004710			
2736	011202	000240			
2737	011204	000240			
2738	011206	000240			
2739	011210				
2740	011210	000137			
2741	011212	002234			
2742	011214	377	377	000	
2743	011217	015	042412	042116	
2744	011224	050040	051501	000123	

```

*****
*TEST 121      END OF TEST CODE
*****
TST121: SCOPE
MOV      #1,$TIMES      ;;DO 1 ITERATION
RESET
MOV      DRIVE,@TACS      ;SELECT DRIVE
MOVB     #REWIND!GO,@TACS
WAITREADY
.SBTTL  END OF PASS ROUTINE

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

SEOP:
SCOPE
CLR      $STNM      ;;ZERO THE TEST NUMBER
CLR      $TIMES     ;;ZERO THE NUMBER OF ITERATIONS
INC      $PASS      ;;INCREMENT THE PASS NUMBER
BIC      #10000,$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC      (PC)+      ;;LOOP?

SEOPCT: .WORD      1
BGT      $DOAGN     ;;YES
MOV      (PC)+,@(PC)+ ;;RESTORE COUNTER

SENDCT: .WORD      1
TYPE     $SENDMG    ;;TYPE "END PASS"
MOV      @#42,R0    ;;GET MONITOR ADDRESS
BEQ      $DOAGN     ;;BRANCH IF NO MONITOR
RESET    ;;CLEAR THE WORLD
JSR     PC,(R0)     ;;GO TO MONITOR
NOP      ;;SAVE ROOM
NOP      ;;FOR
NOP      ;;ACT11

SDOAGN: JMP      @(PC)+      ;;RETURN
SRTNAD: .WORD      START
$ENULL: .BYTE      -1,-1,0  ;;NULL CHARACTER STRING
SENDMG: .ASCIZ    '<15><12>/END PASS/'

```

```

2745 .SBTTL SCOPE HANDLER ROUTINE
2746
2747 ;:*****
2748 ;:THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
2749 ;:AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
2750 ;:AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
2751 ;:THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2752 ;:*SW14=1 LOOP ON TEST
2753 ;:*SW11=1 INHIBIT ITERATIONS
2754 ;:*SW09=1 LOOP ON ERROR
2755 ;:*SW08=1 LOOP ON TEST IN SWR<7:0>
2756 ;:CALL
2757 ;:* SCOPE ;:SCOPE=IOT
2758
2759 $SCOPE:
2760 011232 104406 040000 167676 1$: CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
2761 011234 032777 040000 167676 1$: BIT #BIT14,$SWR ;:LOOP ON PRESENT TEST?
2762 011242 001111 ;:BNE $OVER ;:YES IF SW14=1
2763 ;:*****START OF CODE FOR THE XOR TESTER*****
2764 011244 000416 $XTSTR: BR 6$ ;:IF RUNNING ON THE "XOR" TESTER CHANGE
2765 ;:THIS INSTRUCTION TO A "NOP" (NOP=240)
2766 011246 013746 000004 MOV $#ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
2767 011252 012737 011272 000004 MOV #5,$#ERRVEC ;:SET FOR TIMEOUT
2768 011260 005737 177060 TST $#177060 ;:TIME OUT ON XOR?
2769 011264 012637 000004 MOV (SP)+,$#ERRVEC ;:RESTORE THE ERROR VECTOR
2770 011270 000463 BR $SVLAD ;:GO TO THE NEXT TEST
2771 011272 022626 5$: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
2772 011274 012637 000004 MOV (SP)+,$#ERRVEC ;:RESTORE THE ERROR VECTOR
2773 011300 000423 BR 7$ ;:LOOP ON THE PRESENT TEST
2774 011302 6$;*****END OF CODE FOR THE XOR TESTER*****
2775 011302 032777 000400 167630 BIT #BIT08,$SWR ;:LOOP ON SPEC. TEST?
2776 011310 001404 BEQ 2$ ;:BR IF NO
2777 011312 127767 167622 167562 CMPB $SWR,$STNM ;:ON THE RIGHT TEST? SWR<7:0>
2778 011320 001462 BEQ $OVER ;:BR IF YES
2779 011322 105767 167555 2$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
2780 011326 001421 BEQ 3$ ;:BR IF NO
2781 011330 126767 167561 167545 CMPB $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
2782 011336 101015 BHI 3$ ;:BR IF NO
2783 011340 032777 001000 167572 BIT #BIT09,$SWR ;:LOOP ON ERROR?
2784 011346 001404 BEQ 4$ ;:BR IF NO
2785 011350 016767 167534 167530 7$: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
2786 011356 000443 BR $OVER
2787 011360 105067 167517 4$: CLRB $ERFLG ;:ZERO THE ERROR FLAG
2788 011364 005067 167576 CLR $TIMES ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE
2789 011370 000415 BR 1$ ;:ESCAPE TO THE NEXT TEST
2790 011372 032777 004000 167540 3$: BIT #BIT11,$SWR ;:INHIBIT ITERATIONS?
2791 011400 001011 BNE 1$ ;:BR IF YES
2792 011402 005767 167472 TST $PASS ;:IF FIRST PASS OF PROGRAM
2793 011406 001406 BEQ 1$ ;: INHIBIT ITERATIONS
2794 011410 005267 167470 INC $ICNT ;:INCREMENT ITERATION COUNT
2795 011414 026767 167546 167462 CMP $TIMES,$ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE
2796 011422 002021 BGE $OVER ;:BR IF MORE ITERATION REQUIRED
2797 011424 012767 000001 167452 1$: MOV #1,$ICNT ;:REINITIALIZE THE ITERATION COUNTER
2798 011432 016767 000044 167526 MOV $MXCNT,$TIMES ;:SET NUMBER OF ITERATIONS TO DO
2799 011440 105267 167436 $SVLAD: INCB $STNM ;:COUNT TEST NUMBERS
2800 011444 011667 167436 MOV (SP),$LPADR ;:SAVE SCOPE LOOP ADDRESS

```

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAAC.P11 SCOPE HANDLER ROUTINE

2801	011450	011667	167434		MOV	(SP), \$LPERR	::SAVE ERROR LOOP ADDRESS
2802	011454	005067	167510		CLR	\$ESCAPE	::CLEAR THE ESCAPE FROM ERROR ADDRESS
2803	011460	112767	000001	167427	MOVB	#1, \$ERMAX	::ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2804	011466	016777	167410	167446	\$OVER: MOV	\$TSTNM, @DISPLAY	::DISPLAY TEST NUMBER
2805	011474	016716	167406		MOV	\$LPADR, (SP)	::FUDGE RETURN ADDRESS
2806	011500	000002			RTI		::FIXES PS
2807	011502	003720			\$MXCNT: 2000.		::MAX. NUMBER OF ITERATIONS

```

2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822 011504
2823 011504 104406
2824 011506 011437 001162
2825 011512 011537 001164
2826 011516 010037 001124
2827 011522 010137 001126
2828 011526 105267 167351
2829 011532 001775
2830 011534 016777 167342 167400
2831 011542 032777 002000 167370
2832 011550 001402
2833 011552 104401 001172
2834 011556 005267 167330
2835 011562 011667 167330
2836 011566 162767 000002 167322
2837 011574 117767 167316 167312
2838 011602 032777 020000 167330
2839 011610 001004
2840 011612 004767 000060
2841 011616 104401 001177
2842 011622
2843 011622 005777 167312
2844 011626 100002
2845 011630 000000
2846 011632 104406
2847 011634 032777 001000 167276
2848 011642 001402
2849 011644 016716 167240
2850 011650 005767 167314
2851 011654 001402
2852 011656 016716 167306
2853 011662
2854 011662 022737 011200 000042
2855 011670 001001
2856 011672 000000
2857 011674
2858 011674 000002
2859

```

.SBTTL ERROR HANDLER ROUTINE

```

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

```

```

$ERROR: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
MOV @TACS,@$SREG0 ;;SAVE THE STATUS REG.
MOV @TADB,@$SREG1 ;;SAVE THE DATA BUFFER
ERROR1: MOV RO,@$GDDAT ;;RO WILL CONTAIN THE GOOD DATA
MOV RI,@$BDDAT ;;RI WILL CONTAIN THE BAD DATA
7$: INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,$SWR ;;BELL ON ERROR?
BEQ 1$ ;;NO - SKIP
TYPE $BELL ;;RING BELL
1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,$ERRPC
MOVB @$ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,$SWR ;;SKIP TYPEOUT IF SET
BNE 20$ ;;SKIP TYPEOUTS
JSR PC,TYPERR ;;GO TO USER ERROR ROUTINE
TYPE ,SCLF
20$:
25: TST $SWR ;;HALT ON ERROR
BPL 3$ ;;SKIP IF CONTINUE
HALT ;;HALT ON ERROR!
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
3$: BIT #BIT09,$SWR ;;LCOP ON ERROR SWITCH SET?
BEQ 4$ ;;BR IF NO
MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
BEQ 5$ ;;BR IF NONE
MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
5$:
6$: CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
BNE 6$ ;;BRANCH IF NO
HALT ;;YES
RTI ;;RETURN

```

```

2860                                     ;*****
2861                                     ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
2862
2863 011676 104401 001177 TYPERR: TYPE      SCRLF          ;TYPE A CARRIAGE RETURN & LINE FEED
2864 011702 010046      MOV      RO,-(SP)      ;SAVE RO
2865 011704 113700 001114      MOV      2(ITEMB,RO) ;PICKUP THE ITEM INDEX
2866 011710 005300      DEC      RO          ;ADJUST THE INDEX
2867 011712 006300      ASL      RO          ;SO IT WILL WORK FOR
2868 011714 006300      ASL      RO          ;THE ERROR TABLE
2869 011716 006300      ASL      RO
2870 011720 062700 001236      ADD      2(ERTB,RO) ;FORM THE TABLE POINTER
2871 011724 012067 000002      MOV      (RO)+,1$  ;PICKUP "ERROR MESSAGE" POINTER
2872 011730 104401      TYPE     ;TYPE "ERROR MESSAGE"
2873 011732 000000      1$:    0          ;"ERROR MESSAGE POINTER" GOES HERE
2874 011734 104401 001177      TYPE     SCRLF          ;
2875 011740 012067 000004      MOV      (RO)+,2$  ;PICKUP "DATA HEADER" POINTER
2876 011744 001404      BEQ     3$          ;IF "0" DON'T TYPE
2877 011746 104401      TYPE     ;TYPE "DATA HEADER"
2878 011750 000000      2$:    0          ;"DATA HEADER" POINTER GOES HERE
2879 011752 104401 001177      TYPE     SCRLF          ;
2880 011756 012000 3$:    MOV      (RO)+,RO  ;PICKUP "DATA POINTER"
2881 011760 001004      BNE     5$          ;IF THERE IS DATA TO TYPE GO DO IT
2882 011762 012600 4$:    MOV      (SP)+,RO  ;RESTORE RO
2883 011764 104401 001177      TYPE     SCRLF          ;TYPE A CARRAGE RETURN&LINE FEED
2884 011770 000207      RTS      PC          ;RETURN
2885 011772 5$:
2886 011772 013046      MOV      2(RO)+,-(SP) ;:SAVE 2(RO)+ FOR TYPEOUT
2887
2888 011774 104402      TYPOC   ;:TYPE DATA
2889 011776 005710      TST     (RO)        ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2890 012000 001770      BEQ     4$          ;:TERMINATOR?
2891 012002 104401 012010      TYPE     6$          ;:BR IF YES
2892 012006 000771      BR      5$          ;:TYPE 2 SPACES
2893 012010 020040 000      6$:    .ASCIZ  / /      ;:LOOP
2894 012014 .EVEN      .EVEN          ;:ASCII STRING OF 2 SPACES

```

```

2895                                     ;*****
2896                                     ;ROUTINE TO WAIT ON THE READY BIT
2897
2898 012014                               WAIT.ON.READY:
2899 012014 005067 G00044                 CLR     WAIT2           ;SETUP MAX. TIME TO WAIT ON "READY"
2900 012020 016767 000072 000044       MOV     MAXCNT,HGHTIM
2901 012026 012637 001202               MOV     (SP)+, @#SAVPC  ;GET THE PC OF THE WAITREADY INSTRUCTION
2902 012032 162737 000002 001202       SUB     #2, @#SAVPC
2903 012040 012637 001204               MOV     (SP)+, @#SAVPS  ;SAVE THE PS
2904 012044 032714 000040             WAIT1: BIT     #READY, @TACS ;READY=1?
2905 012050 001013                       BNE     WAIT3           ;GO ON IF YES
2906 012052 105714                       TSTB   @TACS           ;CHECK TRANSFER REQUEST
2907 012054 100002                       BPL     WAIT4
2908 012056 104004                       ERROR  4               ;"TRANSFER REQUEST" SET WHILE WAITING ON "READY"
2909 012060 000407                       BR      WAIT3
2910 012062 005227             WAIT4: INC     (PC)+       ;COUNT FAST COUNTER
2911 012064 000000             WAIT2: 0
2912 012066 001366                       BNE     WAIT1           ;GO CHECK "READY" AGAIN
2913 012070 005327                       DEC     (PC)+       ;COUNT LOOP COUNTER
2914 012072 000000             HGHTIM: 0
2915 012074 003363                       BGT     WAIT1           ;GO LOOP AGAIN
2916 012076 104002                       ERROR  2               ;"READY" FAILED TO SET
2917 012100 013746 001204             WAIT3: MOV     @#SAVPS, -(SP) ;GET THE STATUS BACK
2918 012104 013746 001202             MOV     @#SAVPC, -(SP) ;GET THE PC
2919 012110 062716 000002             ADD     #2, (SP)
2920 012114 000002             RTI
2921 012116 000000             MAXCNT: 0
2922
2923                                     ;*****
2924                                     ;ROUTINE TO WAIT ON TRANSFER REQUEST
2925
2926 012120                               WAIT.FOR.XFER.REQ:
2927 012120 005067 000044                 CLR     2$           ;SETUP WASTE TIME LOOP
2928 012124 013767 012116 000044       MOV     @#MAXCNT, 3$
2929 012132 012637 001202               MOV     (SP)+, @#SAVPC  ;GET THE PC OF THE WAITXFER INSTRUCTION
2930 012136 162737 000002 001202       SUB     #2, @#SAVPC
2931 012144 012637 001204               MOV     (SP)+, @#SAVPS  ;SAVE THE PS
2932 012150 105714             1$: TSTB   @TACS           ;CHECK XFER REQ
2933 012152 100414                       BMI     4$
2934 012154 032714 000040             BIT     #READY, @TACS ;LOOK AT READY
2935 012160 001402                       BEQ     5$           ;BR IF "READY" ISN'T SET
2936 012162 104004                       ERROR  4               ;"READY" SET WHILE WAITING FOR "XFER REQ"
2937 012164 000407                       BR      4$
2938 012166 005227             5$: INC     (PC)+       ;COUNT
2939 012170 000000             2$: 0
2940 012172 001366                       BNE     1$
2941 012174 005327                       DEC     (PC)+       ;BR IF MORE TO DO
2942 012176 000000             3$: 0
2943 012200 003363                       BGT     1$
2944 012202 104003                       ERROR  3               ;"TRANSFER REQUEST" FAILED TO SET
2945 012204 013746 001204             4$: MOV     @#SAVPS, -(SP) ;GET THE STATUS BACK
2946 012210 013746 001202             MOV     @#SAVPC, -(SP) ;GET THE PC
2947 012214 062716 000002             ADD     #2, (SP)
2948 012220 000002             RTI

```

```

2949 ;*****
2950 .SBTTL ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2951
2952 ;CALL
2953 JSR PC, @ASKDRV
2954 RETURN ;NOTE: R0 AND R1 ARE DESTROYED
2955
2956 ASKDRV: TYPE MSGDRV ;<CRLF>"DRIVE(S)"
2957 CLR DRVKEY ;GO GET A DRIVE
2958 RDLIN ;SETUP TO CHECK FOR VALID DRIVE(S)
2959 MOV (SP)+, R0 ;WAS A DRIVE SELECTED?
2960 TSTB @R0 ;BR IF NO
2961 BEQ NOTLGL
2962 MOV #DRVKEY, R1
2963 CMPB #'A, @R0 ;WAS DRIVE "A" SELECTED?
2964 BNE NOTA ;BR IF NO
2965 MOVB (R0)+, (R1)+ ;SET KEY FOR DRIVE "A"
2966 BR NEXT
2967 CMPB #'B, @R0 ;WAS DRIVE "B" SELECTED?
2968 BNE NOTB ;BR IF NO
2969 MOVB (R0)+, (R1)+ ;SET KEY FOR DRIVE "B"
2970 BR NEXT
2971 CMPB #54, @R0 ;WAS A COMMA TYPED?
2972 BNE NOTLGL ;BR IF NO
2973 TSTB (R0)+ ;DUMP THE COMMA
2974 @R0 ;TERMINATOR?
2975 BEQ EXIT ;BR IF YES
2976 CMP #DRVKEY+2, R1 ;TWO DRIVES SELECTED?
2977 BHI LOOP ;BR IF NO
2978 TYPE , $QUES ;ILLEGAL INPUT DETECTED
2979 BR ASKDRV ;GO TRY AGAIN
2980 TST DRVKEY ;ANY DRIVE SELECTED?
2981 BEQ NOTLGL ;BR IF NO
2982 RTS PC
2983
2984

```



```

2985 ;:*****
2986 ;.SBTTL ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY
2987 ;CALL
2988 ;JSR PC,@ASKADR
2989
2990 012332 010046 ASKADR: MOV RO,-(SP) ;SAVE RO
2991 012334 104401 015603 1$: TYPE ,MSGASK ;"TACS?"
2992 012340 104411 RDOCT ;GET VALUE
2993 012342 012600 MOV (SP)+,RO ;PICK UP THE OCTAL NUMBER
2994 012344 001423 BEQ 3$ ;IF "0" USE OLD VALUES
2995 012346 020027 160000 CMP RO,#160000 ;MAKE SURE IT IS A BUS ADDRESS
2996 012352 103770 BLO 1$
2997 012354 010037 001206 MOV RO,@TACSL ;SAVE TOE TACS
2998 012360 062700 000002 ADD #2,RO ;STEP TO TADB ADDRESS
2999 012364 010037 001212 MOV RO,@TADBL ;AND SAVE IT
3000 012370 013737 001206 001210 MOV @TACSL,@TACSH ;SETUP TACS UPPER
3001 012376 005237 001210 INC @TACSH ;BYTE POINTER
3002
3003 012402 013737 001212 001214 MOV @TADBL,@TADBH ;SETUP TADB UPPER
3004 012410 005237 001214 INC @TADBH ;BYTE POINTER
3005 012414 104401 015613 3$: TYPE ,MSGVEC ;"VECTOR?"
3006 012420 104411 RDOCT
3007 012422 012600 MOV (SP)+,RO
3008 012424 001411 BEQ 5$
3009 012426 020027 001000 CMP RO,#1000 ;MAKE SURE ADDRESS IS IN VECTOR AREA
3010 012432 103370 BHIS 3$
3011 012434 010037 001216 MOV RO,@TAVEC ;SAVE AS VECTOR ADDRESS
3012 012440 062700 000002 ADD #2,RO
3013 012444 010037 001220 MOV RO,@TAVEC+2
3014 012450 104401 015623 5$: TYPE ,MSGPRI ;ASK FOR PRIORITY
3015 012454 104411 RDOCT
3016 012456 012600 MOV (SP)+,RO
3017 012460 001413 BEQ 6$ ;IF "0" USE OLD VALUE
3018 012462 020027 000007 CMP RO,#7 ;MAKE SURE ITS VALID
3019 012466 101370 BHI 5$
3020 012470 000300 SWAB RO ;PUT INTO HIGH BYTE
3021 012472 006200 ASR RO ;AND SHIFT
3022 012474 006200 ASR RO ;INTO PROPER
3023 012476 006200 ASR RO ;POSITION
3024 012500 042700 177437 BIC #C(340),RO ;SAVE ONLY PRIORITY BITS
3025 012504 010037 001222 MOV RO,@TAPRIO ;STORE IT AWAY
3026 012510 104401 015635 6$: TYPE ,MTACS ;TACS="
3027 012514 016746 166466 MOV TACSL,-(SP) ;;SAVE TACSL FOR TYPEOUT
3028 012520 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3029 012522 104401 015643 TYPE ,MTADB ;"TADB="
3030 012526 016746 166460 MOV TADBL,-(SP) ;;SAVE TADBL FOR TYPEOUT
3031 012532 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3032 012534 104401 015652 TYPE ,MTAVEC ;"VECTOR="
3033 012540 016746 166452 MOV TAVEC,-(SP) ;;SAVE TAVEC FOR TYPEOUT
3034 012544 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3035 012546 104401 015663 TYPE ,MTAPRI ;"PRIORITY="
3036 012552 016746 166444 MOV TAPRIO,-(SP) ;;SAVE TAPRIO FOR TYPEOUT
3037 012556 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3038 012560 104401 015676 TYPE ,MSGOK ;"OK?"
3039 012564 104407 RDCHR ;GO READ ONE CHARACTER
3040 012566 012600 MOV (SP)+,RO ;GET IT

```



3041	012570	022700	000015		CMP	#15,RO	: IS IT "CR"?
3042	012574	001406			BEQ	7\$	: BRANCH IF YES
3043	012576	022700	000131		CMP	#'Y,RO	: IS IT "Y"?
3044	012602	001403			BEQ	7\$	: IT WAS
3045	012604	104401	0C1176		TYPE	\$QUES	: TYPE "?"
3046	012610	000651			BR	1\$	: AND LET HIM CORRECT THEM
3047	012612	104401	015704	7\$:	TYPE	MYES	: TYPE OUT "YES"
3048	012616	012600			MOV	(SP)+,RO	: RESTORE RO
3049	012620	000207			RTS	PC	: AND RETURN

3050  
3051  
3052  
3053  
3054  
3055  
3056  
3057  
3058  
3059  
3060  
3061  
3062  
3063  
3064  
3065  
3066  
3067  
3068  
3069  
3070  
3071  
3072  
3073  
3074  
3075  
3076  
3077  
3078  
3079  
3080  
3081  
3082  
3083  
3084  
3085  
3086  
3087  
3088  
3089  
3090  
3091  
3092  
3093  
3094  
3095  
3096  
3097  
3098  
3099  
3100  
3101  
3102  
3103  
3104  
3105

012622 012706 001100  
012626 013704 001206  
012632 013705 001212  
012636 000005  
012640 012737 012622 001110  
012646 004737 013626  
012652 010314  
012654 112714 000017  
012660 032714 000040  
012664 001775  
012666 112714 000001  
012672 104412  
012674 032714 020000  
012700 001772  
012702 000000  
012704 000746

001110

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

\*\*\*\*\*  
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, @SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
        JSR PC, @NXTDRV ;GO SETUP FOR NEXT DRIVE
        MOV DRIVE, @TACS ;SELECT DRIVE
        MOVB #REWIND!GO, @TACS ;SEND TAPE TO "BOT"
100$: BIT #READY, @TACS ;WAIT ON READY
        BEQ 100$
1$: MOVB #WFG!GO, @TACS ;WRITE A FILE GAP
        WAITREADY ;WAIT ON READY
        BIT #LEADER, @TACS ;AT "CLEAR LEADER"?
        BEQ 1$ ;BR IF NO
        HALT ;STOP IF YES
        BR WFGSUB ;LOOP ON CONT.
```

\*\*\*\*\*  
WRITE CONTINUOUS BLOCKS OF DATA  
START AT 224  
THE PROGRAM WILL HALT THREE(3) TIMES  
AFTER EACH HALT SET THE SWR AND PRESS CONTINUE  
\*\*\* IF USING SOFTWARE SWITCH REGISTER  
PRESS CONTINUE AND  
PROGRAM WILL TYPE "SWR=XXXXXX NEW="  
AND ALLOW A NEW VALUE TO BE INPUT  
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"  
\*\*\*\*\*

012706 004737 013370  
012712 012706 001100  
012716 013704 001206  
012722 013705 001212  
012726 000005  
012730 012737 012712 001110  
012736 004737 013626  
012742 010314  
012744 112714 000017  
012750 032714 000040

001110

```
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, @SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
        JSR PC, @NXTDRV ;GO SETUP FOR NEXT DRIVE
        MOV DRIVE, @TACS ;SELECT DRIVE
        MOVB #REWIND!GO, @TACS ;SEND TAPE TO "BOT"
100$: BIT #READY, @TACS ;WAIT ON READY
```

```

3106 012754 001775          BEQ      100$
3107 012756 004737 013504  1$:      JSR      PC, @#WRTBLK      ;WRITE A BLOCK
3108 012762 032714 020000          BIT      @#LEADER, @TACS      ;AT "CLEAR LEADER"?
3109 012766 001773          BEQ      1$      ;BR IF NO
3110 012770 000000          HALT
3111 012772 000747          BR       WLOOP      ;STOP IF "EOT"
                                   ;LOOP IF CONT.
3112
3113
3114      ;*****
3115      ; READ CONTINUOUS BLOCKS OF DATA
3116      ; START AT 230
3117      ; THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
3118      ; AND THE "THRESHOLD POT".
3119      ;*****
3120 012774 012706 001100  RDSUB:  MOV      @STACK, SP      ;KEEP THE STACK OUT OF THE WAY
3121 013000 013704 001206          MOV      @#TACSL, TACS      ;SETUP THE TA11 STATUS AND
3122 013004 013705 001212          MOV      @#TADBL, TADB      ;DATA BUFFER REGISTERS
3123 013010 000005          RESET      ;RESET THE WORLD
3124 013012 012737 012774 001110  MOV      @#RDSUB, @#SLPERR      ;SETUP THE LOOP ON ERROR ADDRESS
3125 013020 004737 013626          JSR      PC, @#NXTDRV      ;GO SETUP FOR NEXT DRIVE
3126 013024 010314          MOV      DRIVE, @TACS      ;SELECT DRIVE
3127 013026 112714 000017          MOV      @#REWIND!GO, @TACS      ;SEND TAPE TO "BOT"
3128 013032 032714 000040  100$:  BIT      @#READY, @TACS      ;WAIT ON READY
3129 013036 001775          BEQ      100$
3130 013040 004737 013546  1$:      JSR      PC, @#RDBLK      ;READ A BLOCK
3131 013044 032714 020000          BIT      @#LEADER, @TACS      ;AT "CLEAR LEADER"?
3132 013050 001351          BNE      RDSUB      ;BR IF YES---LOOP
3133 013052 000772          BR       1$
3134
3135
3136      ;*****
3137      ; WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO FAT
3138      ; START AT 234
3139      ; THE PROGRAM WILL HALT THREE(3) TIMES
3140      ; AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
3141      ; **** IF USING SOFTWARE SWITCH REGISTER
3142      ; PRESS CONTINUE AND
3143      ; PROGRAM WILL TYPE "SWR=XXXXXX NEW="
3144      ; AND ALLOW A NEW VALUE TO BE INPUT
3145      ; HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
3146      ; HALT 2 --- SWR<7:0> = PATTERN DESIRED
3147      ; HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
3148      ; THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
3149      ; AND THE "GAP TIME MONO".
3150      ;*****
3151 013054 004737 013370  WGPBLK: JSR      PC, @#SETBUF      ;GET BLOCK SIZE AND PATTERN
3152 013060 012706 001100  WGBLOP: MOV      @STACK, SP      ;KEEP THE STACK OUT OF THE WAY
3153 013064 013704 001206          MOV      @#TACSL, TACS      ;SETUP THE TA11 STATUS AND
3154 013070 013705 001212          MOV      @#TADBL, TADB      ;DATA BUFFER REGISTERS
3155 013074 000005          RESET      ;RESET THE WORLD
3156 013076 012737 013060 001110  MOV      @#WGBLOP, @#SLPERR      ;SETUP THE LOOP ON ERROR ADDRESS
3157 013104 004737 013626          JSR      PC, @#NXTDRV      ;GO SETUP FOR NEXT DRIVE
3158 013110 010314          MOV      DRIVE, @TACS      ;SELECT DRIVE
3159 013112 112714 000017          MOV      @#REWIND!GO, @TACS      ;SEND TAPE TO "BOT"
3160 013116 032714 000040  100$:  BIT      @#READY, @TACS      ;WAIT ON READY
3161 013122 001775          BEQ      100$

```

```

3162 013124 112714 000001 1$: MOVB #WFG!GO,@TACS ;WRITE A FILE GAP
3163 013130 104412 WAITREADY ;WAIT ON READY
3164 013132 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
3165 013136 001005 BNE 2$ ;BR IF YES
3166 013140 004737 013504 JSR PC,@#WRTBLK ;WRITE A BLOCK
3167 013144 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
3168 013150 001765 BEQ 1$ ;BR IF NO
3169 013152 000000 2$: HALT ;STOP AT "EOT"
3170 013154 000741 BR WGBLOP ;START OVER ON CONT.
3171
3172
3173 ;*****
3174 ; READ A BLOCK OF DATA AND A FILE GAP
3175 ; START AT 240
3176 ; THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
3177 ; IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
3178 ; AND THE "TAPE BLANK MONO".
3179 ;*****
3180 013156 012706 001100 RGPBLK: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
3181 013162 013704 001206 MOV @#TACSL,TACS ;SETUP THE TA11 STATUS AND
3182 013166 013705 001212 MOV @#TADBL,TADB ;DATA BUFFER REGISTERS
3183 013172 000005 RESET ;RESET THE WORLD
3184 013174 012737 013156 001110 MOV #RGPBLK,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3185 013202 004737 013626 JSR PC,@#NXTDRV ;GO SETUP FOR NEXT DRIVE
3186 013206 010314 MOV DRIVE,@TACS ;SELECT DRIVE
3187 013210 112714 000017 MOVB #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
3188 013214 032714 000040 100$: BIT #READY,@TACS ;WAIT ON READY
3189 013220 001775 BEQ 100$
3190 013222 004737 013546 1$: JSR PC,@#RDBLK ;READ A BLOCK OF DATA
3191 013226 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
3192 013232 001351 BNE RGPBLK ;BR IF YES
3193 013234 112714 000015 MOVB #SFBG!GO,@TACS ;GET INTO A FILE GAP
3194 013240 104412 WAITREADY
3195 013242 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
3196 013246 001343 BNE RGPBLK ;BR IF YES
3197 013250 000764 BR 1$ ;LOOP
3198
3199
3200 ;*****
3201 ; SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
3202 ; START AT 244
3203 ; THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
3204 ; SPACE FOWARD (TAPE BLANK MONO CAN BE ADJUSTED), OR AFTER READ OR
3205 ; WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
3206 ; (SIGNAL MONO CAN BE CHECKED).
3207 ;*****
3208 013252 012706 001100 SFFGSB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
3209 013256 013704 001206 MOV @#TACSL,TACS ;SETUP THE TA11 STATUS AND
3210 013262 013705 001212 MOV @#TADBL,TADB ;DATA BUFFER REGISTERS
3211 013266 000005 RESET ;RESET THE WORLD
3212 013270 012737 013252 001110 MOV #SFFGSB,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
3213 013276 004737 013626 JSR PC,@#NXTDRV ;GO SETUP FOR NEXT DRIVE
3214 013302 010314 MOV DRIVE,@TACS ;SELECT DRIVE
3215 013304 112714 000017 MOVB #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
3216 013310 032714 000040 100$: BIT #READY,@TACS ;WAIT ON READY
3217 013314 001775 BEQ 100$

```

# K06

```
3218 013315 112714 000013 1$: MOVB #SFFG!GO,@TACS ;SPACE INTO A FILE GAP
3219 013322 104412 WAITREADY ;WAIT ON READY
3220 013324 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
3221 013330 001772 BEQ 1$ ;BR IF NO
3222 013332 000000 HALT ;STOP AT "EOT"
3223 013334 000746 BR SFFGSB ;LOOP ON CONT.
3224
3225
3226 ;*****
3227 ; BACK SPACE FILE GAP
3228 ;START AT 250
3229 ;THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
3230 ;*****
3231 013336 000005 BSFGSB: RESET ;RESET THE WORLD
3232 013340 012737 013336 001110 MOV #BSFGSB,@$SLPERR ;LOOP ON ERROR ADDRESS
3233 013346 010314 MOV DRIVE,@TACS ;SELECT DRIVE
3234 013350 112714 000007 1$: MOVB #BSFG!GO,@TACS ;BACK SPACE A FILE GAP
3235 013354 104412 WAITREADY ;WAIT ON READY
3236 013356 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
3237 013362 001772 BEQ 1$ ;BR IF NO
3238 013364 000000 HALT ;STOP AT BOT
3239 013366 000763 BR BSFGSB ;START OVER ON CONT.
3240
3241
3242 ;*****
3243 ; SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
3244 ;*****
3245 013370 005000 SETBUF: CLR RO
3246 013372 000000 HALT ;OPERATOR PUTS BYTE COUNT IN SWR<7:0>
3247 013374 022767 000176 165536 CMP #SWREG,SWR ;SOFTWARE SWITCH REG.?
3248 013402 001001 BNE 20$ ;NO-LET HIM BE
3249 013404 104405 GTSWR ;GET NEW VALUE
3250 013406 20$: ;CONTINUE
3251 013406 157700 165526 BISB @SWR,RO ;PICKUP THE BYTE COUNT
3252 013412 001006 BNE 2$ ;BR IF NON-ZERO
3253 013414 105777 165521 TSTB @SWR+1 ;CHECK IF GREATER THAN 377
3254 013420 001402 BEQ 1$ ;BR IF NO
3255 013422 012700 000376 MOV #376,RO ;SET FOR MAX ALLOWED
3256 013426 005200 1$: INC RO ;MAKE IT 377 OR 1
3257 013430 010037 013500 2$: MOV RO,@#BLKLIM ;SETUP THE BLOCK LIMIT
3258 013434 005037 013502 CLR @#PATRN
3259 013440 000000 HALT ;OPERATOR PUTS PATTERN IN SWR<7:0>
3260 013442 022767 000176 165470 CMP #SWREG,SWR ;SOFTWARE SWITCH REG.?
3261 013450 001001 BNE 21$ ;NO-LET HIM BE
3262 013452 104405 GTSWR ;GET NEW VALUE
3263 013454 21$: ;CONTINUE
3264 013454 117737 165460 013502 MOVB @SWR,@#PATRN ;PICK UP THE PATTERN
3265 013462 000000 HALT ;SET OPERATIONAL SWITCHES
3266 013464 022767 000176 165446 CMP #SWREG,SWR ;SOFTWARE SWITCH REG.?
3267 013472 001001 BNE 22$ ;NO-LET HIM BE
3268 013474 104405 GTSWR ;GET NEW VALUE
3269 013476 22$: ;CONTINUE
3270 013476 000207 RTS PC ;RETURN
3271 013500 000000 BLKLIM: 0 ;READ AND WRITE BLOCK SIZE
3272 013502 000000 PATRN: 0 ;PATTERN TO GO ON THE TAPE
3273
```

```

3274
3275
3276
3277
3278 013504 013701 013500
3279 013510 112714 000003
3280 013514 104413
3281 013516 032714 000040
3282 013522 001010
3283 013524 005301
3284 013526 002403
3285 013530 113715 013502
3286 013534 000767
3287 013536 052714 000020
3288 013542 104412
3289 013544 000207
3290
3291
3292
3293
3294
3295 013546 013702 013500
3296 013552 013700 013502
3297 013556 112714 000005
3298 013562 104413
3299 013564 032714 000040
3300 013570 001012
3301 013572 005302
3302 013574 002405
3303 013576 011501
3304 013600 120001
3305 013602 001767
3306 013604 104007
3307 013606 000406
3308 013610 052714 000020
3309 013614 104412
3310 013616 005714
3311 013620 100001
3312 013622 104001
3313 013624 000207
3314
3315
3316
3317
3318 013626 105777 165306
3319 013632 100416
3320 013634 005003
3321 013636 013701 001230
3322 013642 122127 000101
3323 013646 001402
3324 013650 012703 000400
3325 013654 105711
3326 013656 001002
3327 013660 012701 001224
3328 013664 010137 001230
3329 013670 000207

```

```

;*****
; WRITE ROUTINE FOR THE MANUAL OPERATIONS
;*****
WRTBLK: MOV     @#BLKLIM,R1      ; PICKUP THE BLOCK SIZE
        MOVB   #WRITE!GO,@TACS ; START A WRITE
1$:     WAITXFER                ; WAIT ON TRANSFER REQUEST
        BIT    #READY,@TACS    ; DID READY SET?
        BNE   3$               ; BR IF YES
        DEC   R1               ; COUNT THIS REQUEST
        BLT   2$               ; BR IF TIME FOR ILBS
        MOVB   @#PATRN,@TADB   ; PUT DATA ON TAPE
        BR    1$               ; LOOP
2$:     BIS    #ILBS,@TACS     ; WRITE CRC AND SHUT DOWN
        WAITREADY              ; WAIT ON THE READY FLAG
3$:     RTS    PC

```

```

;*****
; READ ROUTINE FOR THE MANUAL OPERATIONS
;*****
RDBLK:  MOV     @#BLKLIM,R2     ; PICKUP THE BLOCK SIZE
        MOV     @#PATRN,R0     ; USE THIS DATA PATTERN TO COMPARE TO
        MOVB   #READ!GO,@TACS ; START A READ
1$:     WAITXFER                ; WAIT ON TRANSFER REQUEST
        BIT    #READY,@TACS    ; IS READY SET?
        BNE   3$               ; BR IF YES
        DEC   R2               ; COUNT THIS REQUEST
        BLT   2$               ; BR IF TIME FOR ILBS
        MOV    @TADB,R1        ; READ THE DATA BUFFER
        CMPB   R0,R1           ; CHECK THE DATA
        BEQ   1$               ; BR IF OK
        ERROR  7               ; BAD DATA
        BR    4$               ; GET OUT
2$:     BIS    #ILBS,@TACS     ; READ ILBS
        WAITREADY              ; WAIT ON READY
3$:     TST   @TACS            ; CHECK FOR ERROR
        BPL   4$               ; BR IF NONE
        ERROR  1               ; ERROR OCCURRED
4$:     RTS    PC              ; RETURN

```

```

;*****
; ROUTINE TO CHANGE DRIVES
;*****
NXTDRV: TSTB   @SWR            ; IS SW07 ON A (1)?
        BMI   3$               ; BR IF YES
        CLR   DRIVE           ; SET DRIVE TO "A"
        MOV   @#DRVPNT,R1     ; GET DRIVE POINTER
        CMPB (R1)+,#'A        ; IS IT DRIVE "A"?
        BEQ   1$               ; BR IF YES
        MOV   #UNIT,DRIVE     ; SET DRIVE TO "B"
1$:     TSTB   (R1)            ; LAST DRIVE BEEN SELECTED
        BNE   2$               ; BR IF NO
        MOV   #DRVKEY,R1      ; RESET DRIVE POINTER
2$:     MOV   R1,@#DRVPNT     ; SAVE DRIVE POINTER FOR NEXT TIME
3$:     RTS    PC              ; GO BACK

```

```

3330 ;*****
3331 ;
3332 ;SBTTL          ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
3333 ;
3334 ;CALL:
3335 ;           MOV      #DRVKEY,RO
3336 ;           JSR      PC,@#EXAM          ;R1 IS DESTROYED
3337 ;           NORMAL RETURN
3338 ;           ERROR RETURN
3339 ;
3340 013672 013701 001206 EXAM: MOV      @#TACSL,R1          ;PICKUP THE "CONTROL & STATUS" REG. ADR.
3341 013676 005011        CLR      (R1)          ;DRIVE="A" FUNCTION="WFG"
3342 013700 122710 000101        CMPB   #'A,(R0)          ;EXAMINE DRIVE "A"?
3343 013704 001402        BEQ      1$          ;BR IF YES
3344 013706 052711 000400        BIS      #UNIT,(R1)          ;SELECT DRIVE "B"
3345 013712 032711 000040 1$: BIT      #READY,(R1)          ;WAIT ON READY
3346 013716 001775        BEQ      1$
3347 013720 005711        TST      (R1)          ;ANY ERROR?
3348 013722 100024        BPL      4$          ;BR IF NO
3349 013724 032711 001000        BIT      #OFFLINE,(R1)          ;ERROR DUE TO "OFF LINE"?
3350 013730 001017        BNE      3$          ;BR IF YES
3351 013732 032711 010000        BIT      #WRTLOCK,(R1)          ;ERROR DUE TO "WRITE LOCK"?
3352 013736 001411        BEQ      2$          ;BR IF NO
3353 013740 122777 000201 165172 CMPB   #BIT07!BIT00,@SWR          ;"READONLY" SELECTED? (RD1PAS)
3354 013746 001412        BEQ      4$          ;BR IF YES
3355 013750 122777 000203 165162 CMPB   #BIT07!BIT01!BIT00,@SWR          ;(RD2PAS)?
3356 013756 001406        BEQ      4$          ;BR IF YES
3357 013760 000403        BR       3$          ;TAKE THE ERROR EXIT
3358 013762 032711 020000 2$: BIT      #LEADER,(R1)          ;ERROR DUE TO "CLEAR LEADER"?
3359 013766 001002        BNE      4$          ;BR IF YES
3360 013770 062716 000002 3$: ADD      #2,(SP)          ;TAKE ERROR RETURN
3361 013774 000207 4$: RTS      PC          ;RETURN

```

```

3362          .SBTTL  TYPE ROUTINE
3363
3364          ;;*****
3365          ;;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3366          ;;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3367          ;;*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3368          ;;*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3369          ;;*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3370          ;;*
3371          ;;*CALL:
3372          ;;*1) USING A TRAP INSTRUCTION
3373          ;;*      TYPE      ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRJ
3374          ;;*OR
3375          ;;*      TYPE
3376          ;;*      MESADR
3377          ;;*
3378
3379 013776 105767 165155 $TYPE:  TSTB      $STPFLG          ;; IS THERE A TERMINAL?
3380 014002 100002          BPL          1$          ;; BR IF YES
3381 014004 000000          HALT          ;; HALT HERE IF NO TERMINAL
3382 014006 000407          BR          3$          ;; LEAVE
3383 014010 010046          1$:  MOV      RD, -(SP)          ;; SAVE RD
3384 014012 017600 000002  MOV      @2(SP), RD          ;; GET ADDRESS OF ASCIZ ST
3385 014016 112046          2$:  MOVVB   (RD)+, -(SP)          ;; PUSH CHARACTER TO BE
3386 014020 001005          BNE          4$          ;; BR IF IT ISN'T THE T
3387 014022 005726          TST      (SP)+          ;; IF TERMINATOR POP IT OFF THE STACK
3388 014024 012600          60$: MOV      (SP)+, ?          ;; RESTORE RD
3389 014026 062716 000002  3$:  ADD      #2, (SP)          ;; ADJUST RETURN PC
3390 014032 000002          RTI          ;; RETURN
3391 014034 122716 000011  4$:  CMPB    #HT, (SP)          ;; BRANCH IF <HT>
3392 014040 001430          BEQ          8$
3393 014042 122716 000200  CMPB    #CRLF, (SP)          ;; BRANCH IF NOT <CRLF>
3394 014046 001006          BNE          5$
3395 014050 005726          TST      (SP)+          ;; POP <CR><LF> EQUIV
3396 014052 104401          TYPE          ;; TYPE A CR AND LF
3397 014054 001177          $CRLF
3398 014056 105067 000130  CLRB    $CHARCNT          ;; CLEAR CHARACTER COUNT
3399 014062 000755          BR          2$          ;; GET NEXT CHARACTER
3400 014064 004767 000056  5$:  JSR     PC, $TYPEC          ;; GO TYPE THIS CHARACTER
3401 014070 126726 165062  6$:  CMPB    $FILLC, (SP)+          ;; IS IT TIME FOR FILLER CHARS.?
3402 014074 001350          BNE          2$          ;; IF NO GO GET NEXT CHAR.
3403 014076 016746 165052  MOV     $NULL, -(SP)          ;; GET # OF FILLER CHARS. NEEDED
3404          ;; AND THE NULL CHAR.
3405 014102 105366 000001  7$:  DECB    1(SP)          ;; DOES A NULL NEED TO BE TYPED?
3406 014106 002770          BLT          6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
3407 014110 004767 000032  JSR     PC, $TYPEC          ;; GO TYPE A NULL
3408 014114 105367 000072  DECB    $CHARCNT          ;; DO NOT COUNT AS A COUNT
3409 014120 000770          BR          7$          ;; LOOP
3410
3411          ;HORIZONTAL TAB PROCESSOR
3412
3413 014122 112716 000040  8$:  MOVVB   #' (SP)          ;; REPLACE TAB WITH SPACE
3414 014126 004767 000014  9$:  JSR     PC, $TYPEC          ;; TYPE A SPACE
3415 014132 132767 000007 000052  BITB    #7, $CHARCNT          ;; BRANCH IF NOT AT
3416 014140 001372          BNE          9$          ;; TAB STOP
3417 014142 005726          TST      (SP)+          ;; POP SPACE OFF STACK

```



```

01418 014144 000724          BH      2$          ;;GET NEXT CHARACTER
01419 014146 105777 164776 $TYPEC: TSTB  2$STPS  ;;WAIT UNTIL PRINTER IS READY
01420 014152 100375          BPL      $TYPEC
01421 014154 116677 000002 164770 MOVB    2(SP),2$STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
01422 014162 122766 000015 000002 CMPB    *CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
01423 014170 001003          BNE      1$          ;;BRANCH IF NO
01424 014172 105067 000014 CLRB    $CHARCNT    ;;YES--CLEAR CHARACTER COUNT
01425 014176 000406          BR       $TYPEX     ;;EXIT
01426 014200 122766 000012 000002 1$: CMPB    LF,2(SP)    ;;IS CHARACTER A LINE FEED?
01427 014208 001402          BEQ     $TYPEX     ;;BRANCH IF YES
01428 014210 105227          INCB    (PC)+      ;;COUNT THE CHARACTER
01429 014212 000000          $CHARCNT: .WORD  0 ;;CHARACTER COUNT STORAGE
01430 014214 000207          $TYPEX: RTS      PC

.SBTTL REAL -N OCTAL NUMBER FROM THE TTY

*****
; THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
; CHANGE IT TO BINARY.
; CALL:
; *   RDOCT          ;; READ AN OCTAL NUMBER
; *   RETURN HERE   ;; LOW ORDER BITS ARE ON TOP OF THE STACK
; *                ;; HIGH ORDER BITS ARE IN $HIOCT

01431 014216 011646          $RDOCT: MOV    (SP)-,(SP) ;; PROVIDE SPACE FOR THE
01432 014220 016566 000004 000002 MOV    4(SP),2(SP)    ;; INPUT NUMBER
01433 014226 010046          MOV    R0,-(SP)     ;; PUSH R0 ON STACK
01434 014230 010146          MOV    R1,-(SP)     ;; PUSH R1 ON STACK
01435 014232 010246          MOV    R2,-(SP)     ;; PUSH R2 ON STACK
01436 014234 104410 1$: ROLIN  ;; READ AN ASCII LINE
01437 014236 012600          MOV    (SP)+,R0     ;; GET ADDRESS OF 1ST CHARACTER
01438 014240 005001          CLR    R1           ;; CLEAR DATA WORD
01439 014242 005002          CLR    R2
01440 014244 112046 2$: MOVB    (R0)+,-(SP) ;; PICKUP THIS CHARACTER
01441 014246 001412          BEQ    3$           ;; IF ZERO GET OUT
01442 014250 006301          ASL   R1            ;; *2
01443 014252 006102          ROL   R2            ;; *4
01444 014254 006301          ASL   R1            ;; *4
01445 014256 006102          ROL   R2            ;; *8
01446 014260 006301          ASL   R1            ;; *8
01447 014262 006102          ROL   R2
01448 014264 042716 177770 BIC    *1C7,(SP)    ;; STRIP THE ASCII JUNK
01449 014270 062601          ADD   (SP)+,R1     ;; ADD IN THIS DIGIT
01450 014272 000764          BR    2$           ;; LOOP
01451 014274 005726 3$: TST    (SP)+      ;; CLEAN TERMINATOR FROM STACK
01452 014276 010166 000012 MOV    R1,12(SP)    ;; SAVE THE RESULT
01453 014302 010267 000010 MOV    R2,$HIOCT
01454 014306 012602          MOV    (SP)+,R2    ;; POP STACK INTO R2
01455 014310 012601          MOV    (SP)+,R1    ;; POP STACK INTO R1
01456 014312 012600          MOV    (SP)+,R0    ;; POP STACK INTO R0
01457 014314 000002          RTI              ;; RETURN
01458 014316 000000          $HIOCT: .WORD  0  ;; HIGH ORDER BITS GO HERE
01459 .SBTTL TTY INPUT ROUTINE

*****
; ENABL  LSB

```

3477  
3478  
3479  
3480  
3481  
3482  
3483  
3484  
3485  
3486  
3487  
3488  
3489  
3490  
3491  
3492  
3493  
3494  
3495  
3496  
3497  
3498  
3499  
3500  
3501  
3502  
3503  
3504  
3505  
3506  
3507  
3508  
3509  
3510  
3511  
3512  
3513  
3514  
3515  
3516  
3517  
3518  
3519  
3520  
3521  
3522  
3523  
3524  
3525  
3526  
3527  
3528  
3529

014320 022767 000176 164612  
014326 001074  
014330 105777 164610  
014334 100071  
014336 117746 164604  
014342 042716 177600  
014346 022726 000007  
014352 001062  
014354 126727 164554 000001  
014362 001456  
  
014364 104401 015045  
014370 104401 015052  
014374 016746 163576  
014400 104402  
014402 104401 015063  
014406 005046  
014410 005046  
014412 105777 164526  
014416 100375  
  
014420 117746 164522  
014424 042716 177600  
  
014430 021627 000025  
014434 001005  
014436 104401 015040  
014442 062706 000006  
014446 000757  
  
014450 021627 000015  
014454 001022  
014456 005766 000004  
014462 001403  
014464 016677 000002 164446  
014472 062706 000006  
014476 104401 001177  
014502 126727 164427 000001  
014510 001003  
014512 012777 000100 164424  
014520 000002  
014522 004767 177420  
014526 021627 000060  
014532 002420  
014534 021627 000067  
014540 003015  
014542 042726 000060

```
*****
*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
*WHEN OPERATING IN TTY FLAG MODE.
$CKSWR: CMP      #SWREG,SWR      ;; IS THE SOFT-SWR SELECTED?
        BNE      15$           ;; BRANCH IF NO
        TSTB     @STKS          ;; CHAR THERE?
        BPL      15$           ;; IF NO, DON'T WAIT AROUND
        MOVB     @STKB,-(SP)     ;; SAVE THE CHAR
        BIC      #177,(SP)      ;; STRIP-OFF THE ASCII
        CMP      #7,(SP)+       ;; IS IT A CONTROL G?
        BNE      15$           ;; NO, RETURN TO USER
        CMPB     $AUTOB,#1      ;; ARE WE RUNNING IN AUTO-MODE?
        BEQ      15$           ;; BRANCH IF YES

$GTSWR: TYPE     ,SCNTLG        ;; ECHO THE CONTROL-G (↑G)
        TYPE     $MSWR          ;; TYPE CURRENT CONTENTS
        MOV      SWREG,-(SP)    ;; SAVE SWREG FOR TYPEOUT
        TYPOC    ,SMNEW        ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
        TYPE     -(SP)         ;; PROMPT FOR NEW SWR
19$:    CLR      -(SP)          ;; CLEAR COUNTER
        CLR      -(SP)         ;; THE NEW SWR
7$:    TSTB     @STKS          ;; CHAR THERE?
        BPL      7$           ;; IF NOT TRY AGAIN

        MOVB     @STKB,-(SP)    ;; PICK UP CHAR
        BIC      #177,(SP)     ;; MAKE IT 7-BIT ASCII

9$:    CMP      (SP),#25        ;; IS IT A CONTROL-U?
        BNE      10$          ;; BRANCH IF NOT
        TYPE     ,SCNTLU       ;; YES, ECHO CONTROL-U (↑U)
20$:   ADD      #5,SP          ;; IGNORE PREVIOUS INPUT
        BR       19$          ;; LET'S TRY IT AGAIN

10$:   CMP      (SP),#15        ;; IS IT A <CR>?
        BNE      16$          ;; BRANCH IF NO
        TST     4(SP)          ;; YES, IS IT THE FIRST CHAR?
        BEQ     11$          ;; BRANCH IF YES
        MOV     2(SP),@SWR     ;; SAVE NEW SWR
11$:   ADD      #6,SP          ;; CLEAR UP STACK
14$:   TYPE     ,SCRLF         ;; ECHO <CR> AND <LF>
        CMPB    $INTAG,#1     ;; RE-ENABLE TTY KBD INTERRUPTS?
        BNE     15$          ;; BRANCH IF NOT
        MOV     #100,@STKS    ;; RE-ENABLE TTY KBD INTERRUPTS
15$:   RTI                    ;; RETURN
16$:   JSR     PC,$TYPEC      ;; ECHO CHAR
        CMP     (SP),#60      ;; CHAR < 0?
        BLT     18$          ;; BRANCH IF YES
        CMP     (SP),#67      ;; CHAR > 7?
        BGT     18$          ;; BRANCH IF YES
        BIC     #60,(SP)+     ;; STRIP-OFF ASCII
```

```

3530 014546 005766 000002          TST      2(SP)          ;; IS THIS THE FIRST CHAR
3531 014552 001403          BEQ      17$          ;; BRANCH IF YES
3532 014554 006316          ASL      (SP)        ;; NO, SHIFT PRESENT
3533 014556 006316          ASL      (SP)        ;; CHAR OVER TO MAKE
3534 014560 006316          ASL      (SP)        ;; ROOM FOR NEW ONE.
3535 014562 005266 000002 17$: INC      2(SP)        ;; KEEP COUNT OF CHAR
3536 014566 056616 177776          BIS      -2(SP), (SP) ;; SET IN NEW CHAR
3537 014572 000707          BR       7$          ;; GET THE NEXT ONE
3538 014574 104401 001176 18$: TYPE  $QUES      ;; TYPE ?<CR><LF>
3539 014600 000720          BR       20$        ;; SIMULATE CONTROL-U
3540          .DSABL  LSB
3541
3542
3543          ;; *****
3544          ;; THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3545          ;; CALL:
3546          ;;          RDCHR          ;; INPUT A SINGLE CHARACTER FROM THE TTY
3547          ;;          RETURN HERE  ;; CHARACTER IS ON THE STACK
3548          ;;
3549          ;;
3550
3551 014602 011646          $RDCHR: MOV      (SP), -(SP) ;; PUSH DOWN THE PC
3552 014604 016666 000004 000002 MOV      4(SP), 2(SP) ;; SAVE THE PS
3553 014612 105777 164326 1$: TSTB    2$TKS      ;; WAIT FOR
3554 014616 100375          BPL      1$          ;; A CHARACTER
3555 014620 117766 164322 000004 MOVB    2$TKB, 4(SP) ;; READ THE TTY
3556 014626 042766 177600 000004 BIC     #1C<177>, 4(SP) ;; GET RID OF JUNK IF ANY
3557 014634 026627 000004 000023 CMP     4(SP), #23   ;; IS IT A CONTROL-S?
3558 014642 001013          BNE      3$          ;; BRANCH IF NO
3559 014644 105777 164274 2$: TSTB    2$TKS      ;; WAIT FOR A CHARACTER
3560 014650 100375          BPL      2$          ;; LOOP UNTIL ITS THERE
3561 014652 117746 164270 MOVB    2$TKB, -(SP) ;; GET CHARACTER
3562 014656 042716 177600 BIC     #1C<177>, (SP) ;; MAKE IT 7-BIT ASCII
3563 014662 022627 000021 CMP     (SP)+, #21   ;; IS IT A CONTROL-Q?
3564 014666 001366          BNE      2$          ;; IF NOT DISCARD IT
3565 014670 000750          BR       1$          ;; YES, RESUME
3566 014672 026627 000004 000140 3$: CMP     4(SP), #140 ;; IS IT UPPER CASE?
3567 014700 002407          BLT     4$          ;; BRANCH IF YES
3568 014702 026627 000004 000175 CMP     4(SP), #175 ;; IS IT A SPECIAL CHAR?
3569 014710 003003          BGT     4$          ;; BRANCH IF YES
3570 014712 042766 000040 000004 BIC     #40, 4(SP)   ;; MAKE IT UPPER CASE
3571 014720 000002          RTI          ;; GO BACK TO USER
3572          ;; *****
3573          ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3574          ;; CALL:
3575          ;;          RDLIN          ;; INPUT A STRING FROM THE TTY
3576          ;;          RETURN HERE  ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3577          ;;
3578          ;;
3579          ;;
3579 014722 010346          $RDLIN: MOV      R3, -(SP) ;; SAVE R3
3580 014724 012703 015030 1$: MOV     #TTYIN, R3   ;; GET ADDRESS
3581 014730 022703 015040 2$: CMP     #TTYIN+8., R3 ;; BUFFER FULL?
3582 014734 101405          BLOS    4$          ;; BR IF YES
3583 014736 104407          RDCHR   ;; GO READ ONE CHARACTER FROM THE TTY
3584 014740 112613          MOVB    (SP)+, (R3) ;; GET CHARACTER
3585 014742 122713 000177 10$: CMPB   #177, (R3) ;; IS IT A RUBOUT

```

```

3586 014746 001003          BNE      3$          ;; SKIP IF NOT
3587 014750 104401 001176 4$:      TYPE    $QUES      ;; TYPE A '?'
3588 014754 000763          BR       1$          ;; CLEAR THE BUFFER AND LOOP
3589 014756 111367 000044 3$:      MOV     (R3),9$      ;; ECHO THE CHARACTER
3590 014762 104401 015026          TYPE    9$
3591 014766 122723 000015          CMP     #15,(R3)+    ;; CHECK FOR RETURN
3592 014772 001356          BNE     2$          ;; LOOP IF NOT RETURN
3593 014774 105063 177777          CLRB   -1(R3)      ;; CLEAR RETURN (THE 15)
3594 015000 104401 001200          TYPE    $LF        ;; TYPE A LINE FEED
3595 015004 012603          MOV     (SP)+,R3    ;; RESTORE R3
3596 015006 011646          MOV     (SP),-(SP)  ;; ADJUST THE STACK AND PUT ADDRESS OF THE
3597 015010 016666 000004 000002 MOV     4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
3598 015016 012766 015030 000004 MOV     $TTYIN,4(SP)
3599 015024 000002          RTI              ;; RETURN
3600 015026 000          9$:      .BYTE   0          ;; STORAGE FOR ASCII CHAR. TO TYPE
3601 015027 000          .BYTE   0          ;; TERMINATOR
3602 015030 000010          $TTYIN: .BLKB   8.    ;; RESERVE 8 BYTES FOR TTY INPUT
3603 015040 052536 005015 000          $CNTLU: .ASCIZ  /↑U/<15><12>  ;; CONTROL "U"
3604 015045 136 006507 000012          $CNTLG: .ASCIZ  /↑G/<15><12>  ;; CONTROL "G"
3605 015052 005015 053523 020122          $MSWR:  .ASCIZ  <15><12>/SWR = /
3606 015060 020075 000          $MNEW:  .ASCIZ  / NEW = /
3607 015063 040 047040 053505
3608 015070 036440 000040          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
3609
3610
3611          ;; *****
3612          ;; *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3613          ;; *OCTAL (ASCII) NUMBER AND TYPE IT.
3614          ;; *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3615          ;; *CALL:
3616          ;; *      MOV     NUM,-(SP)          ;; NUMBER TO BE TYPED
3617          ;; *      TYPOS          ;; CALL FOR TYPEOUT
3618          ;; *      .BYTE   N          ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3619          ;; *      .BYTE   M          ;; M=1 OR 0
3620          ;; *          ;; 1=TYPE LEADING ZEROS
3621          ;; *          ;; 0=SUPPRESS LEADING ZEROS
3622
3623          ;; *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3624          ;; *$TYPOS OR $TYPOC
3625          ;; *CALL:
3626          ;; *      MOV     NUM,-(SP)          ;; NUMBER TO BE TYPED
3627          ;; *      TYPON          ;; CALL FOR TYPEOUT
3628
3629          ;; *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3630          ;; *CALL:
3631          ;; *      MOV     NUM,-(SP)          ;; NUMBER TO BE TYPED
3632          ;; *      TYPOC          ;; CALL FOR TYPEOUT
3633
3634 015074 017646 000000          $TYPOS: MOV     2(SP),-(SP)  ;; PICKUP THE MODE
3635 015100 116667 000001 000211          MOV     1(SP),$OFILL  ;; LOAD ZERO FILL SWITCH
3636 015106 112667 000207          MOV     (SP)+,$OMODE+1 ;; NUMBER OF DIGITS TO TYPE
3637 015112 062716 000002          ADD     #2,(SP)      ;; ADJUST RETURN ADDRESS
3638 015116 000406          BR     $TYPON
3639 015120 112767 000001 000171          $TYPOC: MOV     #1,$OFILL  ;; SET THE ZERO FILL SWITCH
3640 015126 112767 000006 000165          MOV     #6,$OMODE+1  ;; SET FOR SIX(6) DIGITS
3641 015134 112767 000005 000154          $TYPON: MOV     #5,$OCNT  ;; SET THE ITERATION COUNT

```

3642	015142	010346		MOV	R3,-(SP)	::SAVE R3
3643	015144	010446		MOV	R4,-(SP)	::SAVE R4
3644	015146	010546		MOV	R5,-(SP)	::SAVE R5
3645	015150	116704	000145	MOV	\$OMODE+1,R4	::GET THE NUMBER OF DIGITS TO TYPE
3646	015154	005404		NEG	R4	
3647	015156	062704	000006	ADD	#6,R4	::SUBTRACT IT FOR MAX. ALLOWED
3648	015162	110467	000132	MOV	R4,\$OMODE	::SAVE IT FOR USE
3649	015166	116704	000125	MOV	\$OFILL,R4	::GET THE ZERO FILL SWITCH
3650	015172	016605	000012	MOV	12(SP),R5	::PICKUP THE INPUT NUMBER
3651	015176	005003		CLR	R3	::CLEAR THE OUTPUT WORD
3652	015200	006105		1\$: ROL	R5	::ROTATE MSB INTO "C"
3653	015202	000404		BR	3\$	::GO DO MSB
3654	015204	006105		2\$: ROL	R5	::FORM THIS DIGIT
3655	015206	005105		ROL	R5	
3656	015210	006105		RCL	R5	
3657	015212	010503		MOV	R5,R3	
3658	015214	006103		3\$: ROL	R3	::GET LSB OF THIS DIGIT
3659	015216	105367	000076	DECB	\$OMODE	::TYPE THIS DIGIT?
3660	015222	100016		BPL	7\$	::BR IF NO
3661	015224	042703	177770	BIC	#177770,R3	::GET RID OF JUNK
3662	015230	001002		BNE	4\$	::TEST FOR 0
3663	015232	005704		TST	R4	::SUPPRESS THIS 0?
3664	015234	001403		BEQ	5\$	::BR IF YES
3665	015236	005204		4\$: INC	R4	::DON'T SUPPRESS ANYMORE 0'S
3666	015240	052703	000060	BIS	#'0,R3	::MAKE THIS DIGIT ASCII
3667	015244	052703	000040	5\$: BIS	#' R3	::MAKE ASCII IF NOT ALREADY
3668	015250	110367	000040	MOV	R3,8\$	::SAVE FOR TYPING
3669	015254	104401	015314	TYPE	8\$	::GO TYPE THIS DIGIT
3670	015260	105367	000032	7\$: DECB	\$OCNT	::COUNT BY 1
3671	015264	003347		BGT	2\$	::BR IF MORE TO DO
3672	015266	002402		BLT	6\$	::BR IF DONE
3673	015270	005204		INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
3674	015272	000744		BR	2\$	::GO DO THE LAST DIGIT
3675	015274	012605		6\$: MOV	(SP)+,R5	::RESTORE R5
3676	015276	012604		MOV	(SP)+,R4	::RESTORE R4
3677	015300	012603		MOV	(SP)+,R3	::RESTORE R3
3678	015302	016666	000002 000004	MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
3679	015310	012616		MOV	(SP)+,(SP)	
3680	015312	000002		RTI		::RETURN
3681	015314	000		8\$: .BYTE	0	::STORAGE FOR ASCII DIGIT
3682	015315	000		.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
3683	015316	000		\$OCNT: .BYTE	0	::OCTAL DIGIT COUNTER
3684	015317	000		\$OFILL: .BYTE	0	::ZERO FILL SWITCH
3685	015320	000000		\$OMODE: .WORD	0	::NUMBER OF DIGITS TO TYPE

3686  
3687  
3688  
3689  
3690  
3691  
3692  
3693  
3694 015322 010046  
3695 015324 016600 000002  
3696 015330 005740  
3697 015332 111000  
3698 015334 006300  
3699 015336 016000 015356  
3700 015342 000200

.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.

```

```

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

```

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

3701  
3702  
3703  
3704  
3705 015344 011646  
3706 015346 016666 000004 000002  
3707 015354 000002

```

$TRAP2: 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.

```

3708  
3709  
3710  
3711  
3712  
3713  
3714  
3715  
3716 015356 015344  
3717 015360 013776  
3718 015362 015120  
3719 015364 015074  
3720 015366 015134  
3721  
3722 015370 014370  
3723  
3724 015372 014320  
3725 015374 014602  
3726 015376 014722  
3727 015400 014216  
3728 015402 012014  
3729 015404 012120

	ROUTINE		
\$TRPAD:	.WORD	\$TRAP2	
	\$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.

```

3730          .SBTTL  POWER DOWN AND UP ROUTINES
3731
3732          ;:*****
3733          :POWER DOWN ROUTINE
3734 015406 012737 015552 000024 $PWRDN: MOV    $SILLUP,@#PWRVEC ;:SET FOR FAST UP
3735 015414 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;:PRIO:7
3736 015422 010046          MOV    RO,-(SP) ;:PUSH RO ON STACK
3737 015424 010146          MOV    R1,-(SP) ;:PUSH R1 ON STACK
3738 015426 010246          MOV    R2,-(SP) ;:PUSH R2 ON STACK
3739 015430 010346          MOV    R3,-(SP) ;:PUSH R3 ON STACK
3740 015432 010446          MOV    R4,-(SP) ;:PUSH R4 ON STACK
3741 015434 010546          MOV    R5,-(SP) ;:PUSH R5 ON STACK
3742 015436 017746 163476      MOV    @SWR,-(SP) ;:PUSH @SWR ON STACK
3743 015442 010667 000110      MOV    SP,$SAVR6 ;:SAVE SP
3744 015446 012737 015460 000024      MOV    $PWRUP,@#PWRVEC ;:SET UP VECTOR
3745 015454 000000          HALT
3746 015456 000776          BR      .-2 ;:HANG UP
3747
3748          ;:*****
3749          :POWER UP ROUTINE
3750 015460 012737 015552 000024 $PWRUP: MOV    $SILLUP,@#PWRVEC ;:SET FOR FAST DOWN
3751 015466 016706 000064      MOV    $SAVR6,SP ;:GET SP
3752 015472 005067 000060      CLR    $SAVR6 ;:WAIT LOOP FOR THE TTY
3753 015476 005267 000054      IS:   INC    $SAVR6 ;:WAIT FOR THE INC
3754 015502 001375          BNE    IS ;:OF WORD
3755 015504 012677 163430      MOV    (SP)+,@SWR ;:POP STACK INTO @SWR
3756 015510 012605          MOV    (SP)+,R5 ;:POP STACK INTO R5
3757 015512 012604          MOV    (SP)+,R4 ;:POP STACK INTO R4
3758 015514 012603          MOV    (SP)+,R3 ;:POP STACK INTO R3
3759 015516 012602          MOV    (SP)+,R2 ;:POP STACK INTO R2
3760 015520 012601          MOV    (SP)+,R1 ;:POP STACK INTO R1
3761 015522 012600          MOV    (SP)+,R0 ;:POP STACK INTO R0
3762 015524 012737 015406 000024      MOV    $PWRDN,@#PWRVEC ;:SET UP THE POWER DOWN VECTOR
3763 015532 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;:PRIO:7
3764 015540 104401          TYPE ;:REPORT THE POWER FAILURE
3765 015542 015560      SPWRMG: .WORD $POWER ;:POWER FAIL MESSAGE POINTER
3766 015544 012716          MOV    (PC)+,(SP) ;:RESTART AT PWRST
3767 015546 002222      SPWRAD: .WORD PWRST ;:RESTART ADDRESS
3768 015550 000002          RTI
3769 015552 000000      $SILLUP: HALT ;:THE POWER UP SEQUENCE WAS STARTED
3770 015554 000776          BR      .-2 ;:BEFORE THE POWER DOWN WAS COMPLETE
3771 015556 000000      $SAVR6: 0 ;:PUT THE SP HERE
3772 015560 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
3773 015566 000122
3774          .EVEN

```

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
 DZTRAC.P11 POWER DOWN AND UP ROUTINES

3775	015570	042200	044522	042526	MSGDRV: .ASCIZ	<CRLF>/DRIVE(S)?/
3776	015576	051450	037451	000		
3777	015603	015	052012	041501	MSGASK: .ASCIZ	<15><12>/TACS?/
3778	015610	037523	000			
3779	015613	126	041505	047524	MSGVEC: .ASCIZ	/VECTOR?/
3780	015620	037522	000			
3781	015623	120	044522	051117	MSGPRI: .ASCIZ	/PRIORITY?/
3782	015630	052111	037531	000		
3783	015635	124	041501	036523	MTACS: .ASCIZ	/TACS=/
3784	015642	000				
3785	015643	040	040524	041104	MTADB: .ASCIZ	/ TADB=/
3786	015650	000075				
3787	015652	053040	041505	047524	MTAVEC: .ASCIZ	/ VECTOR=/
3788	015660	036522	000			
3789	015663	040	051120	047511	MTAPRI: .ASCIZ	/ PRIORITY=/
3790	015670	044522	054524	000075		
3791	015676	005015	045517	000077	MSGOK: .ASCIZ	<15><12>/OK?/
3792	015704	042531	006523	000012	MYES: .ASCIZ	/YES/<15><12>
3793	015712	052123	052101	051525	EMI: .ASCIZ	/STATUS PROBLEM/
3794	015720	050040	047522	046102		
3795	015726	046505	000			
3796	015731	042	042522	042101	EM2: .ASCIZ	/"READY" FAILED TO SET/
3797	015736	021131	043040	044501		
3798	015744	042514	020104	047524		
3799	015752	051440	052105	000		
3800	015757	042	051124	047101	EM3: .ASCIZ	/"TRANSFER REQUEST" FAILED TO SET/
3801	015764	043123	051105	051040		
3802	015772	050505	042525	052123		
3803	016000	020042	040506	046111		
3804	016006	042105	052040	020117		
3805	016014	042523	000124			
3806	016020	044124	020105	051127	EM4: .ASCIZ	/THE WRONG FLAG SET/
3807	016026	047117	020107	046106		
3808	016034	043501	051440	052105		
3809	016042	000				
3810	016043	103	052517	052116	EM5: .ASCIZ	/COUNT PATTERN FAILED/
3811	016050	050040	052101	042524		
3812	016056	047122	043040	044501		
3813	016064	042514	000104			
3814	016070	040504	040524	050040	EM6: .ASCIZ	/DATA PROBLEM/
3815	016076	047522	046102	046505		
3816	016104	000				
3817	016105	101	042104	042522	EM10: .ASCIZ	/ADDRESS FAILED/
3818	016112	051523	043040	044501		
3819	016120	042514	000104			
3820	016124	041520	020040	020040	DH1: .ASCIZ	/PC TACS/
3821	016132	020040	040524	051503		
3822	016140	000				
3823	016141	120	020103	020040	DH2: .ASCIZ	/PC TACS WAIT ADDRESS/
3824	016146	020040	052040	041501		
3825	016154	020123	020040	053440		
3826	016162	044501	020124	042101		
3827	016170	051104	051505	000123		
3828	016176	041520	020040	020040	DH5: .ASCIZ	/PC TACS EXPECT RCV'D/
3829	016204	020040	040524	051503		
3830	016212	020040	020040	054105		



TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
 DZTAA.P11 POWER DOWN AND UP ROUTINES

3831	016220	042520	052103	020040				
3832	016226	041522	023526	000104				
3833	016234	041520	020040	020040	DH6:	.ASCIZ	/PC	TA0B/
3834	016242	020040	040524	041104				
3835	016250	000						
3836	016251	120	020103	020040	DH10:	.ASCIZ	/PC	ADDRESS/
3837	016256	020040	040440	042104				
3838	016264	042522	051523	000				
3839		016272						.EVEN
3840	016272	001116	001162	000000	DT1:	.WORD		\$ERRPC,\$REGO,0
3841	016300	001116	001162	001202	DT2:	.WORD		\$ERRPC,\$REGO,\$AVPC,0
3842	016306	000000						
3843	016310	001116	001162	001124	DT5:	.WORD		\$ERRPC,\$REGO,\$GDDAT,\$BDDAT,0
3844	016316	001126	000000					
3845	016322	001116	001164	000000	DT6:	.WORD		\$ERRPC,\$REG1,0
3846	016330	001116	001122	000000	DT10:	.WORD		\$ERRPC,\$BDADR,0
3847	016336	001116	001206	000000	DT201:	.WORD		\$ERRPC,TACSL,0
3848								
3849	016344	001116	000000		DT202:	.WORD		\$ERRPC,0
3850								
3851	016350	040524	030461	043040	EM201:	.ASCIZ		"TA11 FAILED TO RESPOND"
3852	016356	044501	042514	020104				
3853	016364	047524	051040	051505				
3854	016372	047520	042116	000				
3855	016377	116	020117	051104	EM202:	.ASCIZ		"NO DRIVE AVAILABLE"
3856	016404	053111	020105	053101				
3857	016412	044501	040514	046102				
3858	016420	000105						
3859	016422	041520	020040	020040	DH201:	.ASCIZ	/PC	TACS/
3860	016430	020040	040524	051503				
3861	016436	000						
3862	016437	120	000103		DH202:	.ASCIZ	/PC/	
3863		000001				.END		



TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAA.C.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

DISPLA	001142	860#	1021*	1029*	2804*	2830*								
DISPRE	000174	724#	1029											
DRIVE	=%000003	711#	798	820	1174*	1178*	1202	1656	1666	1676	1687	1697	1962	2011
		2036	2060	2075	2090	2101	2119	2130	2142	2169	2189	2216	2221	2228
		2244	2259	2274	2289	2304	2319	2335	2340	2347	2370	2375	2382	2339
		2404	2411	2428	2433	2440	2457	2462	2469	2486	2491	2498	2514	2534
		2541	2577	2630	2656	2682	2706	3070	3103	3126	3158	3186	3214	3233
		3320*	3324*											
DRVKEY	001224	888#	889	982*	1124	1137	1158	1189	2958*	2963	2977	2981	3327	
DRVPT	001230	889#	1138*	1158*	1175	1190*	3321	3328*						
DSWR	= 177570	583#	859	1020										
DT1	016272	913	3840#											
DT10	016330	955	3846#											
DT2	016300	919	925	931	3841#									
DT201	016336	962	3847#											
DT202	016344	967	3849#											
DT5	016310	937	949	3843#										
DT6	016322	943	3845#											
EMTVEC	= 000030	672#	1004*	1005*	1227	1228*	1232*	1250	1251*	1255*	1273	1274*	1278*	1296
		1297*	1301*	1319	1320*	1324*	1342	1343*	1347*					
EM1	015712	911	3793#											
EM10	016105	953	3817#											
EM2	015731	917	3796#											
EM201	016350	960	3851#											
EM202	016377	965	3855#											
EM3	015757	923	3800#											
EM4	016020	929	3806#											
EM5	016043	935	3810#											
EM6	016070	941	947	3814#										
ERROR	= 100000	689#	2153											
ERROR1	011516	1228	1251	1274	1297	1320	1343	2826#						
ERRVEC	= 000004	665#	1018	1019*	1030*	1086*	1092*	1222*	1225*	1234*	1245*	1248*	1257*	1268*
		1271*	1280*	1291*	1294*	1303*	1314*	1317*	1326*	1337*	1340*	1349*	2766	2767*
		2769*	2772*											
EXAM	013672	1125	1129	1134	3340#									
EXIT	012322	2976	2981#											
FGAP	= 004000	693#												
FUNCTI	= 000016	705#	1920	1923	1924	1933	1946	1950						
FUNCO	= 000002	703#	705											
FUNC1	= 000004	702#	705											
FUNC2	= 000010	701#	705											
GNS	= ***** U	723	1049	1171	1182	3717	3718	3719	3720	3722	3724	3725	3726	3727
		3728	3729											
GO	= 000001	704#	799	802	821	824	1203	2014	2039	2062	2077	2092	2121	2143
		2151	2170	2172	2190	2218	2222	2229	2246	2261	2276	2291	2306	2321
		2337	2341	2348	2372	2376	2383	2401	2405	2412	2430	2434	2441	2459
		2463	2470	2488	2492	2499	2536	2538	2579	2581	2593	2591	2593	2631
		2633	2636	2657	2659	2662	2683	2685	2688	2707	3071	3074	3104	3127
		3159	3162	3187	3193	3215	3218	3234	3279	3297				
GTSWR	= 104405	1044	1197	3249	3262	3268	3722#							
HERE	001760	1035	1037	1048	1050#									
HGHTIM	012072	2174	2900*	2914#										
HT	= 000011	575#	3391	3432										
ILBS	= 000020	700#	1920	1923	1924	1933	1946	1950	2561	2589	2600	2646	2672	2698
		3287	3308											
INT.EN	= 000100	698#	1920	1923	1924	1933	1946	1950						

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
 DZTAA.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

IOTVEC=	000020	670#	1002*	1003*															
ITEMS2	001336	958#																	
LEADER=	020000	691#	1963	2024	2049	2148	2153	2192	2203	2234	2353	2388	2417	2446					
		2475	2504	3076	3108	3131	3164	3167	3191	3195	3220	3236	3358						
LF =	00.012	576#	3426	3432															
LOOP	012246	2964#	2978																
LOUPL	000500	787#	788																
LOOP2	000600	797#	808																
LOOP3	000700	819#	830																
MANUAL	002202	1144	1152#																
MAXCNT	012116	1166#	2174*	2175*	2176*	2177*	2900	2921#	2928										
MSGASK	015603	2991	3777#																
MSGDRV	015570	2957	3775#																
MSGID	001732	1038	1050#	1157															
MSGOK	015676	3038	3791#																
MSGPRI	015623	3014	3781#																
MSGVEC	015613	3005	3779#																
MTACS	015635	3026	3783#																
MTADB	015643	3029	3785#																
MTAPRI	015663	3035	3789#																
MTAVEC	015652	3032	3787#																
MYES	015704	3047	3792#																
NEXT	012302	2967	2971	2975#															
NOTA	012260	2965	2968#																
NOTB	012272	2969	2972#																
NOTLGL	012314	2962	2973	2979#	2982														
NXTDRV	013626	3069	3102	3125	3157	3185	3213	3318#											
OFFLIN=	001000	695#	3349																
OK	002214	1153	1155#																
PATRN	013502	3258#	3264*	3272*	3285	3296													
PC =	%000007	596#	1065*	1068*	1125*	1129*	1134*	2096*	2125*	2199*	2725*	2728*	2735*	2740					
		2840*	2884*	2910*	2913*	2938*	2941*	2983*	3049*	3069*	3096*	3102*	3107*	3125*					
		3130*	3151*	3157*	3166*	3185*	3190*	3213*	3270*	3289*	3313*	3329*	3361*	3400*					
		3407*	3414*	3428*	3430*	3524*	3765												
		582#																	
PIRQ =	177772	676#																	
PIRQVE=	000240	599#																	
PRO =	000000	600#																	
PR1 =	000040	601#																	
PR2 =	000100	602#																	
PR3 =	000140	603#																	
PR4 =	000200	604#																	
PR5 =	000240	605#																	
PR6 =	000300	606#																	
PR7 =	000340	579#	580	1161*															
PS =	177776	582#																	
PSW =	177776	1157#	3767																
PWRST	002222	671#	1008*	1009*	3734*	3735*	3744*	3750*	3762*	3763*									
PWRVEC=	000024	3130	3190	3295#															
ROBLK	013546	3039	3583	3725#															
ROCHR =	104407	2959	3447	3726#															
ROLIN =	104410	2992	3006	3015	3727#														
RODOCT =	104411	733	3120#	3124	3132														
RDSUB	012774	680#	824	1974	2261	2376	2383	2593	3297										
READ =	000004	699#	800	807	822	829	1204	2015	2021	2040	2046	2063	2078	2094					
READY =	000040	2102	2123	2131	2197	2223	2247	2262	2277	2292	2307	2322	2342	2377					

	2406	2435	2464	2493	2904	2934	3072	3105	3128	3160	3189	3216	3281
RESVEC= 000010	3299	3345											
REWIND= 000016	666*												
	685*	799	821	1203	1994	2014	2062	2092	2143	2170	2190	2218	2337
	2372	2401	2430	2459	2488	2536	2579	2591	2631	2657	2683	2707	3071
RGPBLK 013156	3104	3127	3159	3187	3215								
RO =%000000	735	3180*	3184	3192	3196								
	587*	827*	1087*	1090*	1093	1099*	1124*	1127*	1131*	1133*	1136*	1137*	1138
	1139	1141*	1142	1149*	1193*	1630*	1634	1637*	1639	1715*	1716	1718*	1723
	1735*	1736	1738*	1743	1755*	1756	1758*	1763	1775*	1776	1778*	1783	1795*
	1796	1798*	1803	1815*	1816	1818*	1823	1835*	1836	1838*	1843	1855*	1856
	1858*	1863	1875*	1876	1878*	1882	2013*	2019*	2038*	2044*	2220*	2225*	2339*
	2344*	2349*	2352*	2374*	2379*	2403*	2408*	2432*	2437*	2461*	2466*	2490*	2495*
	2543*	2544*	2550*	2551*	2611*	2619*	2635*	2638	2641	2644*	2661*	2664	2667
	2670*	2687*	2690	2693	2696*	2732*	2735	2826	2864	2865*	2866*	2867*	2868*
	2869*	2870*	2871	2875	2880*	2882*	2886	2889	2960*	2961	2964	2966	2968
	2970	2972	2974	2975	2990	2993*	2995	2997	2998*	2999	3007*	3009	3011
	3012*	3013	3016*	3018	3020*	3021*	3022*	3023*	3024*	3025	3040*	3041	3043
	3048*	3245*	3251*	3255*	3256*	3257	3296*	3304	3342	3383	3384*	3385	3388*
R1 =%000001	3444	3448*	3451	3467*	3694	3695*	3696	3697*	3698*	3699*	3700*	3736	3761*
	588*	1175*	1176	1184	1187	1189*	1190	1632*	1633*	1634	1716*	1717*	1719*
	1720	1721*	1722*	1723	1736*	1737*	1739*	1740	1741*	1742*	1743	1756*	1757*
	1759*	1760	1761*	1762*	1763	1776*	1777*	1779*	1780	1781*	1782*	1783	1796*
	1797*	1799*	1800	1801*	1802*	1803	1816*	1817*	1819*	1820	1821*	1822*	1823
	1836*	1837*	1839*	1840	1841*	1842*	1843	1856*	1857*	1859*	1860	1861*	1862
	1863	1876*	1877*	1879*	1880	1881*	1882*	1883	2012*	2017*	2037*	2042*	2059*
	2065*	2074*	2080*	2243*	2249*	2258*	2264*	2273*	2279*	2288*	2294*	2303*	2309*
	2318*	2324*	2640*	2641	2666*	2667	2692*	2693	2827	2963*	2966*	2970*	2977
	3278*	3283*	3303*	3304	3321*	3322	3325	3327*	3328	3340*	3341*	3344*	3345
	3347	3349	3351	3358	3445	3449*	3453*	3455*	3457*	3460*	3463	3466*	3737
	3760*												
R2 =%000002	589*	1631*	1637	1638	3295*	3301*	3446	3450*	3454*	3456*	3458*	3464	3465*
	3738	3759*											
R3 =%000003	590*	3579	3580*	3581	3584*	3585	3589	3591	3593*	3595*	3642	3651*	3657*
	3658*	3661*	3666*	3667*	3668	3677*	3739	3758*					
R4 =%000004	591*	1062*	1097	1147	1152	1154*	1155	3643	3645*	3646*	3647*	3648	3649*
	3663	3665*	3573*	3676*	3740	3757*							
R5 =%000005	592*	981*	988*	990*	992*	1062	1063*	1066*	1096*	1146*	3644	3650*	3652*
	3654*	3655*	3656*	3657	3675*	3741	3756*						
R6 =%000006	593*	595	996*	997*	998								
R7 =%000007	594*	596											
SAVPC 001202	880*	2901*	2902*	2918	2929*	2930*	2946	3841					
SAVPS 001204	881*	2903*	2917	2931*	2945								
SETBUF 013370	3096	3151	3245*										
SFBG = 000014	684*	1990	2321	2492	2499	3193							
SFFG = 000012	683*	1986	2306	2463	2470	3218							
SFFGSB 013252	736	3208*	3212	3223									
SP =%000006	595*	1000*	1018*	1026*	1030	1091	1221*	1226	1227*	1232	1244*	1249	1250*
	1255	1267*	1272	1273*	1278	1290*	1295	1296*	1301	1313*	1318	1319*	1324
	1336*	1341	1342*	1347	2766*	2769	2771	2772	2800	2801	2805*	2835	2849*
	2852*	2864*	2882	2886*	2901	2903	2917*	2918*	2919*	2929	2931	2945*	2946*
	2947*	2960	2990*	2993	3007	3016	3027*	3030*	3033*	3036*	3040	3048	3064*
	3097*	3120*	3152*	3180*	3208*	3360*	3383*	3384	3385*	3387	3388	3389*	3391
	3393	3395	3401	3403*	3405*	3413*	3417	3421	3422	3426	3442*	3443*	3444*
	3445*	3446*	3448	3451*	3459*	3460	3462	3463*	3465	3466	3467	3484*	3485*
	3486	3493*	3496*	3497*	3501*	3502*	3506	3509*	3513	3515	3517	3518*	3525

TAII BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAA.C.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

STACK = 001100	3527	3529*	3530	3532*	3533*	3534*	3535*	3536*	3551*	3552*	3555*	3556*	3557
START 002234	3561*	3562*	3563	3566	3568	3570*	3579*	3584	3595	3596*	3597*	3598*	3634*
STK LMT= 177774	3635	3636	3637*	3642*	3643*	3644*	3650	3675	3676	3677	3678*	3679*	3694*
SWR 001142	3695	3705*	3706*	3736*	3737*	3738*	3739*	3740*	3741*	3742*	3743	3751*	3755
	3756	3757	3758	3759	3760	3761	3766*						
	570#	1000	1221	1244	1267	1290	1313	1336	3064	3097	3120	3152	3180
	3208												
	1156	1159#	2741										
	581#												
	787	805	859#	998	1020*	1022	1028*	1042	1195	2761	2775	2777	2783
	2790	2831	2838	2843	2847	3247	3251	3253	3260	3264	3266	3318	3353
	3355	3480	3517*	3742	3755*								
	725#	1028	1042	1135	3247	3260	3266	3480	3493				
SWREG 000176	634#												
SW0 = 000001	624#	634											
SW00 = 000001	623#	633											
SW01 = 000002	622#	632											
SW02 = 000004	621#	631											
SW03 = 000010	620#	630											
SW04 = 000020	619#	629											
SW05 = 000040	618#	628											
SW06 = 000100	617#	627											
SW07 = 000200	616#	626											
SW08 = 000400	615#	625											
SW09 = 001000	633#												
SW1 = 000002	614#												
SW10 = 002000	613#												
SW11 = 004000	612#												
SW12 = 010000	611#												
SW13 = 020000	610#												
SW14 = 040000	609#												
SW15 = 100000	632#												
SW2 = 000004	631#												
SW3 = 000010	630#												
SW4 = 000020	629#												
SW5 = 000040	628#												
SW6 = 000100	627#												
SW7 = 000200	626#												
SW8 = 000400	625#												
SW9 = 001000	712#												
TACS = 000004	787*	798*	799*	800	802*	803	807	820*	821*	822	824*	825	
	829	1162*	1202*	1203*	1204	1223*	1229	1269*	1275	1362	1371	1380	1389
	1398	1407	1416	1425	1434	1443	1452	1466*	1467	1471*	1472	1480*	1481
	1486	1493*	1494	1498*	1499	1507*	1508	1513	1520*	1521	1525*	1526	1534*
	1535	1540	1547*	1548	1552*	1553	1561*	1562	1567	1574*	1575	1579*	1580
	1588*	1589	1594	1601*	1602	1606*	1607	1615*	1616	1621	1629*	1632	1638*
	1656*	1657	1666*	1667	1676*	1677*	1678	1687*	1688	1697*	1698	1715	1720*
	1721	1735	1740*	1741	1755	1760*	1761	1775	1780*	1781	1795	1800*	1801
	1815	1820*	1821	1835	1840*	1841	1855	1860*	1861	1875	1880*	1881	1898*
	1900	1909*	1910*	1919*	1920	1924	1932*	1933	1936*	1937	1945*	1946	1950
	1962*	1963	1966*	1967	1970*	1971	1974*	1975	1978*	1979	1982*	1983	1986*
	1987	1990*	1991	1994*	1995	2011*	2014*	2015	2021	2024	2036*	2039*	2040
	2046	2049	2060*	2062*	2063	2075*	2077*	2078	2090*	2092*	2094	2101*	2102
	2119*	2121*	2123	2130*	2131	2142*	2143*	2145	2148	2151*	2153	2169*	2170*
	2172*	2189*	2190*	2192	2195*	2197	2203	2216*	2218*	2221*	2222*	2223	2228*
	2229*	2231	2234	2244*	2246*	2247	2259*	2261*	2262	2274*	2276*	2277	2289*
	2291*	2292	2304*	2306*	2307	2319*	2321*	2322	2335*	2337*	2340*	2341*	2342











TAII BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAA.C.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

\$SCOPE 011232	1002	2759#												
\$SETUP= 000137	931#	1001	1002	1004	1006	1008	1010	1011	1012	1014	1036	1039	2721	
	2760	2823	2846	2854	3475	3609								
\$STUP = 177777	981#													
\$SVLAD 011440	2770	2799#												
\$SWR = 167400	497#	507	558	559	560	561	562	563	564	673	874	875	1011	
	1012	1014	1015	1220	1243	1266	1269	1312	1335	1360	1369	1378	1387	
	1396	1405	1414	1423	1432	1441	1450	1466	1479	1493	1506	1520	1533	
	1547	1560	1574	1587	1601	1614	1628	1654	1664	1674	1685	1695	1713	
	1733	1753	1773	1793	1813	1833	1853	1873	1896	1907	1918	1931	1944	
	1962	2007	2032	2057	2072	2087	2115	2138	2165	2186	2212	2241	2256	
	2271	2286	2301	2316	2331	2366	2395	2424	2453	2482	2511	2530	2573	
	2609	2618	2626	2652	2678	2704	2714	2722	2734	2740	2742	2751	2752	
	2753	2754	2755	2761	2773	2775	2776	2779	2780	2781	2788	2789	2790	
	2801	2804	2807	2814	2815	2816	2817	2818	2831	2838	2843	2847	2859	
	3768													
\$SWRMK= 000000	564	565	2755	2756	2777									
\$TIMES 001166	873#	1011*	1220*	1243*	1266*	1289*	1312*	1335*	1479*	1506*	1533*	1560*	1587*	
	1614*	2007*	2032*	2057*	2072*	2087*	2115*	2138*	2165*	2186*	2212*	2241*	2256*	
	2271*	2286*	2301*	2316*	2331*	2366*	2395*	2424*	2453*	2482*	2511*	2530*	2573*	
	2626*	2652*	2676*	2704*	2722*	2788*	2795	2798*	2807					
\$TKB 001146	862#	3473	3484	3501	3555	3561								
\$TKS 001144	861#	3473	3482	3498	3522*	3553	3559							
\$TN = 000122	497#	507	1216	1220#	1239	1243#	1262	1266#	1285	1289#	1308	1312#	1331	
	1335#	1356	1360#	1363	1365	1369#	1372	1374	1378#	1381	1383	1387#	1390	
	1392	1396#	1399	1401	1405#	1408	1410	1414#	1417	1419	1423#	1426	1428	
	1432#	1435	1437	1441#	1444	1446	1450#	1453	1462	1466#	1473	1475	1479#	
	1487	1489	1493#	1500	1502	1506#	1514	1516	1520#	1527	1529	1533#	1541	
	1543	1547#	1554	1556	1560#	1568	1570	1574#	1581	1583	1587#	1595	1597	
	1601#	1608	1610	1614#	1622	1624	1628#	1650	1654#	1658	1660	1664#	1668	
	1670	1674#	1679	1681	1685#	1689	1691	1695#	1699	1709	1713#	1724	1729	
	1733#	1744	1749	1753#	1764	1769	1773#	1784	1789	1793#	1804	1809	1813#	
	1824	1829	1833#	1844	1849	1853#	1864	1869	1873#	1884	1892	1896#	1901	
	1903	1907#	1912	1914	1918#	1925	1927	1931#	1938	1940	1944#	1951	1958	
	1962#	1996	2003	2007#	2010	2025	2028	2032#	2035	2050	2053	2057#	2064	
	2068	2072#	2079	2083	2087#	2111	2115#	2117	2132	2134	2138#	2140	2161	
	2165#	2167	2182	2186#	2208	2212#	2214	2235	2237	2241#	2248	2252	2256#	
	2263	2267	2271#	2278	2282	2286#	2293	2297	2301#	2308	2312	2316#	2323	
	2327	2331#	2333	2360	2362	2366#	2368	2389	2391	2395#	2397	2418	2420	
	2424#	2426	2447	2449	2453#	2455	2476	2478	2482#	2484	2505	2507	2511#	
	2526	2530#	2532	2567	2569	2573#	2575	2605	2609#	2612	2614	2618#	2620	
	2622	2626#	2648	2652#	2674	2678#	2700	2704#						
\$TPB 001152	864#	3421*	3432											
\$TPFLG 001157	868#	3379	3432											
\$TPS 001150	863#	3419	3432											
\$TRAP 015322	1006	3694#												
\$TRAP2 015344	3705#	3716												
\$TRP = 000014	3709#	3718#	3719#	3720#	3721#	3722	3723#	3724	3725#	3726#	3727#	3728#	3729#	
	3730#													
\$TRPAD 015356	3699	3716#												
\$STNM 001102	841#	1173*	2721*	2750	2777	2799*	2804	2808	2830	2859				
\$TTYIN 015030	3580	3581	3598	3602#										
\$TYPN= ***** U	3721													
\$TYPOS= ***** U	3721													
\$TYPE 013776	3379#	3709	3717											
\$TYPEC 014146	3400	3407	3414	3419#	3420	3524								



TAII BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
 DZTAAAC.P11 CROSS REFERENCE TABLE -- MACRO NAMES

BINIT	717#	1475	1502	1529	1556	1583	1610								
BITBAN	717#	1462	1489	1516	1543	1570	1597								
BITCHK	717#	1356	1365	1374	1383	1392	1401	1410	1419	1428	1437	1446	1650	1660	1670
	1681	1691													
BTOGGL	717#	1462	1489	1516	1543	1570	1597								
CLERR	717#	2208	2327	2362	2391	2420	2449	2478							
COMMEN	1#	677#	717#	3056	3080	3112	3134	3171	3198	3224	3240	3273	3290	3314	
ENDCOM	1#	677#													
ERROR	571#	1095	1145	1230	1253	1276	1299	1322	1345	1364	1373	1382	1391	1400	1409
	1418	1427	1436	1445	1454	1469	1474	1483	1488	1496	1501	1510	1515	1523	1528
	1537	1542	1550	1555	1564	1569	1577	1582	1591	1596	1604	1609	1618	1623	1636
	1659	1669	1680	1690	1700	1725	1745	1765	1785	1805	1825	1845	1865	1885	1902
	1913	1922	1926	1935	1939	1948	1952	1965	1969	1973	1977	1981	1985	1989	1993
	1997	2023	2026	2048	2051	2067	2082	2099	2104	2128	2133	2147	2150	2156	2194
	2202	2205	2233	2236	2251	2266	2281	2296	2311	2326	2358	2361	2387	2390	2416
	2419	2445	2448	2474	2477	2503	2506	2518	2548	2555	2559	2564	2568	2598	2603
	2613	2621	2643	2659	2695	2908	2916	2936	2944	3306	3312				
ESCAPE	1#	677#	1628	1918	1931	1944	2010	2035	2088	2117	2140	2187	2214	2333	2368
	2397	2426	2455	2484	2532	2575	2628	2654	2680						
GETPRI	1#	677#													
GETSWR	1#	677#	1039#												
JUMP	717#														
MORETA	717#	880													
MULT	1#	677#													
NEWTST	1#	677#	1216	1239	1262	1285	1308	1331	1356	1365	1374	1383	1392	1401	1410
	1419	1428	1437	1446	1462	1475	1489	1502	1516	1529	1543	1556	1570	1583	1597
	1610	1624	1650	1660	1670	1681	1691	1709	1729	1749	1769	1789	1809	1829	1849
	1869	1892	1903	1914	1927	1940	1958	2003	2028	2053	2068	2083	2111	2134	2161
	2182	2208	2237	2252	2267	2282	2297	2312	2327	2362	2391	2420	2449	2478	2507
	2526	2569	2605	2614	2622	2648	2674	2700							
POP	1#	677#	3465	3755	3756										
PUSH	1#	677#	3444	3736	3742										
RONLY	717#	1706	1726	1746	1766	1786	1806	1826	1846	1866					
READYC	717#	2053	2068	2237	2252	2267	2282	2297	2312						
READYI	717#	2106													
READYS	717#	2003	2028												
REGADR	717#	1212	1235	1258	1281	1304	1327								
REPORT	1#	677#													
ROMERR	717#	1966	1970	1974	1978	1982	1986	1990	1994						
SAVE	717#	2824													
SCOPE	572#	1219	1242	1265	1288	1311	1334	1359	1368	1377	1386	1395	1404	1413	1422
	1431	1440	1449	1465	1478	1492	1505	1519	1532	1546	1559	1573	1586	1600	1613
	1627	1653	1663	1673	1684	1694	1712	1732	1752	1772	1792	1812	1832	1852	1872
	1895	1906	1917	1930	1943	1961	2006	2031	2056	2071	2086	2114	2137	2164	2185
	2211	2240	2255	2270	2285	2300	2315	2330	2365	2394	2423	2452	2481	2510	2529
	2572	2608	2617	2625	2651	2677	2703	2720							
SETPRI	1#	677#													
SETTRA	3709#	3718	3719	3720	3722	3724	3725	3726	3727	3728	3729				
SETUP	1#	677#	993												
SKIP	1#	677#	1363	1372	1381	1390	1399	1408	1417	1426	1435	1444	1453	1473	1487
	1500	1514	1527	1541	1554	1568	1581	1595	1608	1622	1658	1668	1679	1689	1699
	1724	1744	1764	1784	1804	1824	1844	1864	1884	1901	1912	1925	1938	1951	1996
	2025	2050	2064	2079	2132	2167	2235	2248	2263	2278	2293	2308	2323	2360	2389
	2418	2447	2476	2505	2567	2612	2620								
SLASH	1#	677#	706	715	739	744	970	1206	1211	1351	1355	1456	1461	1642	1649
	1701	1705	1887	1891	1998	2002	2178	2181	2520	2524	3050	3054			



TAII BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAA.P11 CROSS REFERENCE TABLE -- MACRO NAMES

.SCATC	1#	497#	717
.SCMTA	1#	497#	832
.SDB2D	1#		
.SDB20	1#		
.SDIV	1#		
.SEOP	1#	497#	2709
.SERRO	1#	497#	2808
.SERRT	1#		
.SMULT	1#		
.SPOWE	1#	497#	3730
.SRAND	1#		
.SRDDE	1#		
.SRDOC	1#	497#	3432
.SREAD	1#	497#	3470
.SR2AZ	1#		
.SSAVE	1#	497#	
.SSB2D	1#		
.SSB20	1#		
.SSCOP	1#	497#	2745
.SSIZE	1#		
.SSPAC	497#		
.SSUPR	1#		
.STRAP	1#	497#	3686
.STYPB	1#		
.STYPD	1#		
.STYPE	1#	497#	3362
.STYPO	1#	497#	3609
.S4OCA	1#		
.1170	1#		





# MO8

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
 DZTAA.C.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

MACY11 27(732) 25-SEP-76 11:04 PAGE 107

CLR	981	997	1011	1012	1087	1131	1160	1173	1174	1193	1223	1246	1629	1630	1919
	1932	1945	2012	2037	2093	2122	2196	2206	2687	2721	2722	2788	2802	2899	2927
	2958	3245	3258	3320	3341	3459	3450	3496	3497	3651	3732				
CLRB	1136	1141	1269	1292	1315	1338	1899	1910	2556	2787	3398	3424	3593		
CMP	985	998	1022	1036	1042	1091	1152	1195	1226	1249	1272	1295	1318	1341	1634
	1723	1743	1763	1783	1803	1823	1843	1863	1883	2771	2795	2654	2977	2995	3009
	3018	3041	3043	3247	3260	3266	3480	3486	3506	3513	3525	3527	3557	3563	3566
	3568	3581													
CMPB	983	1139	1176	2641	2667	2693	2777	2781	2964	2968	2972	3304	3322	3342	3353
	3355	3391	3393	3401	3422	3426	3488	3520	3585	3591					
COM	1717	1737	1757	1777	1797	1817	1837	1857	1877						
DEC	1063	1066	2019	2044	2725	2866	2913	2941	3283	3301					
DECB	3405	3408	3659	3670											
EMT	571														
HALT	723	1194	2845	2856	3078	3110	3169	3222	3238	3246	3259	3265	3381	3745	3769
INC	1034	1090	1133	2017	2042	2175	2195	2352	2549	2723	2794	2834	2910	2938	3001
	3004	3256	3535	3665	3673	3753									
INCB	2096	2125	2199	2696	2799	2828	3428								
IOT	572														
JMP	727	728	729	730	731	732	733	734	735	736	737	1101	1151	1233	1256
	1279	1302	1325	1348	2740										
JSR	1065	1068	1125	1129	1134	2735	2840	3069	3096	3102	3107	3125	3130	3151	3157
	3166	3185	3190	3213	3400	3407	3414	3524							
MOV	787	798	805	820	827	982	988	990	992	996	1000	1002	1003	1004	1005
	1006	1007	1008	1009	1010	1014	1015	1018	1019	1020	1021	1026	1028	1029	1030
	1062	1086	1092	1096	1099	1124	1137	1138	1146	1149	1154	1155	1158	1159	1161
	1162	1163	1166	1175	1178	1189	1190	1202	1220	1221	1222	1225	1227	1228	1229
	1232	1234	1243	1244	1245	1248	1250	1251	1252	1255	1257	1266	1267	1268	1271
	1273	1274	1275	1278	1280	1289	1290	1291	1294	1296	1297	1298	1301	1303	1312
	1313	1314	1317	1319	1320	1321	1324	1326	1335	1336	1337	1340	1342	1343	1344
	1347	1349	1360	1369	1378	1387	1396	1405	1414	1423	1432	1441	1450	1479	1480
	1506	1507	1533	1534	1560	1561	1587	1588	1614	1615	1628	1631	1632	1654	1656
	1664	1666	1674	1676	1685	1687	1695	1697	1713	1715	1716	1720	1721	1733	1735
	1736	1740	1741	1753	1755	1756	1760	1761	1773	1775	1776	1780	1781	1793	1795
	1796	1800	1801	1813	1815	1816	1820	1821	1833	1835	1836	1840	1841	1853	1855
	1856	1860	1861	1873	1875	1876	1880	1881	1896	1898	1907	1909	1918	1931	1944
	1949	1962	2007	2008	2010	2011	2013	2032	2033	2035	2036	2038	2057	2059	2060
	2072	2074	2075	2087	2088	2090	2115	2116	2117	2119	2130	2138	2139	2140	2142
	2165	2169	2186	2187	2189	2212	2213	2214	2216	2220	2221	2228	2241	2243	2244
	2256	2258	2259	2271	2273	2274	2286	2288	2289	2301	2303	2304	2316	2318	2319
	2331	2332	2333	2335	2339	2340	2347	2349	2366	2367	2368	2370	2374	2375	2382
	2395	2396	2397	2399	2403	2404	2411	2424	2425	2426	2428	2432	2433	2440	2453
	2454	2455	2457	2461	2462	2469	2482	2483	2484	2486	2490	2491	2498	2511	2512
	2514	2530	2531	2532	2534	2541	2543	2550	2573	2574	2575	2577	2609	2611	2618
	2619	2626	2627	2628	2630	2635	2640	2652	2653	2654	2656	2661	2666	2678	2679
	2680	2682	2692	2704	2706	2728	2732	2766	2767	2769	2772	2785	2797	2798	2800
	2801	2804	2805	2824	2825	2826	2827	2830	2835	2849	2852	2864	2871	2875	2880
	2882	2886	2900	2901	2903	2917	2918	2928	2929	2931	2945	2946	2960	2963	2990
	2993	2997	2999	3000	3003	3007	3011	3013	3016	3025	3027	3030	3033	3036	3040
	3048	3064	3065	3066	3068	3070	3097	3098	3099	3101	3103	3120	3121	3122	3124
	3126	3152	3153	3154	3156	3158	3180	3181	3182	3184	3186	3208	3209	3210	3212
	3214	3232	3233	3255	3257	3278	3295	3296	3303	3321	3324	3327	3328	3340	3383
	3384	3388	3403	3442	3443	3444	3445	3446	3448	3463	3464	3465	3466	3467	3493
	3517	3522	3551	3552	3579	3580	3595	3596	3597	3598	3634	3642	3643	3644	3650
	3657	3675	3676	3677	3678	3679	3694	3695	3699	3705	3706	3734	3735	3736	3737
	3738	3739	3740	3741	3742	3743	3744	3750	3751	3755	3756	3757	3758	3759	3760

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTRAC.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

MOV8	3761	3762	3763	3766											
	799	872	821	824	1013	1046	1127	1184	1203	1923	1936	1966	1970	1974	1978
	1982	1986	1990	1994	2014	2039	2062	2077	2092	2121	2143	2151	2170	2172	2190
	2218	2222	2229	2246	2261	2276	2291	2306	2321	2337	2341	2348	2372	2376	2383
	2401	2405	2412	2430	2434	2441	2459	2463	2470	2488	2492	2499	2536	2538	2542
	2581	2583	2585	2587	2591	2593	2631	2633	2636	2638	2657	2659	2662	2664	2683
	2685	2688	2690	2707	2803	2837	2865	2966	2970	3071	3074	3104	3127	3159	3162
	3187	3193	3215	3218	3234	3264	3279	3285	3297	3385	3413	3421	3451	3484	3501
	3555	3561	3584	3589	3635	3636	3639	3640	3641	3645	3648	3649	3668	3697	
NEG	3646														
NOP	2736	2737	2738												
RESET	797	819	1361	1370	1379	1388	1397	1406	1415	1424	1433	1442	1451	1485	1512
	1539	1566	1593	1620	1655	1665	1675	1686	1696	1714	1734	1754	1774	1794	1814
	1834	1854	1874	1897	1908	2009	2034	2058	2073	2089	2100	2118	2129	2141	2168
	2188	2215	2227	2242	2257	2272	2287	2302	2317	2334	2346	2369	2381	2398	2410
	2427	2439	2456	2468	2485	2497	2513	2533	2540	2576	2610	2629	2655	2681	2705
	2734	3067	3100	3123	3155	3183	3211	3231							
ROL	2176	2177	3454	3456	3458	3652	3654	3655	3656	3658					
RTI	1027	2806	2858	2920	2948	3390	3468	3523	3571	3599	3680	3707	3768		
RTS	2884	2983	3049	3270	3289	3313	3329	3361	3430	3700					
SUB	2174	2836	2902	2930											
SWAB	3020														
TRAP	3709	3718	3719	3720	3722	3724	3725	3726	3727	3728	3729				
TST	1040	1088	1093	1097	1142	1147	1164	1167	1191	1200	1967	1971	1975	1979	1983
	1987	1991	1995	2145	2166	2231	2356	2385	2414	2443	2472	2501	2566	2595	2768
	2792	2843	2850	2889	2981	3310	3347	3387	3395	3417	3462	3515	3530	3663	3696
TSTB	803	825	1187	2350	2354	2546	2553	2557	2562	2596	2601	2779	2906	2932	2961
	2974	2975	3253	3318	3325	3379	3419	3482	3498	3553	3559				
.ASCII	876	877													
.ASCIZ	875	878	1050	1172	1183	2743	2893	3603	3604	3605	3607	3772	3775	3777	3773
	3781	3783	3785	3787	3789	3791	3792	3793	3796	3800	3806	3810	3814	3817	3820
	3823	3828	3833	3836	3851	3855	3859	3862							
.BLKB	3602														
.BYTE	841	842	847	848	856	857	865	866	867	868	2742	3600	3601	3681	3682
	3683	3684													
.DSABL	3540														
.ENABL	1	497	3473												
.END	3863														
.ENDC	502	510	554	561	563	564	565	567	571	663	677	687	709	717	728
	741	745	781	784	792	795	813	816	835	839	841	869	873	874	875
	976	880	892	972	973	980	981	1000	1001	1004	1006	1008	1010	1011	1012
	1014	1016	1032	1036	1038	1042	1048	1050	1053	1062	1071	1086	1105	1124	1172
	1183	1208	1212	1214	1217	1218	1219	1220	1221	1237	1240	1241	1242	1243	1244
	1260	1263	1264	1265	1266	1267	1283	1286	1287	1288	1289	1290	1306	1309	1310
	1311	1312	1313	1329	1332	1333	1334	1335	1336	1353	1356	1357	1358	1359	1360
	1361	1364	1366	1367	1368	1369	1370	1373	1375	1376	1377	1378	1379	1382	1384
	1385	1386	1387	1388	1391	1393	1394	1395	1396	1397	1400	1402	1403	1404	1405
	1406	1409	1411	1412	1413	1414	1415	1418	1420	1421	1422	1423	1424	1427	1429
	1430	1431	1432	1433	1436	1438	1439	1440	1441	1442	1445	1447	1448	1449	1450
	1451	1454	1458	1462	1463	1464	1465	1466	1474	1476	1477	1478	1479	1480	1488
	1490	1491	1492	1493	1501	1503	1504	1505	1506	1507	1515	1517	1518	1519	1520
	1528	1530	1531	1532	1533	1534	1542	1544	1545	1546	1547	1555	1557	1558	1559
	1560	1561	1569	1571	1572	1573	1574	1582	1584	1585	1586	1587	1588	1596	1598
	1599	1600	1601	1609	1611	1612	1613	1614	1615	1623	1625	1626	1627	1628	1629
	1644	1650	1651	1652	1653	1654	1655	1659	1661	1662	1663	1664	1665	1669	1671
	1672	1673	1674	1675	1680	1682	1683	1684	1685	1686	1690	1692	1693	1694	1695

1696	1700	1703	1706	1707	1710	1711	1712	1713	1714	1725	1727	1730	1731	1732	
1733	1734	1745	1747	1750	1751	1752	1753	1754	1765	1767	1770	1771	1772	1773	
1774	1785	1787	1790	1791	1792	1793	1794	1805	1807	1810	1811	1812	1813	1814	
1825	1827	1830	1831	1832	1833	1834	1845	1847	1850	1851	1852	1853	1854	1855	
1867	1870	1871	1872	1873	1874	1885	1889	1892	1893	1894	1895	1896	1897	1902	
1904	1905	1906	1907	1908	1913	1915	1916	1917	1918	1919	1926	1928	1929	1930	
1931	1932	1939	1941	1942	1943	1944	1945	1952	1956	1959	1960	1961	1962	1970	
1974	1978	1982	1986	1990	1994	1996	1997	1998	2000	2003	2004	2005	2006	2007	
2008	2009	2011	2026	2027	2029	2030	2031	2032	2033	2034	2036	2051	2052	2054	
2055	2056	2057	2058	2065	2069	2070	2071	2072	2073	2080	2084	2085	2086	2087	
2088	2099	2107	2112	2113	2114	2115	2116	2117	2118	2133	2135	2136	2137	2138	
2139	2140	2141	2159	2162	2163	2164	2165	2166	2168	2180	2182	2183	2184	2185	
2186	2187	2188	2209	2210	2211	2212	2213	2214	2215	2230	2236	2238	2239	2240	
2241	2242	2249	2253	2254	2255	2256	2257	2264	226	2269	2270	2271	2272	2279	
2283	2284	2285	2286	2287	2294	2298	2299	2300	2301	2302	2309	2313	2314	2315	
2316	2317	2324	2328	2329	2330	2331	2332	2333	2334	2355	2361	2363	2364	2365	
2366	2367	2368	2369	2384	2390	2392	2393	2394	2395	2396	2397	2398	2413	2419	
2421	2422	2423	2424	2425	2426	2427	2442	2448	2450	2451	2452	2453	2454	2455	
2456	2471	2477	2479	2480	2481	2482	2483	2484	2485	2500	2506	2508	2509	2510	
2511	2512	2513	2522	2525	2527	2528	2529	2530	2531	2532	2533	2568	2570	2571	
2572	2573	2574	2575	2576	2606	2607	2608	2609	2610	2613	2615	2616	2617	2619	
2621	2623	2624	2625	2626	2627	2628	2629	2649	2650	2651	2652	2653	2654	2655	
2675	2676	2677	2678	2679	2680	2681	2701	2702	2703	2704	2705	2712	2713	2714	
2716	2718	2721	2727	2730	2731	2732	2734	2740	2742	2745	2748	2751	2756	2761	
2763	2774	2777	2778	2779	2781	2783	2790	2794	2799	2800	2804	2807	2808	2811	
2814	2828	2835	2840	2841	2842	2843	2854	2858	2859	2861	2896	2924	2950	2986	
3052	3055	3059	3064	3083	3096	3115	3120	3137	3151	3174	3180	3201	3208	3227	
3231	3243	3245	3276	3278	3293	3295	3317	3331	3365	3385	3435	3437	3470	3473	
3474	3476	3504	3540	3544	3572	3573	3580	3582	3585	3587	3603	3609	3612	3689	
3695	3698	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727	3728	3729	
3733	3742	3743	3749	3755	3756	3766	3768	3775							
.EQUIV	571	572	580	595	596	625	626	627	628	629	630	631	632	633	634
.EVEN	653	654	655	656	657	658	659	660	661	662					
.IF	1050	1172	1183	2894	3774	3839									
	498	507	553	561	562	563	564	565	566	569	635	663	686	706	715
	726	739	744	780	783	791	794	812	815	834	838	840	869	873	874
	875	879	880	970	972	979	981	995	1000	1002	1004	1006	1008	1010	1011
	1012	1014	1032	1035	1036	1037	1039	1042	1049	1051	1061	1069	1085	1103	1123
	1171	1182	1206	1211	1213	1216	1218	1220	1221	1236	1239	1241	1243	1244	1259
	1262	1264	1266	1267	1282	1285	1287	1289	1290	1305	1308	1310	1312	1313	1328
	1331	1333	1335	1336	1351	1355	1356	1358	1360	1361	1363	1365	1367	1369	1370
	1372	1374	1376	1378	1379	1381	1383	1385	1387	1388	1390	1392	1394	1396	1397
	1399	1401	1403	1405	1406	1408	1410	1412	1414	1415	1417	1419	1421	1423	1424
	1426	1428	1430	1432	1433	1435	1437	1439	1441	1442	1444	1446	1448	1450	1451
	1453	1456	1461	1462	1464	1466	1473	1475	1477	1479	1480	1487	1489	1491	1493
	1500	1502	1504	1506	1507	1514	1516	1518	1520	1527	1529	1531	1533	1534	1541
	1543	1545	1547	1554	1556	1558	1560	1561	1568	1570	1572	1574	1581	1583	1585
	1587	1588	1595	1597	1599	1601	1608	1610	1612	1614	1615	1622	1624	1626	1628
	1642	1649	1650	1652	1654	1655	1658	1660	1662	1664	1665	1668	1670	1672	1674
	1675	1679	1681	1683	1685	1686	1689	1691	1693	1695	1696	1699	1701	1705	1706
	1709	1711	1713	1714	1724	1726	1729	1731	1733	1734	1744	1746	1749	1751	1753
	1754	1764	1766	1769	1771	1773	1774	1784	1786	1789	1791	1793	1794	1804	1806
	1809	1811	1813	1814	1824	1826	1829	1831	1833	1834	1844	1846	1849	1851	1853
	1854	1864	1866	1869	1871	1873	1874	1884	1887	1891	1892	1894	1896	1897	1901
	1903	1905	1907	1908	1912	1914	1916	1918	1925	1927	1929	1931	1938	1940	1942
	1944	1951	1955	1958	1960	1962	1968	1970	1972	1974	1976	1978	1980	1982	1984

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAAC.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

1986	1988	1990	1992	1994	1996	1998	2002	2003	2005	2007	2008	2009	2010	2025
2026	2027	2028	2030	2032	2033	2034	2035	2053	2051	2053	2055	2057	2058	2064
2068	2070	2072	2073	2079	2083	2085	2087	2088	2106	2111	2113	2115	2116	2117
2132	2134	2136	2138	2139	2140	2158	2161	2163	2165	2166	2167	2178	2181	2182
2184	2186	2187	2208	2210	2212	2213	2214	2230	2235	2237	2239	2241	2242	2243
2252	2254	2256	2257	2263	2267	2269	2271	2272	2278	2282	2294	2286	2287	2293
2297	2299	2301	2302	2308	2312	2314	2316	2317	2323	2327	2329	2331	2332	2333
2349	2360	2362	2364	2366	2367	2368	2384	2389	2391	2393	2395	2396	2397	2413
2418	2420	2422	2424	2425	2426	2442	2447	2449	2451	2453	2454	2455	2471	2476
2478	2480	2482	2483	2484	2500	2505	2507	2509	2511	2512	2513	2520	2524	2526
2528	2530	2531	2532	2567	2569	2571	2573	2574	2575	2605	2607	2609	2610	2612
2614	2616	2618	2620	2622	2624	2626	2627	2628	2648	2650	2652	2653	2654	2674
2676	2678	2679	2680	2700	2702	2704	2705	2711	2712	2713	2714	2715	2716	2720
2726	2729	2731	2732	2734	2740	2742	2743	2745	2747	2750	2755	2761	2773	2775
2776	2777	2779	2780	2781	2790	2792	2800	2801	2806	2807	2808	2810	2813	2824
2831	2838	2840	2841	2843	2847	2854	2858	2859	2860	2895	2923	2949	2985	3050
3054	3058	3063	3082	3095	3114	3119	3136	3150	3173	3179	3200	3207	3226	3230
3242	3244	3275	3277	3292	3294	3316	3330	3364	3385	3434	3437	3449	3472	3474
3475	3476	3504	3543	3544	3572	3580	3581	3585	3586	3602	3603	3609	3611	3688
3694	3698	3709	3718	3719	3720	3721	3722	3724	3725	3726	3727	3728	3729	3732
3742	3743	3748	3755	3756	3764	3766	3768	3772						
507	554	561	563	564	565	567	569	687	706	715	739	745	781	784
792	795	813	816	835	838	840	869	880	970	973	980	1000	1035	1037
1051	1062	1069	1086	1103	1124	1206	1212	1214	1217	1218	1219	1220	1221	1237
1240	1241	1242	1243	1244	1260	1263	1264	1265	1266	1267	1283	1286	1287	1288
1289	1290	1306	1309	1310	1311	1312	1313	1329	1332	1333	1334	1335	1336	1351
1356	1357	1358	1359	1360	1364	1366	1367	1368	1369	1373	1375	1376	1377	1378
1382	1384	1385	1386	1387	1391	1393	1394	1395	1396	1400	1402	1403	1404	1405
1409	1411	1412	1413	1414	1418	1420	1421	1422	1423	1427	1429	1430	1431	1432
1436	1438	1439	1440	1441	1445	1447	1448	1449	1450	1454	1456	1462	1463	1464
1465	1466	1474	1476	1477	1478	1479	1488	1490	1491	1492	1493	1501	1503	1504
1505	1506	1515	1517	1518	1519	1520	1528	1530	1531	1532	1533	1542	1544	1545
1546	1547	1555	1557	1558	1559	1560	1569	1571	1572	1573	1574	1582	1584	1585
1586	1587	1596	1598	1599	1600	1601	1609	1611	1612	1613	1614	1623	1625	1626
1627	1628	1629	1642	1650	1651	1652	1653	1654	1658	1659	1661	1662	1663	1664
1669	1671	1672	1673	1674	1680	1682	1683	1684	1685	1689	1690	1692	1693	1694
1695	1699	1700	1701	1706	1707	1710	1711	1712	1713	1725	1727	1730	1731	1732
1733	1745	1747	1750	1751	1752	1753	1765	1767	1770	1771	1772	1773	1785	1787
1790	1791	1792	1793	1805	1807	1810	1811	1812	1813	1825	1827	1830	1831	1832
1833	1845	1847	1850	1851	1852	1853	1865	1867	1870	1871	1872	1873	1885	1887
1892	1893	1894	1895	1896	1902	1904	1905	1906	1907	1913	1915	1916	1917	1918
1919	1926	1928	1929	1930	1931	1932	1939	1941	1942	1943	1944	1945	1952	1956
1959	1960	1961	1962	1969	1970	1973	1974	1977	1978	1981	1982	1985	1986	1989
1990	1993	1994	1996	1997	1998	2003	2004	2005	2006	2007	2011	2026	2029	2030
2031	2032	2036	2051	2054	2055	2056	2057	2065	2069	2070	2071	2072	2080	2084
2085	2086	2087	2088	2107	2112	2113	2114	2115	2118	2133	2135	2136	2137	2138
2141	2159	2162	2163	2164	2165	2166	2168	2178	2182	2183	2194	2185	2186	2187
2209	2210	2211	2212	2215	2236	2238	2239	2240	2241	2249	2253	2254	2255	2256
2264	2268	2269	2270	2271	2279	2283	2284	2285	2286	2294	2298	2299	2300	2301
2309	2313	2314	2315	2316	2324	2328	2329	2330	2331	2334	2361	2363	2364	2365
2366	2369	2390	2392	2393	2394	2395	2398	2419	2421	2422	2423	2424	2427	2448
2450	2451	2452	2453	2456	2477	2479	2480	2481	2482	2485	2506	2508	2509	2510
2511	2520	2525	2527	2528	2529	2530	2533	2568	2570	2571	2572	2573	2576	2606
2607	2608	2609	2613	2615	2616	2617	2618	2621	2623	2624	2625	2626	2628	2649
2650	2651	2652	2654	2675	2676	2677	2678	2680	2701	2702	2703	2704	2705	2712
2715	2721	2727	2730	2742	2748	2774	2777	2778	2781	2807	2808	2811	2813	2831

.IFF

	2854	2859	2861	2896	2924	2950	2986	3050	3055	3059	3064	3083	3096	3115	3120
	3137	3151	3174	3180	3201	3208	3227	3231	3243	3245	3276	3278	3293	3295	3317
	3331	3365	3435	3473	3476	3544	3546	3551	3572	3573	3582	3586	3603	3612	3689
	3695	3733	3744	3766											
.IFT	1050	1172	1183	2789	2841	3453	3469	3470	3546	3551					
.IFTF	1050	1172	1183	1969	1970	1973	1974	1977	1978	1981	1982	1595	1986	1989	1990
	1993	1994	1996	1997	2787	2840	3449	3453	3469	3491	3544	3547			
.IIF	497	502	507	558	559	560	561	564	565	566	723	879	1001	1004	1010
	1011	1012	1014	1015	1036	1362	1371	1380	1389	1398	1407	1416	1425	1434	1443
	1452	1656	1657	1666	1667	1676	1677	1687	1688	1697	1698	2713	2714	2721	2722
	2742	2745	2751	2752	2753	2754	2755	2756	2760	2788	2789	2804	2807	2808	2814
	2815	2816	2817	2818	2823	2846	2854	2859	2887	3028	3031	3034	3037	3432	3473
.IRP	3494	3595	3603	3609	3717	3718	3719	3720	3722	3724	3725	3726	3727	3728	3729
	880	981	1216	1239	1262	1285	1308	1331	1356	1365	1374	1383	1392	1401	1410
	1419	1428	1437	1446	1462	1475	1489	1502	1516	1529	1543	1556	1570	1583	1597
	1610	1624	1650	1660	1670	1681	1691	1706	1709	1729	1749	1769	1789	1809	1829
	1849	1869	1892	1903	1914	1927	1940	1958	2003	2028	2053	2068	2083	2111	2134
	2161	2182	2208	2237	2252	2267	2282	2297	2312	2327	2362	2391	2420	2449	2478
	2507	2526	2569	2605	2614	2622	2648	2674	2700	2824	3444	3465	3736	3742	3755
.LIST	3756														
	1	497	507	508	509	510	564	677	706	707	708	709	715	716	717
	723	738	739	740	741	784	791	816	869	871	872	873	970	971	972
	981	1016	1036	1039	1050	1051	1052	1053	1069	1070	1071	1103	1104	1105	1172
	1183	1206	1207	1208	1216	1220	1239	1243	1262	1266	1285	1289	1308	1312	1331
	1335	1351	1352	1353	1356	1360	1365	1369	1374	1378	1383	1387	1392	1396	1401
	1405	1410	1414	1419	1423	1428	1432	1437	1441	1446	1450	1456	1457	1458	1462
	1466	1475	1479	1489	1493	1502	1506	1516	1520	1529	1533	1543	1547	1556	1560
	1570	1574	1583	1587	1597	1601	1610	1614	1624	1628	1642	1643	1644	1650	1654
	1660	1664	1670	1674	1681	1685	1691	1695	1701	1702	1703	1709	1713	1729	1733
	1749	1753	1769	1773	1789	1793	1809	1813	1829	1833	1849	1853	1869	1873	1887
	1888	1889	1892	1896	1903	1907	1914	1918	1927	1931	1940	1944	1958	1962	1998
	1999	2000	2003	2007	2028	2032	2053	2057	2068	2072	2083	2087	2111	2115	2134
	2138	2161	2165	2178	2179	2180	2182	2186	2208	2212	2237	2241	2252	2256	2267
	2271	2282	2286	2297	2301	2312	2316	2327	2331	2362	2366	2391	2395	2420	2424
	2449	2453	2478	2482	2507	2511	2520	2521	2522	2526	2530	2569	2573	2605	2609
	2614	2618	2622	2626	2648	2652	2674	2678	2700	2704	2721	2734	2755	2854	2860
	2895	2923	2985	3050	3051	3052	3059	3083	3115	3137	3174	3201	3227	3243	3276
	3293	3317	3572	3709	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
	3728	3729	3730												
.MACRO	1	565	717	832	3709										
.MCALL	497	677	1016	1039											
.MLIST	1	497	507	508	509	510	564	677	706	707	708	709	715	716	717
	723	738	739	740	741	784	791	816	869	871	872	873	970	971	972
	981	1016	1036	1039	1050	1051	1052	1053	1069	1070	1071	1103	1104	1105	1172
	1183	1206	1207	1208	1216	1220	1239	1243	1262	1266	1285	1289	1308	1312	1331
	1335	1351	1352	1353	1356	1360	1365	1369	1374	1378	1383	1387	1392	1396	1401
	1405	1410	1414	1419	1423	1428	1432	1437	1441	1446	1450	1456	1457	1458	1462
	1466	1475	1479	1489	1493	1502	1506	1516	1520	1529	1533	1543	1547	1556	1560
	1570	1574	1583	1587	1597	1601	1610	1614	1624	1628	1642	1643	1644	1650	1654
	1660	1664	1670	1674	1681	1685	1691	1695	1701	1702	1703	1709	1713	1729	1733
	1749	1753	1769	1773	1789	1793	1809	1813	1829	1833	1849	1853	1869	1873	1887
	1888	1889	1892	1896	1903	1907	1914	1918	1927	1931	1940	1944	1958	1962	1998
	1999	2000	2003	2007	2028	2032	2053	2057	2068	2072	2083	2087	2111	2115	2134
	2138	2161	2165	2178	2179	2180	2182	2186	2208	2212	2237	2241	2252	2256	2267
	2271	2282	2286	2297	2301	2312	2316	2327	2331	2362	2366	2391	2395	2420	2424
	2449	2453	2478	2482	2507	2511	2520	2521	2522	2526	2530	2569	2573	2605	2609

TA11 BASIC LOGIC TEST (PART 1) MAINDEC-11-DZTAA-C  
DZTAAC.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

	2614	2618	2622	2626	2648	2652	2674	2678	2700	2704	2721	2734	2755	2854	2860
	2895	2923	2985	3050	3051	3052	3059	3083	3115	3137	3174	3201	3227	3243	3276
	3293	3317	3572	3709	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
	3728	3729	3730												
.PAGE	553	677	717	738	791	832	892	970	1051	1103	1206	1258	1351	1456	1642
	1701	1887	1998	2178	2520	2700	2745	2808	2860	2895	2949	2995	3050	3330	3362
	3686	3730	3775												
.REM	1	510	747												
.REPT	507	706	715	723	739	871	970	1051	1069	1103	1206	1351	1456	1642	1701
	1887	1998	2178	2520	3050										
.SBTTL	507	554	567	677	717	726	738	784	791	816	832	892	970	994	1032
	1039	1216	1239	1262	1285	1308	1331	1356	1365	1374	1383	1392	1401	1410	1419
	1428	1437	1446	1462	1475	1489	1502	1516	1529	1543	1556	1570	1583	1597	1610
	1624	1650	1660	1670	1681	1691	1709	1729	1749	1769	1789	1809	1829	1849	1869
	1892	1903	1914	1927	1940	1958	2003	2028	2053	2068	2083	2111	2134	2161	2182
	2238	2237	2252	2267	2282	2297	2312	2327	2362	2391	2420	2449	2478	2507	2520
	2526	2569	2605	2614	2622	2648	2674	2700	2709	2745	2808	2860	2895	2923	2951
	2985	2986	3050	3059	3083	3115	3137	3174	3201	3227	3243	3276	3293	3317	3332
	3362	3432	3470	3609	3686	3709	3730								
.TITLE	497														
.WORD	723	724	725	840	843	844	845	846	849	850	851	852	853	854	855
	858	859	860	869	871	872	880	881	2726	2729	2741	3429	3469	3685	3716
	3765	3767	3840	3841	3843	3845	3846	3847	3849						

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

\*, DZTAAC.SEQ/SOL/CRF/PAGNUM/NL:TOC=DZTAAC.SML,DZTAAC.P11  
RUN-TIME: 55 71 9 SECONDS  
RUN-TIME RATIO: 281/137=2.0  
CORE USED: 35K (69 PAGES)

