

# TM02/TU16

DRIVE FUNCTION TIMER  
MD-11-DZTUG-B

EP-DZTUG-B-DL-A  
COPYRIGHT © 1977  
FICHE 1 OF 1

OCT 1977  
**digital**  
MADE IN USA

This microfiche card contains 16 frames of technical data, arranged in a 4x4 grid. Each frame contains a table with multiple columns and rows of text, likely representing drive function parameters or timing data. The text is small and difficult to read due to the resolution of the scan. The frames are separated by thin white lines.



B01

EOF1DZRMJASEQ

00010000

770804

PDP10 411

Z:HDR1DZTUGBSEQ

00010000

770804

.REM %

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZTUG-B-D  
PRODUCT NAME: TMO2/TE16 DRIVE FUNCTION TIMER  
DATE CREATED: AUG 77  
MAINTAINER: DIAGNOSTIC GROUP  
AUTHOR: R. BARNES

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) 1977 BY DIGITAL EQUIPMENT CORPORATION

TMO2 DRIVE FUNCTION TIMER  
TABLE OF CONTENTS

TABLE OF CONTENTS

45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90

ABSTRACT

CHAPTER 1 REQUIREMENTS

    1.1 EQUIPMENT

    1.2 MEMORY STORAGE

    1.3 PRELIMINARY PROGRAMS

CHAPTER 2 LOADING AND STARTING PROCEDURE

    2.1 ACT11 OPERATION

CHAPTER 3 SWITCH SETTINGS

CHAPTER 4 ERRORS

    4.1 ERROR TYPEOUT FORMAT (HARDWARE)

    4.2 ERROR TYPEOUT FORMAT (FUNCTION OUT OF RANGE)

CHAPTER 5 SUBROUTINE ABSTRACTS

CHAPTER 6 MISCELLANEOUS

    6.1 STACK POINTER

    6.2 EXECUTION TIME

CHAPTER 7 PROGRAM DESCRIPTION

    7.1 FUNCTION TIME DOCUMENT

    7.2 TEST SEQUENCE / RELATED ADJUSTMENTS / ASSOCIATED HARDWARE

    7.3 TEST DESCRIPTIONS

E01

TMO2 DRIVE FUNCTION TIMER  
ABSTRACT

PAGE 3

91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135

ABSTRACT

PROGRAM DZTUG MEASURES THE TIME REQUIRED AND GAP SIZES PRODUCED BY THE TMO2/TE16 MAGTAPE DRIVE/SLAVE.

THE TEST WILL CHECK BOTH THE LOGIC GENERATED TIME DELAYS, AND THE DISTANCES TRAVLED BY THE TAPE IN RESPONSE.

ACTUAL TAPE SPEED MAY ALSO BE CHECKED BY USING THE SPEED TESTS WITH AN 800 BPI SKEW TAPE.

DEVICE ERRORS ARE CHECKED AND PRINTED AS THEY OCCUR. IF THE ERROR IS DATA RELATED(PARITY; ETC) THEY ARE PRINTED AS SOFT ERRORS.

IF THE TIME CHECK IS OUT OF RANGE, IT IS PRINTED AS AN OUT OF RANGE ERROR.

TMO2 DRIVE FUNCTION TIMER  
REQUIREMENTS

PAGE 4

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

## CHAPTER 1

## REQUIREMENTS

PDP-11 FAMILY CENTRAL PROCESSOR WITH 4K MEMORY WITH UP TO 64 TMO2/TE16  
CONTROLLER/MAGTAPE STATIONS.

\*\*\*PROGRAM CAN BE RUN ON A PROCESSOR THAT DOES NOT HAVE A HARDWARE SWITCH REGISTER.  
A SOFTWARE SWITCH REGISTER (SWREG) LOC. 176 IS AUTOMATICALLY SELECTED(REFER TO  
CHAPTER 3.FOR DESCRIPTION OF HOW TO DYNAMICALLY LOAD LOC.176)\*\*\*

## 1.1 OPTIONAL EQUIPMENT USED

1. NONE

## 1.2 STORAGE

PROGRAM LOADS AND RUNS IN THE FIRST 4K OF MEMORY.

## 1.3 PRELIMINARY PROGRAMS (TO ASSURE HARDWARE OPERATION)

MAINDEC-11-DZTUC CONTROL LOGIC TEST  
MAINDEC-11-DZTUB BASIC FUNCTION TEST

GO1

TMO2 DRIVE FUNCTION TIMER  
LOADING AND STARTING PROCEDURE

PAGE 5

CHAPTER 2

LOADING AND STARTING PROCEDURE

THE PROCEDURE IS AS FOLLOWS:

LOAD PROGRAM USING THE ABSOLUTE LOADER  
LOAD ADDRESS = 200  
SET OPERATING SWITCHES  
PRESS START

\*\*\*IF THE SOFTWARE SWITCH REGISTER IS USED THEN THE PROGRAM WILL TYPE SWR=XXXXXX NEW=  
THIS WILL ALLOW LOC. 176 TO BE CHANGED BEFORE THE START OF THE TESTING. (REFER TO CHAP

PROGRAM WILL REQUEST DRIVE (TMO2) AND SLAVE (TE16) NUMBERS TO BE  
TESTED. TYPE DRIVE/SLAVE NUMBERS WITH A COMMA (,) BETWEEN EACH  
DRIVE/SLAVE TO BE TESTED.

REQUESTS FOR TAPE SPEED TESTS AND NRZ ONLY MODE WILL BE MADE.  
RESPONSE TO TAPE SPEED ONLY REQUEST WITH A ONE (1) WILL CAUSE  
THE PROGRAM TO EXECUTE TEST 31 AND 32 ONLY. THIS IS THE ONLY WAY  
TO TEST TAPE SPEED.  
NRZ ONLY MODE WILL CAUSE THE PROGRAM TO SKIP THE 1600 BPI DATA TIME TEST.  
TYPE CONTROL U (↑U) TO DELETE LINE TYPED OR RUBOUT TO DELETE LAST  
CHARACTER(S).

PROGRAM WILL PUBLISH TIMES REQUIRED AND REPORT ERRORS.

2.1 ACT11 OPERATION

IF THE PROGRAM IS RUN IN QUICK VERIFY MODE, FUNCTION TESTS ARE NOT  
ITERATED.

183  
184  
185  
186  
187  
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

H01

TMO2 DRIVE FUNCTION TIMER  
SWITCH SETTINGS

PAGE 6

235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287

CHAPTER 3

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.



288	SW15	HALT ON ERROR	THIS SWITCH WHEN SET WILL HALT THE
289	(100000)		PROCESSOR WHEN AN ERROR IS DETECTED.
290			THE PC+2 AND PSW AT THE TIME OF THE
291			ERROR IS STORED ON THE STACK. PRESSING
292			CONTINUE WILL CAUSE THE ERROR TO BE
293			TYPED (IF SELECTED) AND FURTHER TESTING
294			RESUMED.
295			
296	SW14	LOOP SUBTEST	THIS SWITCH WHEN SET LOOPS THE CURRENT
297	(040000)		SUBTEST REGARDLESS OF ERROR CONDITION.
298			
299	SW13	INHIBIT ERROR	THIS SWITCH WHEN SET INHIBITS ERROR
300	(020000)	TYPEOUT	TYPEOUT.
301			
302	SW11	INHIBIT SUB-	THIS SWITCH WHEN SET CAUSES EACH SUBTEST
303	(004000)	TEST ITERATION	TO BE EXECUTED ONLY ONCE. (INITIAL
304			STARTUP ONLY).
305			
306	SW10	INHIBIT	THIS SWITCH WHEN SET WILL INHIBIT THE
307	(002000)	FUNCTION TIME	PRINTING OF THE FUNCTION TIMES. (SEE
308		PUBLICATION	CHAPTER 8.)
309			
310	SW09	RING BELL	THIS SWITCH WHEN SET WILL RING THE BELL
311	(001000)	ON ERROR	ON THE TTY WHEN AN ERROR IS DETECTED.
312			
313	SW07	HALT AFTER	THIS SWITCH WHEN SET WILL CAUSE THE
314	(000200)	SELECTED TEST	PROGRAM TO HALT AFTER THE TEST SELECTED
315			IN SW05-SW00 IS EXECUTED.
316			
317	SW06	CONTINUOUS	THIS SWITCH WHEN SET WILL CAUSE THE
318	(000100)	CYCLE	PROGRAM TO RUN CONTINUOUSLY UNTIL
319			STOPPED BY THE OPERATOR.
320			
321	SW5-0	TEST SELECT	THE PROGRAM WILL HALT AFTER EXECUTION
322			OF THE TEST SELECTED WHEN SW07 IS SET.

TMO2 DRIVE FUNCTION TIMER  
ERRORS

PAGE 7

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  
362  
363  
364  
365  
366  
367  
368  
369  
370

CHAPTER 4

ERRORS

TWO TYPES OF ERRORS ARE DETECTED BY THIS PROGRAM, HARDWARE ERRORS AND INCORRECT FUNCTION TIMES.

4.1 ERROR TYPEOUT FORMAT (HARDWARE): DATA RELATED ERRORS (IE: PARITY ERROR) ARE PRINTED AND HAVE NO EFFECT ON TIME.

TEST # XXXXXX DEVICE ERROR

CS1 WE BA FC CS2 DS ER1  
AAAAAA BBBB BB CCCCC DDDDD EEEEE FFFFF GGGGG

WHERE:

XXXXXX = TEST NUMBER  
AAAAA-IIIIII = CONTENTS OF TAPE REGISTER 172440-172454

4.2 ERROR TYPEOUT FORMAT (FUNCTION TIME OUT OF RANGE)

TEST # XXXXXX OUT OF RANGE ERROR

RANGE = <AAAAA-BBBBB> ACTUAL = CCCCC

K01

TMO2 DRIVE FUNCTION TIMER  
SUBROUTINE ABSTRACTS

PAGE 8

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  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426

CHAPTER 5  
SUBROUTINE ABSTRACTS

5.1 .SCOPE

THE SCOPE ROUTINE IS CALLED BY THE SCOPE (EMT) INSTRUCTION AT THE START OF EACH SUBTEST. THE .SCOPE ROUTINE PERFORMS THE FOLLOWING FUNCTIONS:

1. LOADS R5 WITH BASE ADDRESS
2. TYPES TIME LINE <SW08>
3. PROVIDES CONTINUOUS LOOP <SW14>
4. MOVES FUNCTION TIME INTO TABLE
5. OUTPUTS LINE ITEM IF SELECTED
6. PROVIDES HALT ON TEST <SW07>
7. DELAYS 350MS BEFORE STARTING TEST
8. INIT'S DRIVE/SLAVE
9. CLEARS THE ERROR FLAG (ERFLG)

THE ROUTINE MONITORS SW14, SW11, SW10, SW08, AND SW07.

\*\*\*THIS ROUTINE WILL CHECK FOR CNTL G(<↑G>) BY DOING A JSR PC,CKSWR (REFER TO CHAPTER 3 FOR DESCRIPTION).

5.2 PUBLISH

THE PUBLISH ROUTINE IS CALLED FROM THE SCOPE ROUTINE IF SW10 IS EQUAL TO 0 (PUBLISH TIME DOCUMENT). THE ROUTINE WILL PRINT A "SINGLE LINE ITEM" EACH TIME IT IS CALLED.

427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440TMO2 DRIVE FUNCTION TIMER  
SUBROUTINE ABSTRACTS

PAGE 9

## 5.3 .HLT

THE HLT ROUTINE IS CALLED BY THE HLT (TRAP) INSTRUCTION WHEN AN ERROR IS DETECTED. A HLT (TRAP) INSTRUCTION FORMATS THE ERROR INFORMATION AS SHOWN IN SEC 4.1. A HLT+1 (TRAP+1) FORMATS THE ERROR AS SHOWN IN SEC 4.2.

\*\*\*THIS ROUTINE WILL CHECK FOR A CNTL G (<IG>) BY DOING A JSR PC,CKSWR (REFER TO CHAPTER 3 FOR DESCRIPTION)>

TMO2 DRIVE FUNCTION TIMER  
MISCELLANEOUS

PAGE 10

441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474

CHAPTER 6  
MISCELLANEOUS

6.1 STACK POINTER

THE STACK POINTER IS INITIALLY SET TO 500 AND IS RESET TO 500 BY THE SCOPEA ROUTINE.

6.2 EXECUTION TIME

WHEN SW11=1 (INHIBIT ITERATIONS) THE TIME REQUIRED IS 2 MIN.

WHEN SW11=0 (ITERATE SUBTESTS) THE TIME REQUIRED IS 9 MIN.

TMO2 DRIVE FUNCTION TIMER  
PROGRAM DESCRIPTION

475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
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

CHAPTER 7

PROGRAM DESCRIPTION

7.1 SAMPLE TIME DOCUMENT

TYPE FIRST ADDRESS OF CONTROLLER 172440  
TYPE TMO2 DRIVE #'S TO BE TESTED 0  
FOR TMO2 DRIVE 0- TYPE SLAVE #'S TO BE TESTED 7  
TAPE SPEED TESTS ONLY? (YES/NO = 1/0) 0  
NRZ ONLY? (YES/NO = 1/0) 0

\*\*\*\*\*  
\* TMO2 DRIVE FUNCTION TIMES- DRIVE # 0 SLAVE # 7 9 CHAN. SER. # 5009  
\*

* FUNCTION	TIME(SPECIFICATION)	TIME(ACTUAL)
* WRITE FROM BOT	RANGE=<154000-150000>	ACTUAL=152740
* WRITE START	RANGE=<009500-008700>	ACTUAL=009120
* WRITE SHUTDOWN	RANGE=<008900-008500>	ACTUAL=008840
* WRITE SETTLEDOWN	RANGE=<013500-008100>	ACTUAL=010970
* READ FROM BOT	RANGE=<037000-033000>	ACTUAL=035580
* READ START	RANGE=<003200-002600>	ACTUAL=002740
* READ SHUTDOWN	RANGE=<004650-004250>	ACTUAL=004360
* READ SETTLEDOWN	RANGE=<013500-008100>	ACTUAL=010970
* READ REV START	RANGE=<003200-002600>	ACTUAL=002740
* READ REV SHUTDOWN	RANGE=<003700-003300>	ACTUAL=003520
* READ REV SETTLEDOWN	RANGE=<013500-008100>	ACTUAL=010970
* TURN AROUND DELAY F-R	RANGE=<016700-010700>	ACTUAL=013600
* TURN AROUND DELAY R-F	RANGE=<016700-010700>	ACTUAL=013660
* GAP SIZE-STOP HALF	RANGE=<012900-009500>	ACTUAL=012200
* GAP SIZE-START HALF	RANGE=<011800-008500>	ACTUAL=010520
* GAP SIZE-INTERRECORD	RANGE=<014800-013700>	ACTUAL=014500
* GAP CONSISANCY	RANGE=<014000-012400>	ACTUAL=013040
* DATA TIME-200 BPI	RANGE=<024100-023100>	ACTUAL=023460
* DATA TIME-556 BPI	RANGE=<024000-023000>	ACTUAL=023350
* DATA TIME-800BPI	RANGE=<024000-023000>	ACTUAL=023400
* DATA TIME-1600BPI	RANGE=<025100-024100>	ACTUAL=024470
* ERASE GAP TIME	RANGE=<101000-099000>	ACTUAL=099510
* WRITE FILE MARK	RANGE=<105000-103000>	ACTUAL=103990

B02

DZTUG-B TMO2/TE16 DRIVE FUNCTION TIMER MACY11 30(1046) 15-JUL-77 12:55 PAGE 13  
DZTUGB.P11 15-JUL-77 12:52

SEQ 0013

531

532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547

TMO2 DRIVE FUNCTION TIMER

PAGE 12

7.1.1 SAMPLE TIME DOCUMENT FOR TAPE SPEED TESTS

TYPE FIRST ADDRESS OF CONTROLLER 172440  
 TYPE TMO2 DRIVE #'S TO BE TESTED 0  
 FOR TMO2 DRIVE 0- TYPE SLAVE #'S TO BE TESTED 7  
 SPEED TESTS ONLY? (YES/NO = 1/0) 1  
 \*\*\*\*\*  
 \*TMO2 DRIVE FUNCTION TIMES- DRIVE # 0 SLAVE # 7 9 CHAN. SER. # 5009  
 \*  

*FUNCTION	TIME(SPECIFICATION)	TIME(ACTUAL)
*TAPE SPEED FWD	RANGE=<022700-021700>	ACTUAL=022500
*TAPE SPEED REV	RANGE=<022700-021700>	ACTUAL=022500



548  
549  
550  
551  
552  
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

TM02 DRIVE FUNCTION TIMER

7.2 TEST SEQUENCE WITH RELATED ADJUSTMENTS AND ASSOCIATED HARDWARE

TEST NO./NAME	RELATED ADJUSTMENTS
1. WRITE FROM BOT	*NONE
2. WRITE START	* "
3. WRITE SHUTDOWN	* "
4. WRITE SETTLEDOWN	* "
5. READ FROM BOT	* "
6. READ START	* "
7. READ SHUTDOWN	* "
10. READ SETTLEDOWN	* "
11. READ REVERSE START	* "
12. READ REVERSE SHUTDOWN	* "
13. READ REVERSE SETTLEDOWN	* "
14. TURN AROUND F-R	* "
15. TURN AROUND R-F	* "
16. GAP SIZE-STOP HALF	*FWD/REV SPEED-START/STOP-RAMPS
17. GAP SIZE-START HALF	*SAME AS IN TEST 16
20. GAP SIZE INTERRECORD	*FWD/REV SPEED

E02

603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633

TMO2 DRIVE FUNCTION TIMER

PAGE 14

- 21. GAP CONSISTENCY \*SAME AS IN TEST 16
- \*TEST NUMBER 22 IS RESERVED FOR FUTURE USE
- 23. DATA TIME 200 BPI \*NONE
- 24. DATA TIME 556 BPI \* "
- 25. DATA TIME 800 BPI \* "
- 26. DATA TIME 1600 BPI \* "
- 27. ERASE GAP TIME \* "
- 30. WRITE FILE MARK \* "
- 31. TAPE SPEED-FORWARD \*FWD SPEED
- 32. TAPE SPEED-REVERSE \*REVERSE SPEED

\*\*\*\*\*NOTE: IF TIME PROBLEMS APPEAR IN T1 THRU T30, RUN TAPE SPEED TESTS FIRST\*\*\*\*\*

634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679

## TMO2 DRIVE FUNCTION TIMER

PAGE 15

## 7.3 TEST DESCRIPTIONS:

THE FIRST THIRTEEN (13) TESTS (T1 - T15) ARE CHECKS OF THE ROM CIRCUITS IN THE TE16 (M9811), THE ACCL COUNTER IN THE TMO2 (M8903), AND THE SETTLEDOWN ONE SHOT (M8910).

## T1. WRITE FROM BOT:

THIS TEST WILL MEASURE ACCELERATION DELAY REQUIRED TO MOVE THE TAPE APPROXIMATELY SEVEN (7) INCHES FORWARD FROM DEAD STOP AT BOT BEFORE STARTING TO TRANSFER DATA.

1. ASSURE TAPE IS STOPPED AT BOT.
2. ISSUE A WRITE COMMAND
3. MONITOR BIT 15 OF TC (ACCL)
4. TIME FROM GO TO ACCL RESET IS BOT DELAY
5. STOP

## T2. WRITE START:

THIS TEST WILL MEASURE ACCELERATION DELAY JUST AS IN T1. HOWEVER THE TIME WILL BE LESS WHEN NOT STARTING FROM BOT.

1. LEAVE TAPE AT ITS PRESENT POSITION. ASSURE THAT IT IS STOPPED
2. ISSUE A WRITE COMMAND
3. MONITOR BIT 15 OF TC (ACCL)
4. TIME FROM GO TO RESET OF ACCL IS START DELAY
5. STOP

## T3. WRITE SHUTDOWN:

THIS TEST WILL MEASURE THE TIME FROM EOR (LAST CHARACTER WRITTEN ON TAPE) TO THE START OF SETTLEDOWN TIME. THIS ASSURES, IN PART, A PROPER INTERROCORD GAP.

1. LEAVE TAPE AT ITS PRESENT POSITION. ASSURE THAT IT IS STOPPED
2. ISSUE A WRITE COMMAND.
3. MONITOR FRAME COUNTER AND BIT 4 OF DS (SDWN)
4. TIME FROM FC=0 TO ASSERTION OF SDWN IS THE SHUTDOWN TIME.
5. STOP

680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710

## T4. WRITE SETTLEDOWN:

THIS TEST WILL MEASURE THE SLOWDOWN TIME. THE TIME FROM THE START OF SLOWDOWN UNTIL THE TAPE SHOULD BE STOPPED. THIS IS A PART OF THE GAP TIMING IN LOGIC. THE MECHANICAL POSITIONING OF THE TAPE IN THE GAP DISTANCE WILL BE MEASURED IN A LATER TEST.

1. LEAVE TAPE AT ITS PRESENT POSITION. ASSURE THAT IT IS STOPPED
2. ISSUE A WRITE COMMAND
3. MONITOR BIT 4 OF DS (SDWN)
4. TIME FROM SET OF SDWN TO RESET OF SDWN IS THE SETTLEDOWN DELAY
5. STOP

## T5. READ FROM BOT

THIS MEASUREMENT IS MADE EXACTLY AS THE WRITE MEASUREMENT IN T1. USE THE SAME RECORD THAT WAS WRITTEN IN T1.

1. REWIND TO BOT
2. ASSURE TAPE HAS HAD TIME TO COME TO A COMPLETE STOP
3. READ FORWARD 1 RECORD.
4. MONITOR BIT 15 OF TC (ACCL)
5. TIME FROM GO TO ACCL IS BOT DELAY
6. STOP

711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743

## T6. READ START

THIS TEST MEASURES THE SAME DELAY AS IN T2.

1. WRITE 1 RECORD, THEN BACKSPACE OVER IT, ASSURE TAPE IS STOPPED.
2. ISSUE A READ FORWARD OF THE RECORD WRITTEN IN STEP 1.
3. MONITOR BIT 15 OF TC (ACCL)
4. TIME FROM GO TO RESET OF ACCL IS START DELAY
5. STOP

## T7. READ SHUTDOWN:

THIS TEST MEASURES THE SAME DELAY AS IN T3.

1. WRITE 1 RECORD, THEN BACKSPACE OVER IT, ASSURE TAPE IS STOPPED.
2. READ FORWARD THE RECORD WRITTEN IN STEP 1.
3. MONITOR FRAME COUNT AND BIT 4 OF DS (SDWN).
4. TIME FROM FC=RECORD SIZE (LAST FRAME READ) TO SDWN=1 IS THE SHUTDOWN TIME.
5. STOP

## T10. READ SETTLEDOWN:

THIS TEST MEASURES THE SAME DELAY AS IN T4.

1. WRITE 1 RECORD, THEN BACKSPACE OVER IT, ASSURE TAPE IS STOPPED.
2. READ FORWARD THE RECORD WRITTEN IN STEP 1.
3. MONITOR BIT 4 OF DS (SDWN)
4. TIME FROM SET OF SDWN TO RESET OF SDWN IS THE SETTLEDOWN DELAY.
5. STOP

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  
791  
792  
793  
794  
795  
796  
797

## TMO2 DRIVE FUNCTION TIMER

PAGE 17

## T11. READ REVERSE START:

THIS TEST WILL MEASURE THE START DELAY IN THE REVERSE DIRECTION.

1. WRITE 1 RECORD, ASSURE TAPE IS STOPPED.
2. READ REVERSE THE RECORD WRITTEN IN STEP 1.
3. MONITOR BIT 15 OF TC (ACCL)
4. THE TIME FROM GO TO RESET OF ACCL IS THE START TIME
5. STOP

## T12. READ REVERSE SHUTDOWN

THIS TEST WILL MEASURE THE READ SHUTDOWN IN THE REVERSE DIRECTION.

1. WRITE 1 RECORD, ASSURE TAPE IS STOPPED.
2. READ REVERSE THE RECORD WRITTEN IN STEP 1.
3. MONITOR FRAME COUNTER AND BIT 4 OF DS (SDWN).
4. TIME FROM FC=RECORD SIZE (LAST FRAME READ) TO SDWN=1 IS THE READ REVERSE SHUTDOWN TIME.
5. STOP

## T13. READ REVERSE SETTLEDOWN:

THIS TEST WILL MEASURE THE READ SETTLEDOWN IN THE REVERSE DIRECTION.

1. WRITE 1 RECORD, ASSURE TAPE IS STOPPED.
2. READ REVERSE THE RECORD WRITTEN IN STEP 1.
3. MONITOR BIT 4 OF DS (SDWN)
4. TIME FROM SET OF SDWN TO RESET OF SDWN IS THE SETTLEDOWN DELAY
5. STOP

## T14. TURN AROUND DELAY-FORWARD TO REVERSE

THIS TEST WILL MEASURE THE TIME REQUIRED FOR THE TAPE TO CHANGE DIRECTION.

1. LEAVE TAPE AT ITS PRESENT POSITION. ASSURE THAT IT IS STOPPED
2. ISSUE A WRITE FORWARD OF AT LEAST 20 FRAMES
3. MONITOR BIT 7 OF DS (DRY)
4. WHEN DRY IS ASSERTED (EOR), IMMEDIATELY ISSUE A READ REVERSE OF THAT RECORD.
5. MONITOR BIT 15 OF TC (ACCL).
6. TIME FROM GO OF READ REVERSE TO RESET OF ACCL IS THE TURNAROUND TIME.
7. STOP

798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813

## TMO2 DRIVE FUNCTION TIMER

PAGE 18

T15. TURN AROUND DELAY-REVERSE TO FORWARD

THIS TEST WILL MEASURE THE TIME AS IN T14, BUT IN THE  
OPPOSITE DIRECTION.

1. WRITE 1 RECORD.
2. ASSURE TAPE IS STOPPED
3. READ REVERSE
4. MONITOR DRY (BIT 7 OF DS)
5. WHEN DRY = 1, ISSUE A READ FORWARD
6. MONITOR ACCL (BIT 15 OF TC)
7. TIME FROM GO FORWARD TO ACCL = 1 IS THE TURN AROUND TIME.
8. STOP.

## TMO2 DRIVE FUNCTION TIMER

PAGE 19

814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869

## GAP MEASUREMENTS:

THE PREVIOUS THIRTEEN (13) TESTS WERE MEASUREMENTS OF LOGIC DELAYS PERFORMED BY THE TMO2 OR TE16 IN ORDER TO ALLOW FOR PROPER ACCELERATION AND DECELERATION OF TAPE ACCORDING TO THE DESIRED INTERCORD GAP (.6 INCHES). THIS TEST, HOWEVER, WILL MEASURE THE PHYSICAL SIZE OF THE INTERCORD GAP THAT EXISTS ON TAPE AS A RESULT OF THE START/STOP TIMES OF THE CAPSTAN ITSELF. BECAUSE THE INTERCORD GAP IS CREATED BY TWO ACTIONS, THE START OF MOTION AND THE STOP OF MOTION IT IS NECESSARY TO MAKE TWO SEPERATE MEASUREMENTS. A THIRD MEASUREMENT, MADE ON THE FLY, OF THE ENTIRE LENGTH OF THE GAP WILL ALSO BE MADE.

## T16. GAP SIZE (STOP HALF)

THIS TEST WILL MEASURE THE DISTANCE TRAVLED BY THE TAPE IN A STOP CYCLE. IN OTHER WORDS, THE DISTANCE INTO THE IRG.

1. WRITE 1 RECORD.
2. ASSURE TAPE IS STOPPED.
3. ISSUE A READ REVERSE OVER THE RECORD
4. MONITOR THE FRAME COUNT FOR THE FIRST FRAME READ (FC = 1)
5. THE TIME FROM GO=1 TO FC=1 IS THE LENGTH OF THE GAP
6. STOP

## T17. GAP SIZE (START HALF)

THIS TEST WILL MEASURE THE DISTANCE OF TAPE TRAVEL DURING START UP.

1. WRITE 1 RECORD, THEN REVERSE OVER IT, ASSURE TAPE IS STOPPED.
2. ISSUE A READ FORWARD
3. MONITOR FC FOR FC=1
4. TIME FROM GO=1 TO FC=1 IS START DISTANCE
5. STOP

## T20. GAP SIZE (INTERRECORD)

THIS TEST WILL MEASURE THE ENTIRE LENGTH OF THE IRG ON THE FLG. THE TIME VALUE OF THIS TEST SHOULD NOT BE EQUAL TO A SUMMATION OF T16 AND T17 DUE TO THE FACT THAT THE ACCELERATION AND DECELERATION CURVES ARE NOT IN EFFECT. THE VALUE HERE SHOULD ACTUALLY BE LESS THAN THE SUM OF T16 AND T17.

1. WRITE 2 RECORDS.
2. READ REVERSE OVER THE SECOND RECORD
3. MONITOR DRY (BIT 7 OF DS)
4. WHEN DRY = 1, ISSUE A SECOND READ REVERSE
5. MONITOR FRAME COUNT
6. TIME FROM GO=1 OF SECOND READ REVERSE TO FC=1 IS THE LENGTH OF THE GAP.



L02

DZTUG-B TMO2/TE16 DRIVE FUNCTION TIMER MACY11 30(1046) 15-JUL-77 12:55 PAGE 23  
DZTUGB.P11 15-JUL-77 12:52

SEQ 0023

870

7. STOP

## TM02 DRIVE FUNCTION TIMER

PAGE 20

871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
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

## T21. GAP CONSISTENCY:

NOW THAT WE HAVE ESTABLISHED THAT THE INTERCORD GAP IS THE PROPER SIZE, LET US DETERMINE THE CONSISTENCY OF THE GAP UNDER VARIOUS COMMAND EXECUTION TIMES. BY WRITING A SERIES OF RECORDS, EACH WITH A DIFFERENT DELAY BETWEEN EXECUTION, WE CAN ESTABLISH THE CONSISTENCY OF THE GAPS BY READING THESE RECORDS AND MONITORING THEIR INTERRECORD GAPS, ON THE FLY.

1. REWIND TAPE TO BOT.
  2. WRITE ONE (1) RECORD TO GET TAPE OFF BOT
  3. WRITE SIXTEEN (16) RECORDS WITH A PROGRESSIVE DELAY OF FROM 0 TO 16 MILLISECONDS (APPROX) BETWEEN COMMANDS.
  4. BACKSPACE 16 RECORDS AND ALLOW THE TAPE TO STOP.
  5. READ FORWARD (NON-STOP) OVER THESE 16 RECORDS, EACH TIME MONITORING THE TIME FROM THE END OF RECORD (DRY) UNTIL THE FRAME COUNT NEXT GOES FROM 0 TO 1 (FC=1).
  6. THE TIMES FROM DRY TO FC=1 IS THE GAP TIME AND IT SHOULD REMAIN CONSISTANT FOR ALL RECORDS.
  7. STOP
- \*\* (SEE GTIMTBL IN DZTUG LISTING FOR GAP TIMES)\*\*

T22. RESERVED FOR FUTURE USE\*\*\*\*\*

## T23. DATA TIME AT 200 BPI:

THIS TEST WILL MEASURE THE TIME REQUIRED TO WRITE ONE (1) INCH OF TAPE AT 200 BPI. BY WRITING A RECORD OF ENOUGH FRAMES TO MOVE THE TAPE 1 INCH (200 FRAMES), DATA RATE CAN BE VARIFIED.

1. REWIND TO BOT AND ALLOW TAPE TO STOP
2. WRITE A RECORD AT 200 BPI.
3. MONITOR DRY (BIT 7 OF DS) FOR EACH RECORD
4. THE TIME FROM FC=FC+1 TO DRY WILL BE THE TIME REQUIRED FOR 1 INCH AT THE SELECTED DENSITY
5. STOP

T24. DATA TIME AT 556 BPI:  
REPEAT STEPS 1 THRU 5 OF T23 AT 556 BPI.

T25. DATA TIME AT 800 BPI:  
REPEAT STEPS 1 THRU 5 AT 800 BPI.

T26. DATA TIME AT 1600 BPI (PE):  
REPEAT STEPS 1 THRU 5 AT 1600 BPI.  
\*\*THIS TEST IS NOT EXECUTED IF NRZ ONLY\*\*

926  
 927  
 928  
 929  
 930  
 931  
 932  
 933  
 934  
 935  
 936  
 937  
 938  
 939  
 940  
 941  
 942  
 943  
 944  
 945  
 946  
 947  
 948  
 949  
 950  
 951  
 952  
 953  
 954  
 955  
 956  
 957

TMO2 DRIVE FUNCTION TIMER

PAGE 21

T27. ERASE:

THE ERASE COMMAND WILL CAUSE AN AREA OF THE THREE (3) INCHES TO BE DC ERASED IN THE FORWARD DIRECTION. THIS TEST WILL ASSURE THAT THE PROPER DISTANCE IS ERASED.

1. LEAVE TAPE AT ITS PRESENT POSITION.
2. ISSUE AN ERASE COMMAND.
3. MONITOR DRY (BIT 7 OF DS)
4. THE TIME FROM GO TO DRY WILL BE THE TIME REQUIRED TO ERASE 3 INCHES OF TAPE AND WILL REFLECT THE DISTANCE. DENSITY IS NOT A FACTOR.
5. STOP

T30. TAPE MARK:

THIS TEST IS ALSO A CHECK ON THE THREE (3) INCH GAP. WHEN A TAPE MARK IS WRITTEN, A 3 INCH GAP IS CREATED BEFORE DATA IS PUT ON TAPE.

1. LEAVE TAPE AT ITS PRESENT POSITION
2. ISSUE A WRITE TAPE MARK COMMAND
3. MONITOR DRY (BIT 7 OF DS)
4. THE TIME FROM GO TO DRY WILL BE THE TIME REQUIRED TO WRITE THE TM RECORD PLUS THE 3 INCH GAP.
5. STOP

## TMO2 DRIVE FUNCTION TIMER

PAGE 22

958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989

## T31. TAPE SPEED FORWARD:

THIS TEST REQUIRES THE USE OF AN 800 BPI SKEW TAPE!  
THE OPERATOR WILL BE REQUIRED TO MOUNT THE SKEW TAPE  
BEFORE EXECUTING THE TEST. THE SKEW TAPE IS THE ONLY  
WAY TO ASSURE THAT TAPE IS MOVING AT THE PROPER SPEED  
BECAUSE THE FREQUENCY OF FRAMES ON A SKEW TAPE IS  
GUARANTEED TO BE ACCURATE.

1. ASSURE TAPE IS STOPPED AT BOT.
2. ISSUE A READ FORWARD (800 BPI, NORMAL)
3. MONITOR FC FOR FC = 800(10)
4. MONITOR FC FOR FC = 8800(10)
5. TIME FROM FC = 800 TO FC = 8800 IS THE TIME REQUIRED  
FOR TAPE TO TRAVEL 10 INCHES
6. DIVIDE THE TIME FOR 10 INCHES BY 10.
7. THE RESULT IS AN AVERAGE SPEED FOR 1 INCH.
8. STOP.

## T32. TAPE SPEED REVERSE:

THIS TEST IS THE SAME AS TEST 31, BUT SPEED IS  
MEASURED IN THE REVERSE DIRECTION.

1. ADVANCE TAPE OFF OF BOT.
2. ISSUE A READ REVERSE.
3. REPEAT STEPS 3 THRU 6 IN THE REVERSE DIRECTION.
4. STOP.

%

```

990
991          .NLIST MC
992          .LIST ME
993          .ABS
994          .MCALL SCPVEC,SCPREG,SCATCH,STYPE
995          .TITLE DZTUG-B TMO2/TE16 DRIVE FUNCTION TIMER
996          .SBTTL STARTING INSTRUCTIONS
997          ;LOADING AND STARTING PROCEEDURE
998          LOAD PROGRAM USING ABS LOADER
999          LOAD ADDRESS 200
1000         SET SWITCH OPTIONS
1001         PRESS START
1002
1003         ;GENERAL REGISTER USAGE:
1004         R0=ADDRESS OF 'FC' REGISTER (SET BY SCOPE)
1005         R1=ADDRESS OF 'DS' REGISTER (SET BY SCOPE)
1006         R2=RETURN PC FROM TIMER (SET BY EACH TEST)
1007         R3=INDEX INDICATING PREVIOUS OSCILLATOR POLARITY (SET BY TIMER)
1008         R4=CONTAINS 'TICK' COUNT WHEN TIMER IS RUNNING (SET BY TIMER)
1009         R5=ADDRESS OF CS1 (SET BY SCOPE)
1010
1011         ;SWITCH REGISTER SWITCH ASSIGNMENTS
1012         SW15= 100000          ;HALT ON ERROR
1013         SW14= 040000          ;LOOP SUBTEST
1014         SW13= 020000          ;INHIBIT ERROR TYPE OUT
1015         SW11= 004000          ;INHIBIT SUBTEST ITERATION
1016         SW10= 002000          ;INHIBIT PUBLICATION OF FUNCTION TIMES
1017         SW09= 001000          ;RING BELL ON ERROR
1018         SW08= 000400          ;TYPE LINE ITEM AFTER EACH ITERATION
1019         SW07= 000200          ;HALT ON TEST SELECTED IN SW05-SW00
1020         SW06= 000100          ;CONTINUOUS CYCLE
1021
1022         .SBTTL MACRO DEFINITIONS
1023         .MACRO SAVE
1024         JSR PC,SAVE          ;SAVE REGISTERS ON THE STACK
1025         .ENDM SAVE
1026         .MACRO RESTORE
1027         JSR PC,RESTORE      ;RESTORE REGISTERS FROM THE STACK
1028         .ENDM RESTORE
1029         .MACRO INPUT
1030         JSR PC,INPUT        ;GET USER INPUT
1031         .ENDM INPUT
1032         .MACRO REWIND
1033         JSR PC,REWIND       ;REWIND SLAVE
1034         BVS 99$             ;BRANCH IF ERROR ON REWIND
1035         .ENDM REWIND
1036         .MACRO TIMEON
1037         JSR PC,TIMON        ;TURN TIMER ON
1038         .ENDM TIMEON
1039         .MACRO TIMCHK
1040         JMP TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
1041         .ENDM TIMCHK
1042         .MACRO SETGO
1043         INC (R5)            ;SET 'GO' BIT
1044         .ENDM SETGO
1045

```

1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101

000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007  
000000  
000001  
000002  
000003  
000004  
000005  
  
177776  
177774  
177772  
177770  
177560  
177562  
177564  
177566  
  
000004  
000010  
000014  
000014  
000014  
000014  
000020  
000024  
000030  
000034  
000060  
000064  
000114  
000240  
000244  
000250  
  
172540  
000104  
177546  
000100  
177514  
177516  
  
172440

.SBTTL REGISTER ASSIGNMENTS  
;;DEFINITIONS AND REGISTER ASSIGNMENTS  
;;GENERAL REGISTER ASSIGNMENTS

R0=%0  
R1=%1  
R2=%2  
R3=%3  
R4=%4  
R5=%5  
SP=%6  
PC=%7  
R10=%0  
R11=%1  
R12=%2  
R13=%3  
R14=%4  
R15=%5

;;REGISTER ADDRESSES

PSW= 177776  
SLR= 177774  
PIRQ= 177772  
UBREAK= 177770  
TKS= 177560  
TKB= 177562  
TPS= 177564  
TPB= 177566

;;PROCESSOR STATUS WORD  
;;STACK LIMIT REGISTER (11/40,11/45)  
;;PROGRAM INTERRUPT REQ. (11/45)  
;;MICRO-BREAK REGISTER (11/45)  
;;KEYBOARD CSR  
;;KEYBOARD DATA BUFFER REGISTER  
;;TELEPRINTER CSR  
;;TELEPRINTER DATA BUFFER REGISTER

;;VECTOR ADDRESSES

ERRVEC=4  
RESVEC=10  
TBITVEC=14  
TRTVEC=14  
BPTVEC=14  
IOTVEC=20  
PFVEC=24  
EMTVEC=30  
TRAPVEC=34  
TKVEC= 60  
TPVEC=64  
PARVEC= 114  
PIRVEC=240  
FPEVEC=244  
MMVEC=250

;;ADDRESS OF ERROR VECTOR  
;;ADDRESS OF RESERVED INST. TRAP VECTOR  
;;ADDRESS OF 'T' BIT TRAP VECTOR  
;;ADDRESS OF 'TRACE' TRAP VECTOR  
;;ADDRESS OF 'BREAKPOINT' TRAP VECTOR  
;;ADDRESS OF IOT TRAP VECTOR  
;;ADDRESS OF POWER FAIL TRAP VECTOR  
;;ADDRESS OF EMT VECTOR  
;;ADDRESS OF TRAP VECTOR  
;;ADDRESS OF TTY KEYBOARD INT. VECTOR  
;;ADDRESS OF TTY PRINTER INTERRUPT VECTOR  
;;ADDRESS OF MA/MF PARITY ERROR VECTOR  
;;ADDRESS OF PIRQ VECTOR  
;;ADDRESS OF FLOATING POINT INT. VECTOR  
;;ADDRESS OF MEM MGMT ERROR TRAP VECTOR

;;CLOCK ADDRESS AND VECTORS

PLKCSR= 172540  
PLKVEC= 104  
LKS= 177546  
LKVEC= 100  
LPS= 177514  
LPB= 177516

;;KW11-P  
;;KW11-L  
;;LP11

;;RH11, TM02/TE16 REGISTERS

TMCS1= 172440

1102  
1103 000000  
1104 000002  
1105 000004  
1106 000006  
1107 000010  
1108 000012  
1109 000014  
1110 000016  
1111 000022  
1112 000024  
1113 000026  
1114 000030  
1115 000032  
1116  
1117  
1118  
1119 000001  
1120 000000  
1121 000002  
1122 000006  
1123 000010  
1124 000026  
1125 000024  
1126 000030  
1127 000032  
1128 000050  
1129 000056  
1130 000060  
1131 000070  
1132 000076  
1133 000100  
1134 000200  
1135 000400  
1136 001000  
1137 002000  
1138 004000  
1139 020000  
1140 040000  
1141 100000  
1142  
1143 000000  
1144 000001  
1145 000002  
1146 000003  
1147 000004  
1148 000005  
1149 000006  
1150 000007  
1151 000010  
1152 000020  
1153 000040  
1154 000100  
1155 000200  
1156 000400  
1157 001000

;TM02/TE16 INDEX VALUES

CS1= 00  
WC= 02  
BA= 04  
FC= 06  
CS2= 10  
DS= 12  
ER= 14  
AS= 16  
DB= 22  
MR= 24  
DT= 26  
SN= 30  
TC= 32

;CONTROL STATUS #1  
;BUS ADDRESS REGISTER  
;FRAME COUNT  
;CONTROL STATUS #2  
;DRIVE STATUS  
;ERROR REG #1  
;ATTENTION SUMMARY  
;DATA BUFFER REG  
;MAINTENANCE REG  
;DRIVE TYPE REG  
;SERIAL NUMBER REGISTER  
;TAPE CONTROL REG

.SBTTL TM02/TE16 REGISTER BITS  
;RHCS1-CS1(R5)

GO= 1  
NOP= 0  
RWDIFF= 2  
RWD= 6  
DRYCLR= 10  
WFMK= 26  
ERASE= 24  
SPCFWD= 30  
SPCREV= 32  
WCHKF= 50  
WCHKR= 56  
WFWD= 60  
RDFWD= 70  
RDREV= 76  
IE= 100  
RDY= 200  
A16= 400  
A17= 1000  
PSEL= 2000  
DVA= 4000  
MCPE= 20000  
TRE= 40000  
SC= 100000

;RHCS2-CS2(R5)

DV0= 0  
DV1= 1  
DV2= 2  
DV3= 3  
DV4= 4  
DV5= 5  
DV6= 6  
DV7= 7  
BAI= 10  
PAT= 20  
CLR= 40  
IR= 100  
OR= 200  
MDPE= 400  
MXF= 1000

1158 002000  
1159 004000  
1160 010000  
1161 020000  
1162 040000  
1163 100000  
1164  
1165 000001  
1166 000002  
1167 000004  
1168 000010  
1169 000020  
1170 000040  
1171 000100  
1172 000200  
1173 000400  
1174 002000  
1175 004000  
1176 010000  
1177 020000  
1178 040000  
1179 100000  
1180  
1181 000001  
1182 000002  
1183 000004  
1184  
1185 000020  
1186 000100  
1187 000200  
1188 000400  
1189 001000  
1190 002000  
1191 004000  
1192 010000  
1193 020000  
1194 040000  
1195  
1196  
1197 000100  
1198  
1199  
1200 002000  
1201 010000  
1202 040000  
1203  
1204  
1205 000300  
1206 000320  
1207 000000  
1208 000400  
1209 001000  
1210 002000  
1211 100000  
1212  
1213

PGE= 2000  
NEH= 4000  
NED= 10000  
UPE= 20000  
MCE= 40000  
DLT= 100000  
;RHDS-DS(R5)  
SLA= 1  
BOT= 2  
TMK= 4  
IDB= 10  
SDWN= 20  
PES= 40  
SSC= 100  
DRY= 200  
DPR= 400  
EOT= 2000  
MRL= 4000  
MOL= 10000  
PIP= 20000  
ERR= 40000  
ATA= 100000  
;RHER-ER(R5)  
ILF= 1  
ILR= 2  
RMR= 4  
FMT= 20  
INCVAE= 100  
PEFLRC= 200  
NSG= 400  
FCE= 1000  
CSITM= 2000  
NEF= 4000  
DTE= 10000  
OPT= 20000  
UNS= 40000  
;RHMR-MR(R5)  
OSC= 100  
;RHDT-DT(R5)  
SPR= 2000  
CH7= 10000  
TAP= 40000  
;RHTC-TC(R5)  
NORM11= 300  
CDM11= 320  
BPI200= 0  
BPI556= 000400  
BPI800= 001000  
PE1600= 002000  
ACCL= 100000  
;INSTRUCTION EQUATES



1214	104400
1215	104000
1216	000004
1217	
1218	
1219	005724
1220	177400
1221	177600
1222	
1223	000003
1224	000011
1225	000012
1226	000015
1227	000017
1228	000025

HLT= TRAP  
SCOPE= EMT  
TYPE= IOT

; MISCELLANEOUS EQUATES

OUTBUF=INIT  
FRMCNT= -256.  
WRDCNT= -128.

; ASCII EQUATES

CNTRLC= 3  
HT= 11  
LF= 12  
CR= 15  
CNTRLO= 17  
CNTRLU= 25

; OUTPUT BUFFER START AT BEG OF PROGRAM  
; FRAME COUNT  
; WORD COUNT

; ASCII CODE FOR CONTROL C (↑C)  
; ASCII CODE FOR HORIZONTAL TAB  
; ASCII CODE FOR LINE FEED  
; ASCII CODE FOR CARRIAGE RETURN  
; ASCII CODE FOR CONTROL O (↑O)  
; ASCII CODE FOR CONTROL U (↑U)

```

1229          ;SETUP TRAP VECTORS
1230          . =TBITVEC
1231 000014   .WORD .+2          ;SET 'T' TRAP TO TIMER ROUTINE
1232 000016   .WORD HALT        ;PRIORITY LEVEL 7
1233 000020   .WORD .TYPE      ;SET IOT TRAP TO .TYPE ROUTINE
1234 000022   .WORD 0          ;PRIORITY LEVEL 0
1235 000024   .WORD PFVEC+2    ;POWER FAIL TRAP TO HALT
1236 000026   .WORD HALT      ;AT PFVEC+2
1237 000030   .WORD .SCOPE     ;SET ENT TRAP TO .SCOPE ROUTINE
1238 000032   .WORD 340       ;PRIORITY LEVEL 7
1239 000034   .WORD .HLT      ;SET TRAP TRAP TO .HLT ROUTINE
1240 000036   .WORD 340       ;PRIORITY LEVEL 7
1241          . =TKVEC
1242 000060   .WORD TKISR
1243 000062   .WORD 340

1244          ;SOFTWARE SWITCH REGISTER LOC. 176
1245          . =176
1246          SWREG: 0          ;SOFTWARE SWITCH REGISTER
1247 000176   000000
1248
1249          . =200
1250 000200   000137 005724   JMP @#INIT          ;GO TO START OF PROGRAM
1251
1252          . =500
1253          STKPTR= 600      ;STACK
1254
1255          . =1000
1256          ;PROGRAM TAGS
1257 001000   177570   SWR: 177570          ;SWITCH REGISTER
1258 001002   000000   SCPADR: .WORD 0
1259 001004   000     DRVNUM: .BYTE 0
1260 001005   000     SLVNUM: .BYTE 0
1261 001006   000000   SLVPTR: .WORD 0
1262 001010   172440   TMBASE: .WORD TMC51
1263 001012   000000   ATIME: .WORD 0
1264 001014   000020   ATIMTBL: .BLKW 16.
1265
1266 001054   000020   GAP: .BLKW 16.
1267 001114   000000   DELTIM: .WORD 0
1268 001116   000000   OCTALD: .WORD 0
1269 001120   000     GAP: .BYTE 0
1270 001121   000     ITCNT: .BYTE 0
1271 001122   000     TSTNUM: .BYTE 0
1272 001123   000     ERFLG: .BYTE 0
1273 001124   000     PRGFLG: .BYTE 0
1274 001125   000     UNTFND: .BYTE 0
1275 001126   000     TYPFLG: .BYTE 0
1276 001127   000     NRZFLG: .BYTE 0
1277 001130   000     ASFLG: .BYTE 0
1278          . EVEN
1279 001132   030460   DIGTAB: "01
1280 001134   031462   "23
1281 001136   032464   "45
1282 001140   033466   "67
1283 001142   034470   "89
1284 001144   000006   ODIGITS: .BLKB 6

```

```

;SET 'T' TRAP TO TIMER ROUTINE
;PRIORITY LEVEL 7
;SET IOT TRAP TO .TYPE ROUTINE
;PRIORITY LEVEL 0
;POWER FAIL TRAP TO HALT
;AT PFVEC+2
;SET ENT TRAP TO .SCOPE ROUTINE
;PRIORITY LEVEL 7
;SET TRAP TRAP TO .HLT ROUTINE
;PRIORITY LEVEL 7

```

```

;SOFTWARE SWITCH REGISTER
;SOFTWARE SWITCH REGISTER
;GO TO START OF PROGRAM

```

```

;STACK
;SWITCH REGISTER
;TMO2 DRIVE UNDER TEST
;TE16 SLAVE UNDER TEST
;POINTER TO SLAVE TABLE (SLVTBL) BELOW
;BASE ADDRESS OF TMO2/TE16 REGISTERS
;CONTAINS 'TICK' COUNT
;EACH ENTRY CONTAINS TIME FOR FUNCTION
;ENTRIES ARE MADE BY 'SCOPE' ROUTINE
;TIMES RECORDED BY 'GAP CONSISTANCY' TEST
;VARIABLE DELAY
;CONTAINS GAP # (USED FOR TST 021)
;ITERATION COUNT
;TEST #
;ERROR FLAG
;PROGRAM FLAG
;UNIT FOUND INDICATOR
;INDICATES IF DRIVE IS NRZ ONLY.
;1/0 = YES/NO.

```

```

;RESERVE SPACE FOR CONVERTED DIGITS

```

DZTUG-B TMO2/TE16 DRIVE FUNCTION TIMER  
DZTUGB.P11 15-JUL-77 12:52

MACY11 30(1046) 15-JUL-77 12:55 PAGE 33  
TMO2/TE16 REGISTER BITS

SEQ 0033

1285	001152	000	
1286		001154	
1287	001154	000010	
1288	001164	000100	
1289	001264	000110	
1290	001374	005015	000
1291	001377	134	000
1292	001401	060	000
1293	001403	007	000
1294	001405	055	000
1295	001407	040	
1296	001410	000040	
1297	001412	004476	000
1298		001416	

	.BYTE	0
	.EVEN	
DRVTBL:	.BLKB	8.
SLVTBL:	.BLKB	64.
INBUF:	.BLKB	72.
CRLF:	.ASCIZ	<CR><LF>
BKSLSH:	.ASCIZ	'\'
ECHO:	.ASCIZ	'0'
BELL:	.ASCIZ	<?>
DASH:	.ASCIZ	'-'
SPACE2:	.ASCIZ	' '
SPACE:	.ASCIZ	' '
ANGTAB:	.ASCIZ	'>'<HT>
	.EVEN	

; TERMINATOR  
 ; A 0/-1 = DRIVE NOT TO BE/TO BE TESTED  
 ; A 0/-1 = SLAVE NOT TO BE/TO BE TESTED  
 ; TELETYPE INPUT BUFFER  
 ; MISCELLANEOUS ASCII CHARACTERS

.SBTTL TIME SPECIFICATION TABLE

:THE BELOW TABLE CONTAINS THE SPECIFIED FUNCTION TIMES IN TENS OF  
:MICROSECONDS. NOTE THAT WHEN TIMES ARE TYPED THAT THEY ARE TYPED IN  
:MICROSECONDS (BY APPENDING A 0).  
:FORMAT IS

.WORD	MAX,MIN	;TIME IN MS	FUNCTION	TEST #
STIMTBL: .WORD	0,0	;SPARE		
.WORD	15400.,15000.	;154.0-150.0	WRITE FROM BOT	TST001
.WORD	00950.,00870.	;9.5-8.7	WRITE START	TST002
.WORD	00890.,00850.	;8.9-8.5	WRITE SHUTDOWN	TST003
.WORD	01350.,00810.	;13.5-8.1	WRITE STLDOWN	TST004
.WORD	03700.,03300.	;37.0-33.0	READ FROM BOT	TST005
.WORD	00320.,00260.	;3.2-2.6	READ START	TST006
.WORD	00465.,00425.	;4.65-4.25	READ SHUTDOWN	TST007
.WORD	01350.,00810.	;13.5-8.1	READ SETTLEDOWN	TST010
.WORD	00320.,00260.	;3.2-2.6	RD REV START	TST011
.WORD	00370.,00330.	;3.7-3.3	RD REV SHTDWN	TST012
.WORD	01350.,00810.	;13.5-8.1	RD REV STLDWN	TST013
.WORD	01670.,01070.	;16.7-10.7	TRN RND DLY F-R	TST014
.WORD	01670.,01070.	;16.7-10.7	TRN RND DLY R-F	TST015
.WORD	01290.,00950.	;12.9-9.5	GAP SIZE STOP	TST016
.WORD	01180.,00850.	;11.8-8.5	GAP SIZE STRT	TST017
.WORD	01480.,01370.	;14.8-13.7	GAP SIZE INTER	TST020
.WORD	01380.,01240.	;13.8-12.4	GAP CONSISANCY	TST021
.WORD	0,0	;0.0-0.0	DUMMY	TST022
.WORD	02410.,02310.	;24.1-23.1	DAT TIME 200BPI	TST023
.WORD	02400.,02300.	;24.0-23.0	DAT TIME 556BPI	TST024
.WORD	02400.,02300.	;24.0-23.0	DAT TIME 800BPI	TST025
.WORD	02510.,02410.	;25.1-24.1	DAT TIME 1600PE	TST026
.WORD	10100.,09900.	;101.0-99.0	ERASE	TST027
.WORD	10500.,10300.	;105.0-103.0	WRT FILE MARK	TST030
.WORD	02270.,02170.	;22.7-21.7	READ 1" TAPE	TST031
.WORD	02270.,02170.	;22.7-21.7	RD REV 1" TAPE	TST032

;NOTE: TEST 31 AND 32 REQUIRE PRERECORDED 800BPI SKEW TAPE.

1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306 001416 000000 000000  
1307 001422 036050 035230  
1308 001426 001666 001546  
1309 001432 001572 001522  
1310 001436 002506 001452  
1311 001442 007164 006344  
1312 001446 000500 000404  
1313 001452 000721 000651  
1314 001456 002506 001452  
1315 001462 000500 000404  
1316 001466 000562 000512  
1317 001472 002506 001452  
1318 001476 003206 002056  
1319 001502 003206 002056  
1320 001506 002412 001666  
1321 001512 002234 001522  
1322 001516 002710 002532  
1323 001522 002544 002330  
1324 001526 000000 000000  
1325 001532 004552 004406  
1326 001536 004540 004374  
1327 001542 004540 004374  
1328 001546 004716 004552  
1329 001552 023564 023254  
1330 001556 024404 024074  
1331 001562 004336 004172  
1332 001566 004336 004172  
1333  
1334

1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342 001572 002602 002412  
1343 001576 002652 002506  
1344 001602 002734 002532  
1345 001606 002734 002532  
1346 001612 002734 002424  
1347 001616 002652 002260  
1348 001622 002652 002260  
1349 001626 002652 002260  
1350 001632 002532 002260  
1351 001636 002532 002260  
1352 001642 002532 002260  
1353 001646 002532 002260  
1354 001652 002532 002260  
1355 001656 002532 002260  
1356 001662 002532 002260  
1357 001666 002532 002260

.SBTTL GAP TIME SPECIFICATION TABLE  
;THIS TABLE CONTAINS THE GAP SIZES (IN TENS OF MICROSECONDS) FOR EACH  
;OF THE 16. GAPS RECORDED BY THE GAP CONSISTANCY TEST (TST021).  
;NOTE: GAP #'S ARE IN OCTAL.

;	.WORD	MAX,MIN(10)	;TIME IN MS(10)	GAP #	DELAY IN MS(10)
GTIMTBL:	.WORD	01410.,01290.	:14.1-12.9	GAP-0	0 MS
	.WORD	01450.,01350.	:14.5-13.5	GAP-1	1.0 MS
	.WORD	01500.,01370.	:15.0-13.7	GAP-2	2.0 MS
	.WORD	01500.,01370.	:15.0-13.7	GAP-3	3.0 MS
	.WORD	01500.,01300.	:15.0-13.0	GAP-4	4.0 MS
	.WORD	01450.,01200.	:14.5-12.0	GAP-5	5.0 MS
	.WORD	01450.,01200.	:14.5-12.0	GAP-6	6.0 MS
	.WORD	01450.,01200.	:14.5-12.0	GAP-7	7.0 MS
	.WORD	01370.,01200.	:13.7-12.0	GAP-10	8.0 MS
	.WORD	01370.,01200.	:13.7-12.0	GAP-11	9.0 MS
	.WORD	01370.,01200.	:13.7-12.0	GAP-12	10.0 MS
	.WORD	01370.,01200.	:13.7-12.0	GAP-13	11.0 MS
	.WORD	01370.,01200.	:13.7-12.0	GAP-14	12.0 MS
	.WORD	01370.,01200.	:13.7-12.0	GAP-15	13.1 MS
	.WORD	01370.,01200.	:13.7-12.0	GAP-16	14.1 MS
	.WORD	01370.,01200.	:13.7-12.0	GAP-17	15.1 MS

1358  
 1359  
 1360 001672 000000  
 1361 001674 014665  
 1362 001676 014707  
 1363 001700 014727  
 1364 001702 014751  
 1365 001704 014775  
 1366 001706 015017  
 1367 001710 015036  
 1368 001712 015060  
 1369 001714 015103  
 1370 001716 015125  
 1371 001720 015152  
 1372 001722 015201  
 1373 001724 015232  
 1374 001726 015263  
 1375 001730 015311  
 1376 001732 015340  
 1377 001734 015370  
 1378 001736 000000  
 1379 001740 015413  
 1380 001742 015437  
 1381 001744 015463  
 1382 001746 015507  
 1383 001750 015534  
 1384 001752 015556  
 1385 001754 015601  
 1386 001756 015623

.SBTTL TEST HEADER POINTERS  
 ; THE BELOW TABLE CONTAINS POINTERS TO EACH TEST'S DESCRIPTOR  
 NAMPTR: .WORD 0

.WORD A.T001  
 .WORD A.T002  
 .WORD A.T003  
 .WORD A.T004  
 .WORD A.T005  
 .WORD A.T006  
 .WORD A.T007  
 .WORD A.T010  
 .WORD A.T011  
 .WORD A.T012  
 .WORD A.T013  
 .WORD A.T014  
 .WORD A.T015  
 .WORD A.T016  
 .WORD A.T017  
 .WORD A.T020  
 .WORD A.T021  
 .WORD 0  
 .WORD A.T023  
 .WORD A.T024  
 .WORD A.T025  
 .WORD A.T026  
 .WORD A.T027  
 .WORD A.T030  
 .WORD A.T031  
 .WORD A.T032

; DUMMY TEST

```

1387
1388
1389
1390
1391
1392 001760 000000
1393 001762 000000
1394 001764 000000
1395 001766 000000
1396
1397 001770 022767 000176 177002 CKSWR: CMP #SWREG,SWR ;SOFTWARE SWITCH REG PRESENT
1398 001776 001120 BNE OUT ;NO, GET OUT
1399 002000 016767 175556 177752 MOV TKB,TIB ;AND STRIP OFF
1400 002006 042767 177600 177744 BIC #177600,TIB ;THE GARBAGE
1401 002014 022767 000007 177736 CMP #7,TIB ;IS IT A '<IG>'
1402 002022 001106 BNE OUT
1403 002024 000004 015645 TYPE,L.CNTG
1404 002030 000004 015652 CNTLU: TYPE,L.SWR
1405 002034 017702 176740 MOV #SWR,R2
1406 002040 004767 000564 JSR PC,TYPECT
1407 002044 000004 015661 TYPE,L.NEW
1408
1409 002050 005067 177706 CLR TEMPST
1410 002054 012767 000007 177702 MOV #7,COUNT
1411 002062 004767 000154 1S: JSR PC,TTIN ;GO READ A CHARACTER
1412 002066 042767 177600 177664 BIC #177600,TIB ;STRIP OFF GARBAGE
1413 002074 122767 000025 177656 CMPB #25,TIB ;IS IT A 'U'?
1414 002102 001001 BNE 2S ;BRANCH IF NOT
1415 002104 000751 3S: BR CNTLU ;START OVER
1416 002106 122767 000015 177644 2S: CMPB #15,TIB ;IS IT A '<CR>?'
1417 002114 001012 BNE 4S ;BRANCH IF NOT
1418 002116 012767 000200 177642 MOV #200,RDSW
1419 002124 004767 000230 JSR PC,TCRLF ;ECHO IT WITH '<LF>'
1420 002130 022767 000007 177626 CMP #7,COUNT ;WAS IT FIRST CHARACTER
1421 002136 001034 BNE 7S ;CHANGE SWR IF NOT FIRST ONE
1422 002140 000437 8S: BR OUT ;GET OUT
1423 002142 122767 000060 177610 4S: CMPB #60,TIB
1424 002150 003004 BGT 5S
1425 002152 122767 000067 177600 CMPB #67,TIB
1426 002160 002003 BGE 6S
1427 002162 000004 015671 5S: TYPE,L.QUEST
1428 002166 000746 BR 3S ;START OVER IF NOT LEGAL CHARACTER
1429 002170 006367 177566 6S: ASL TEMPST
1430 002174 006367 177562 ASL TEMPST
1431 002200 006367 177556 ASL TEMPST
1432 002204 142767 000060 177546 BICB #60,TIB ;GET NITTY-GRITTY
1433 002212 156767 177542 177542 BISB TIB,TEMPST
1434 002220 005367 177540 DEC COUNT ;ONLY WANT 6 DIGITS
1435 002224 001756 BEQ 5S
1436 002226 000715 BR 1S
1437 002230 016777 177526 176542 7S: MOV TEMPST,#SWR ;CHANGE SWITCH REGISTER CONTENTS
1438 002236 000740 BR 8S
1439 002240 000207 OUT: RTS PC
1440

```

```

1441
1442
1443
1444 002242 005067 175312
1445 002246 005067 175310
1446 002252 005067 177502
1447 002256 005267 175276
1448 002262 105767 175272
1449 002266 100375
1450 002270 016767 175266 177462
1451 002276 105767 175262
1452 002302 100375
1453 002304 116767 177450 175254
1454 002312 000207
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464 000011
1465 002314 000
1466 002315 002
1467 002316 000
1468
1469 002317 000
1470 002320 177564
1471 002322 177566
1472 002324 000
1473 002325 000
1474 002326 005015 000
1475 002332
1476
1477 002332 010046
1478 002334 017600 000002
1479 002340 062766 000002 000002
1480 002346 105067 177753
1481
1482 002352 105767 177747
1483 002356 001410
1484 002360 000004 002326
1485 002364 105767 177376
1486 002370 100006
1487 002372 005067 177370
1488 002376 000207
1489 002400 112046
1490 002402 001003
1491 002404 005726
1492 002406 012600
1493 002410 000000
1494
1495 002412 122716 000011
1496 002416 001445

```

;TTY READ SUBROUTINE\*\*\*\*\*

```

TTIN: CLR TKS
      CLR TKB
      CLR TIB
      INC TKS
TTIN1: TSTB TKS
      BPL TTIN1
      MOV TKB,TIB
TTIN2: TSTB TPS
      BPL TTIN2
      MOV TIB,TPB
      RTS PC

```

```

.SBTL PROGRAM SUBROUTINES
.SBTL TYPE SUBROUTINE
;;ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;;CALL: TYPE ;;A TRAP TYPE INSTRUCTION
;; MESADR ;;MESADR IS FIRST ADDRESS OF ASCIZ STRING

```

```

;;TAGS USED BY THE TYPE ROUTINE BELOW
SHT=11 ;;HORIZONTAL TAB
SNUL: .BYTE 0 ;;CONTAINS NULL CHARACTER
SFILL: .BYTE 2 ;;CONTAINS # OF FILLER CHARACTERS
STPFLG: .BYTE 0 ;;CONTAINS TELEPRINTER AVAILABLE FLAG
; ;0/377 = AVAIL/NOT AVAIL
STKFLG: .BYTE 0 ;;CONTAINS KEYBOARD AVAILABLE FLAG
STPS: .WORD 177564 ;;ADDRESS OF TELEPRINTER STATUS REGISTER
STPB: .WORD 177566 ;;ADDRESS OF TELEPRINTER DATA BUFFER
SCHARCNT: .BYTE 0 ;;CONTAINS # OF CHARS TYPED
SCNTRLO: .BYTE 0 ;;CONTAINS CONTROL 0 CHAR (IF TYPED)
SCRLF: .ASCIZ <15><12>
      .EVEN

```

```

.TYPE: MOV RO,-(SP) ;;SAVE RO
      MOV @2(SP),RO ;;GET MESSAGE ADDRESS
      ADD #2,2(SP) ;;ADJUST RETURN PC
      CLR SCNTRLO
TYPE1: TSTB SCNTRLO ;;BRANCH IF CONTROL 0(10) WASN'T TYPED
      BEQ TYPE2
TCRLF: TYPE,SCRLF ;;TYPE <CR><LF>
      TSTB RDSW
      BPL TYPE3
      CLR RDSW
      RTS PC
TYPE2: MOV (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
      BNE TYPE4 ;;BRANCH IF NOT THE TERMINATOR
      TST (SP)+ ;;POP TERMINATOR CHAR OFF THE STACK
      MOV (SP)+,RO ;;RESTORE RO
      RTI ;;RETURN TO CALLER
TYPE3:
TYPE4: CMPB #SHT,(SP) ;;BRANCH IF HORIZONTAL TAB <HT>
      BEQ 9S

```



```

1497 002420 004767 000026      JSR    PC,5$      ;; TYPE CHARACTER
1498 002424 122726 000012      3$:    CMPB    #12,(SP)+  ;; CHECK IF CHARACTER WAS A LINE FEED
1499 002430 001350              BNE     TYPE1     ;; BRANCH IF NOT LINE FEED
1500 002432 016746 177656      MOV     $NULL,-(SP) ;; GET # OF FILLERS REQUIRED AND FILLER
1501                                ;; CHARACTER.
1502
1503 002436 105366 000001      4$:    DECB    1(SP)  ;; DECREMENT FILLERS REQ. COUNT
1504 002442 002770              BLT     3$        ;; BRANCH IF NO MORE FILLERS ARE REQUIRED
1505 002444 004767 000002      JSR    PC,5$      ;; TYPE FILLER CHARACTER
1506 002450 000772              BR     4$
1507
1508 002452 105777 177642      5$:    TSTB    @STPS   ;; WAIT FOR OUTPUT DEVICE
1509 002456 100375              BPL    -4
1510 002460 122737 000017 002325  CMPB    #17,@$CNTRL0 ;; CHECK IF CONTROL 0 WAS TYPED
1511 002466 001403              BEQ    6$        ;; STOP TYPING MESSAGE IF 10 WAS TYPED
1512 002470 116677 000002 177624  MOVB    2(SP),@STPB  ;; OUTPUT CHARACTER
1513 002476 122766 000015 000002  6$:    CMPB    #15,2(SP) ;; BRANCH IF NOT <CR>
1514 002504 001003              BNE     7$
1515 002506 105067 177612      CLRB    $CHARCNT  ;; CLEAR CHARACTERS TYPED COUNT
1516 002512 000406              BR     8$
1517 002514 122766 000012 000002  7$:    CMPB    #12,2(SP) ;; BRANCH IF <LF> OR 'NULL'
1518 002522 002002              BGE    8$
1519 002524 105267 177574      INCB    $CHARCNT  ;; INCREMENT CHARACTER TYPED COUNT
1520 002530 000207      8$:    RTS     PC
1521
1522      ;; HORIZONTAL TAB <HT> PROCESSER
1523 002532 112716 000040      9$:    MOVB    #40,(SP) ;; LOAD 'SPACE'
1524 002536 004767 177710      10$:   JSR    PC,5$      ;; TYPE 'SPACE'
1525 002542 132767 000007 177554  BITB    #7,$CHARCNT ;; TYPE SPACES UNTIL A MULTIPLE
1526 002550 001372              BNE     10$      ;; OF 8 CHARACTERS HAVE BEEN TYPED
1527 002552 105726              TSTB    (SP)+     ;; POP SPACE
1528 002554 000676              BR     TYPE1     ;; GET NEXT CHARACTER
1529
1530      ; SUBROUTINE TO SAVE GENERAL REGISTERS ON THE STACK
1531      ; CALL: SAVE
1532      .SAVE: MOV     R5,-(SP)      ; SAVE REGISTERS ON THE STACK
1533           MOV     R4,-(SP)
1534           MOV     R3,-(SP)
1535           MOV     R2,-(SP)
1536           MOV     R1,-(SP)
1537           MOV     R0,-(SP)
1538 002572 016646 000014      MOV     14(SP),-(SP) ; GET RETURN PC
1539 002576 000207      RTS     PC          ; RETURN
1540
1541      ; SUBROUTINE TO RESTORE GENERAL REGISTERS FROM THE STACK
1542      ; CALL: RESTORE
1543 002600 012666 000014      .RESTORE:MOV    (SP)+,14(SP) ; MOVE RETURN PC
1544           MOV     (SP)+,R0    ; RESTORE REGISTERS
1545           MOV     (SP)+,R1
1546           MOV     (SP)+,R2
1547           MOV     (SP)+,R3
1548           MOV     (SP)+,R4
1549           MOV     (SP)+,R5
1550           RTS     PC          ; RETURN
1551
1552      ; SUBROUTINE TO CONVERT OCTAL DATA TO ASCII

```

```

1553 ;CALL: MOV NUMBER,R2 ;MOVE NUMBER TO R2
1554 ; JSR PC,CNVCT
1555 ;
1556 002622 110667 176300 CNVCT: MOVB SP,TYPFLG ;SET DO NOT TYPE FLAG
1557 002626 000402 BR CNVTO
1558 ;
1559 .SBTTL OCTAL TO ASCII & TYPE ROUTINE
1560 ;SUBROUTINE TO CONVERT OCTAL NUMBER TO ASCII AND TYPE IT OUT
1561 ;CALL: MOV NUMBER,R2 ;PUT # IN R2
1562 ; JSR PC,TYPOCT ;CALL ROUTINE
1563 ;
1564 002630 105037 001126 TYPOCT: CLRB @#TYPFLG ;SET TYPE FLAG
1565 002634 CNVTO:
1566 002634 004767 177716 JSR PC,.SAVE ;SAVE REGISTERS ON THE STACK
1567 002640 012704 001144 MOV #0DIGITS,R4 ;SET PTR TO OUTPUT
1568 002644 005003 CLR R3 ;R3 WILL CONTAIN OCTAL DIGIT
1569 002646 010201 MOV R2,R1 ;GET # TO BE TYPED
1570 002650 006302 1S: ASL R2 ;SHIFT #
1571 002652 006103 ROL R3
1572 002654 012700 000006 MOV #6,R0 ;SET DIGIT COUNTER
1573 002660 000404 BR 3S
1574 ;
1575 002662 006302 2S: ASL R2 ;SHIFT # 3 PLACES LEFT
1576 002664 006103 ROL R3
1577 002666 005301 DEC R1
1578 002670 001374 BNE 2S
1579 002672 012701 000003 3S: MOV #3,R1 ;SET SHIFT COUNTER
1580 002676 116324 001132 MOVB DIGTAB(R3),(R4)+ ;MOVE ASCII EQUIV TO OUTPUT
1581 002702 005003 CLR R3
1582 002704 005300 DEC R0 ;DECREMENT DIGIT COUNT
1583 002706 001365 BNE 2S ;GET NEXT DIGIT
1584 002710 105737 001126 TSTB @#TYPFLG ;BRANCH IF ASCII IS
1585 002714 001002 BNE 4S ;NOT TO BE TYPED
1586 002716 000004 001144 TYPE,0DIGITS
1587 002722 4S:
1588 002722 004767 177652 JSR PC,.RESTORE ;RESTORE REGISTERS FROM THE STACK
1589 002726 000207 RTS PC
1590 ;
1591 ;
1592 ;SUBROUTINE TO CONVERT OCTAL DATA TO DECIMAL ASCII
1593 ;CALL: MOV NUMBER,R2 ;MOVE NUMBER TO R2
1594 ; JSR PC,CNVDEC
1595 ;
1596 002730 110637 001126 CNVDEC: MOVB SP,@#TYPFLG ;SET DO NOT TYPE FLAG
1597 002734 000402 BR CNVTD
1598 ;
1599 .SBTTL OCTAL TO DECIMAL & TYPE ROUTINE
1600 ;THIS ROUTINE CONVERTS AN OCTAL # TO DECIMAL ASCII AND TYPES IT OUT
1601 ;CALL: MOV NUMBER,R2 ;PUT # IN R2
1602 ; JSR PC,TYPDEC ;CALL ROUTINE
1603 002736 105037 001126 TYPDEC: CLRB @#TYPFLG ;SET TYPE FLAG
1604 002742 CNVTD:
1605 002742 004767 177610 JSR PC,.SAVE ;SAVE REGISTERS ON THE STACK
1606 002746 005000 CLR R0 ;R0 IS INDEX TO DECIMAL CONSTANT
1607 002750 012704 001144 MOV #0DIGITS,R4 ;SET OUTPUT PTR
1608 002754 005003 1S: CLR R3 ;R3 CONTAINS DECIMAL DIGIT

```

1609 002756 166002 003036  
 1610 002762 103402  
 1611 002764 005203  
 1612 002766 000773  
 1613 002770 066002 003036  
 1614 002774 116324 001132  
 1615 003000 062700 000002  
 1616 003004 005760 003036  
 1617 003010 001361  
 1618 003012 112724 000060  
 1619 003016 105737 001126  
 1620 003022 001002  
 1621 003024 000004 001144  
 1622 003030  
 1623 003030 004767 177544  
 1624 003034 000207  
 1625  
 1626 003036 023420  
 1627 003040 001750  
 1628 003042 000144  
 1629 003044 000012  
 1630 003046 000001  
 1631 003050 000000  
 1632  
 1633  
 1634  
 1635  
 1636  
 1637  
 1638  
 1639  
 1640  
 1641  
 1642  
 1643  
 1644 003052 010246  
 1645 003054 010346  
 1646 003056 006302  
 1647 003060 006302  
 1648 003062 010203  
 1649 003064 000004 014645  
 1650 003070 016302 001416  
 1651 003074 004767 177636  
 1652 003100 000004 001405  
 1653 003104 016302 001420  
 1654 003110 004767 177622  
 1655 003114 000004 001412  
 1656 003120 000004 014655  
 1657 003124 013702 001012  
 1658 003130 004767 177602  
 1659 003134 000004 001374  
 1660 003140 012603  
 1661 003142 012602  
 1662 003144 000207  
 1663  
 1664

```

25:  SUB    DCONST(R0),R2      ;SUBTRACT DECIMAL CONSTANT UNTIL
    BLO    3$                ;INPUT # GOES NEGATIVE
    INC    R3                ;KEEPING TRACK OF SUBTRACTIONS
    BR     2$
3$:  ADD    DCONST(R0),R2      ;ADD BACK CONSTANT WHEN NEGATIVE
    MOVB   DIGTAB(R3),(R4)+   ;MOVE ASCII EQUIVALENT
    ADD    #2,R0              ;NEXT CONSTANT
    TST    DCONST(R0)        ;UNTIL ALL CONSTANTS DONE
    BNE    1$
    MOVB   #'0,(R4)+         ;LAST DIGIT IS 0
    TSTB   @#TYPFLG          ;BRANCH IF ASCII IS
    BNE    4$                ;NOT TO BE TYPED
    TYPE,ODIGITS
4$:  JSR    PC,.RESTORE       ;RESTORE REGISTERS FROM THE STACK
    RTS    PC
DCONST: .WORD 10000.
        .WORD 1000.
        .WORD 100.
        .WORD 10.
        .WORD 1.
        .WORD 0              ;TERMINATOR
        .SBTTL               TYPE SPECIFIED TIMES ROUTINE
;THIS SUBROUTINE OUTPUTS THE TIME SPECIFICATIONS FOR THE TEST
;AND ALSO THE ACTUAL TIME RECORDED (ATIME)
;FORMAT OF LINE TYPED
;RANGE=<AAAAAA-BBBBBB>          ACTUAL=CCCCCC
;WHERE:
;AAAAAA IS MAXIMUM TIME FOR TEST (STIMTBL(TSTNUMX4)).
;BBBBBB IS MINIMUM TIME FOR TEST (STIMTBL(TSTNUMX4+2)).
;CCCCCC IS ACTUAL TIME RECORDED BY TEST (ATIME).
;CALL:  MOVB   TEST NUMBER,R2 ;LOAD TEST NUMBER
;        MOV    #TIME,@#ATIME ;MOVE TIME TO ATIME
;        JSR    PC,OUTSPC
;OUTSPC: MOV    R2,-(SP)        ;SAVE R2 & R3 ON THE STACK
;        MOV    R3,-(SP)
;        ASL   R2              ;MULTIPLY TEST # TIMES 4
;        ASL   R2              ;TO FORM INDEX INTO STIMTBL
;        MOV   R2,R3          ;R3 CONTAINS INDEX INTO TABLE
;        TYPE,L.RNG
;        MOV   STIMTBL(R3),R2 ;GET MAXIMUM SPEC TIME
;        JSR   PC,TYPDEC      ;CONVERT TO DECIMAL & TYPE
;        TYPE,DASH
;        MOV   STIMTBL+2(R3),R2 ;GET MINIMUM TIME
;        JSR   PC,TYPDEC      ;CONVERT TO DECIMAL & TYPE
;        TYPE,ANGTAB
;        TYPE,L.ACT
;        MOV   @#ATIME,R2     ;GET ACTUAL TIME
;        JSR   PC,TYPDEC      ;CONVERT TO DECIMAL & TYPE
;        TYPE,CRLF
;        MOV   (SP)+,R3
;        MOV   (SP)+,R2
;        RTS    PC           ;RETURN
        .SBTTL               TYPE GAP TIMES SUBROUTINE

```

```

1665      ; THIS SUBROUTINE IS USED TO TYPE THE SPECIFIED GAP SIZES (RECORDED IN
1666      ; TST021). IT IS CALLED BY THE GAPOK ROUTINE IF THE GAP SIZE IS OUT OF
1667      ; RANGE VIA THE HLT ROUTINE (HLT+2).
1668      ; CALL:  MOV#   #GAP,GAP           ; LOAD GAP # INTO GAP
1669      ;         MOV   #TIME,ATIME       ; LOAD ACTUAL TIME INTO ATIME
1670      ;         JSR   PC,OUTGAP
1671
1672      OUTGAP: MOV    R2,-(SP)           ; SAVE R2 AND R3
1673      MOV    R3,-(SP)
1674      MOV#   GAP,R3                   ; GET GAP #
1675      ASL    R3
1676      ASL    R3
1677      TYPE,L,RNG
1678      MOV    GTIMTBL(R3),R2           ; GET MAX TIME
1679      JSR    PC,TYPDEC                ; CONVERT TO DECIMAL & TYPE
1680      TYPE,DASH
1681      MOV    GTIMTBL+2(R3),R2        ; GET MIN TIME
1682      JSR    PC,TYPDEC                ; CONVERT TO DECIMAL & TYPE
1683      TYPE,ANGTAB
1684      TYPE,L,ACT
1685      MOV    @#ATIME,R2              ; GET ACTUAL TIME
1686      JSR    PC,TYPDEC                ; CONVERT TO DECIMAL & TYPE
1687      TYPE,E,GAP
1688      MOV#   @#GAP,R2                ; GET GAP #
1689      JSR    PC,TYPOCT                ; TYPE GAP #
1690      TYPE,CRLF
1691      MOV    (SP)+,R3                 ; RESTORE R3 AND R2
1692      MOV    (SP)+,R2
1693      RTS    PC
1694
1695      .SBTTL      ASCII TO OCTAL CONVERT SUBROUTINE
1696      ; SUBROUTINE TO CONVERT ASCII DATA TO OCTAL. CONVERTED OCTAL DATA
1697      ; IS LEFT IN OCTALO <15-00>.
1698      CNVTAO:
1699      JSR    PC,SAVE                   ; SAVE REGISTERS ON THE STACK
1700      MOV    #INBUF,R0                ; SET PTR TO ASCII DATA
1701      MOV    #OCTALO,R1               ; GET ADDRESS OF OCTAL DATA
1702      CLR    (R1)                     ; CLEAR OUT OLD OCTAL DATA
1703      CLR    2(R1)
1704      1$:  CMPB  #CR,(R0)               ; <CR> TERMINATES INPUT
1705      BEQ   3$
1706      MOV#   (R0)+,R2                 ; GET 'OCTAL' DATA
1707      BIC   #177770,R2                ; STRIP UNUSED BITS
1708      MOV    #3,R3                     ; SET SHIFT COUNT
1709      2$:  ASL    (R1)                 ; SHIFT LAST
1710      ROL   2(R1)                     ; OCTAL DIGIT
1711      DEC   R3
1712      BNE   2$
1713      BIS   R2,(R1)                   ; AND INSERT THIS DIGIT
1714      BR    1$                       ; GO GET NEXT DIGIT
1715      3$:
1716      JSR    PC,.RESTORE              ; RESTORE REGISTERS FROM THE STACK
1717      RTS    PC                       ; RETURN
1718
1719      .SBTTL      PUBLISH SUBROUTINE
1720      ; THE PUBLISH SUBROUTINE AVERAGES THE RECORDED TIMES FOR EACH TEST IT-

```

```

1721
1722
1723
1724
1725 003346
1726 003346 004767 177204
1727 003352 012700 001014
1728 003356 113701 001121
1729 003362 122701 000001
1730 003366 001423
1731 003370 005002
1732 003372 005003
1733 003374 122701 000020
1734 003400 001402
1735 003402 000000
1736 003404 000777
1737
1738
1739 003406 062002
1740 003410 005503
1741 003412 005301
1742 003414 001374
1743
1744 003416 012700 000004
1745 003422 006203
1746 003424 006002
1747 003426 005300
1748 003430 001374
1749 003432 010237 001012
1750
1751 003436 113700 001122
1752 003442 006300
1753 003444 016067 001672 000002
1754 003452 000004
1755 003454 000000
1756 003456 113702 001122
1757 003462 004767 177364
1758 003466 004767 177106
1759 003472 000207
1760
1761
1762
1763
1764
1765
1766 003474 010046
1767 003476 012700 001264
1768 003502 105737 177560
1769 003506 100375
1770
1771 003510 113746 177562
1772 003514 042716 000200
1773 003520 122716 000177
1774 003524 001004
1775 003526 124026
1776 003530 000004 001377
    
```

```

;ERATION (IF 16. ITERATIONS) AND PLACES THE AVERAGE RESULT IN 'ATIME'.
;IT TYPES THE NAME OF THE FUNCTION THAT WAS TIMED,THE TIME SPEC-
;IFICATION AND THE ACTUAL TIME .

PUBLISH:
JSR PC,SAVE ;SAVE REGISTERS ON THE STACK
MOV @ATIMBL,R0 ;GET TABLE ADDRESS CONTAINING TIMES
MOVB @ITCNT,R1 ;GET # OF ENTRIES (GIVEN BY ITERATION COUNT)
CMPB #1,R1 ;BRANCH IF SINGLE ITERATION
BEQ 4$ ;CLEAR 'SUM' REGISTERS
CLR R2 ;BRANCH IF 16. ITERATIONS
CLR R3 ;ITERATION COUNT MUST BE 1 OR 16.
CMPB #16.,R1 ;DO NOT CHANGE POSIT OF SW11
BEQ 1$ ;WHEN TEST IS RUNNING.
BR . ;SUM INDIVIDUAL TIMES

1$: ADD (R0)+,R2
ADC R3
DEC R1
BNE 1$

2$: MOV #4,R0
3$: ASR R3 ;SHIFT TIME IN R3 & R2 4 PLACES
ROR R2 ;RIGHT = DIVIDE BY 16.
DEC R0
BNE 3$
MOV R2,@ATIME ;MOVE AVERAGED TIMES

4$: MOVB @TSTNUM,R0 ;GET TEST #
ASL R0
MOV NAMPTR(R0),5$ ;GET TEST NAME STRING ADDRESS
TYPE
WORD 0
5$: MOVB @TSTNUM,R2 ;GET TEST #
JSR PC,OUTSPC ;OUTPUT TIMES
JSR PC,.RESTORE ;RESTORE REGISTERS FROM THE STACK
RTS PC

.SBTTL INPUT SUBROUTINE
;SUBROUTINE TO GET TTY INPUT
;CALL: JSR PC,INPUT
;INPUT DATA IS RETURNED IN BUFFER BEGINNING AT INBUF.

.INPUT: MOV R0,-(SP) ;SAVE R0 ON THE STACK
1$: MOV #INBUF,R0
2$: TSTB @TKS
BPL 2$

MOV B @TKB,-(SP) ;GET CHARACTER
BIC #200,(SP)
CMPB #177,(SP) ;CHECK RUBOUT
BNE 3$
CMPB -(R0),(SP)+ ;REMOVE CHARACTER FROM INPUT
TYPE,BKSLSH
    
```

```

1777 003534 000762
1778 003536 122716 000025
1779 003542 001004
1780 003544 005726
1781 003546 000004 001374
1782 003552 000751
1783 003554 111637 001401
1784 003560 111620
1785 003562 122726 000015
1786 003566 001403
1787 003570 000004 001401
1788 003574 000742
1789 003576 000004 001374
1790 003602 012600
1791 003604 000207
1792
1793
1794 003606 113746 177562
1795 003612 042716 000200
1796 003616 122716 000017
1797 003622 001003
1798 003624 112667 176475
1799 003630 000002
1800
1801 003632 122726 000003
1802 003636 001003
1803 003640 000005
1804 003642 000137 005724
1805 003646 000002

3$: BR 2$ ;WAIT FOR NEXT CHARACTER
CMPB #CNTRLU,(SP) ;CHECK CONTROL U (↑U)
BNE 4$
TST (SP)+
TYPE,CRLF
BR 1$
4$: MOVB (SP),2$ECHO
MOVB (SP),(RO)+
CMPB #CR,(SP)+
BEQ 5$
TYPE,ECHO
BR 2$
5$: TYPE,CRLF
MOV (SP)+,RO
RTS PC

:KEYBOARD INTERRUPT SERVICE ROUTINE
TKISR: MOVB 2$TKB,-(SP) ;GET TYPED CHARACTER
BIC #200,(SP) ;STRIP PARITY BIT
CMPB #CNTRL0,(SP) ;BRANCH IF NOT CONTROL 0 (↑0)
BNE 1$
MOVB (SP)+,$CNTRL0 ;SET CONTROL 0 INDICATOR IN TYPE ROUTINE
RTI ;EXIT

1$: CMPB #3,(SP)+ ;BRANCH IF NOT CONTROL C (↑C)
BNE 2$
RESET
JMP 2$INIT ;RESTART PROGRAM
2$: RTI ;EXIT

```

```

1806 .SBTTL ERROR SERVICE ROUTINES
1807 :ROUTINE TO PROCESS ERROR TRAPS (TRAPS TO 4)
1808 003650 000000 ERRTRP: HALT
1809
1810 :ERROR SERVICE ROUTINE
1811 :THIS ROUTINE PROCESSES TWO TYPES OF ERRORS (OUT OF RANGE AND HARDWARE)
1812 :THE CALLS FOR AN OUT OF RANGE ERROR ARE <HLT+1>, <HLT+2> AND, FOR A
1813 :HARDWARE ERROR THE CALL IS <HLT>.
1814
1815 003652 004767 176112 .HLT: JSR PC,CKSWR ;CHECK FOR CNTL G
1816 003656 004767 176674 JSR PC,SAVE ;SAVE REGISTERS ON THE STACK
1817 003662 110637 001123 15: MOVB SP,@ERFLG ;SET ERROR FLAG
1818 003666 032777 020000 175104 BIT #SW13,@SWR ;BRANCH IF NO TYP0UT
1819 003674 001075 BNE 4$
1820 003676 000004 014135 TYPE,E.HDR
1821 003702 113702 001122 MOVB @TSTNUM,R2 ;GET TEST #
1822 003706 004767 176716 JSR PC,TYP0CT ;AND TYPE IT
1823 003712 016600 000016 MOV 16(SP),R0 ;GET RETURN PC
1824 003716 162700 000002 SUB #2,R0 ;NOW PC OF HLT CALL
1825 003722 111000 MOVB (R0),R0 ;NOW HLT CALL ITSELF
1826 003724 001417 BEQ 2$ ;BRANCH IF HLT
1827 003726 000004 014220 TYPE,E.HDR2
1828 003732 122700 000002 CMPB #2,R0 ;BRANCH IF NOT HLT+2
1829 003736 001005 BNE 10$
1830 003740 004767 177202 JSR PC,OUTGAP ;TYPE GAP SPECIFIED TIMES
1831 003744 000004 001374 TYPE,CRLF
1832 003750 000447 BR 4$
1833 003752 004767 177074 10$: JSR PC,OUTSPC ;TYPE SPECIFIED TIMES
1834 003756 000004 001374 TYPE,CRLF
1835 003762 000442 BR 4$
1836 003764 016500 000014 2$: MOV ER(R5),R0
1837 003770 032765 002000 000032 BIT #PE1600,TC(R5)
1838 003776 001403 BEQ 20$
1839 004000 042700 102100 BIC #102100,R0
1840 004004 000402 BR 21$
1841 004006 042700 102300 20$: BIC #102300,R0
1842 004012 005700 21$: TST R0
1843 004014 001003 BNE 22$
1844 004016 000004 014111 TYPE,E.SFT ;TYPE SOFT ERROR MESSAGE
1845 004022 000434 BR 6$
1846
1847 004024 000004 014145 22$: TYPE,E.HDR1
1848 004030 010500 MOV R5,R0 ;GET FIRST ADDRESS OF REGS.
1849 004032 012701 000007 MOV #7,R1 ;TYPE FIRST 7 REGS.
1850 004036 012002 3$: MOV (R0)+,R2 ;GET REG CONTENTS
1851 004040 004767 176564 JSR PC,TYP0CT ;AND TYPE IT
1852 004044 000004 001407 TYPE,SPACE2
1853 004050 005301 DEC R1
1854 004052 001371 BNE 3$
1855 004054 016502 000032 MOV TC(R5),R2 ;GET CONTENTS OF TC REGISTER
1856 004060 004767 176544 JSR PC,TYP0CT
1857 004064 000004 001374 TYPE,CRLF
1858
1859 004070 032777 001000 174702 4$: BIT #SW09,@SWR ;BRANCH IF NO RING THE BELL
1860 004076 001402 BEQ 5$
1861 004100 000004 001403 TYPE,BELL
    
```

1862	004104	005777	174670	5S:	TST	JSWR		;HALT ON ERROR?
1863	004110	100001			BPL	6S		
1864	004112	000000			HALT			
1865	004114	004767	175650	6S:	JSR	PC,CKSWR		;CHECK FOR CNTL G
1866	004120	004767	176454		JSR	PC,.RESTORE		;RESTORE REGISTERS FROM THE STACK
1867	004124	000002			RTI			;RETURN
1868								
1869								



```

1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886 004126 004767 175636
1887 004132 013705 001010
1888 004136 032777 000400 174634
1889 004144 001404
1890 004146 113702 001122
1891 004152 004767 176674
1892 004156 032777 040000 174614
1893 004164 001417
1894 004166 004767 000546
1895 004172 004767 000776
1896 004176 105037 001123
1897 004202 016716 174574
1898 004206 010501
1899 004210 062701 000012
1900 004214 010500
1901 004216 062700 000006
1902 004222 000002
1903
1904 004224 105737 001123 2S:
1905 004230 001006
1906 004232 113700 001121
1907 004236 006300
1908 004240 013760 001012 001014
1909 004246 105237 001121 3S:
1910 004252 032777 004000 174520
1911 004260 001004
1912 004262 122737 000020 001121
1913 004270 001336
1914 004272 011637 001002 4S:
1915 004276 032777 002000 174474
1916 004304 001002
1917 004306 004767 177034
1918 004312 105037 001121 5S:
1919 004316 105777 174456
1920 004322 001721
1921 004324 017746 174450
1922 004330 042716 177740
1923 004334 005316
1924 004336 123726 001122
1925 004342 001311

```

```

.SBTTL SCOPE SUBROUTINE
:SCOPE ROUTINE
:THIS ROUTINE IS ENTERED UPON COMPLETION OF EACH SUBTEST
:THE SCOPE ROUTINE:
:   OUTPUTS TIME SPEC ON EACH ITERATION IF SW08 IS SET
:   REPEATS TEST IF SW14 IS SET
:   STORES ACTUAL TIME FOR FUNCTION IN TIME TABLE (ATIMTBL)
:   PUBLISHES TIME IF SW10=0
:   UPDATES ITERATION COUNT AND IF ITERATIONS COMPLETE CONTINUES
:   TO NEXT TEST, OTHERWISE REPEATS TEST.
:   DELAYS BEFORE CONTINUING OR REPEATING TEST.
:   INITIALIZES DRIVE
:RETURNS:   R5=BASE ADDRESS OF TMO2 REGISTERS (ADDRESS OF CS1)
:           R1='DS' REG ADDRESS
:           R0='FC' REG ADDRESS

.SCOPE: JSR   PC,CKSWR           ;CHECK FOR CNTL G
        MOV   @#TMBASE,R5      ;SET R5 TO FIRST TM REG
        BIT   #SW08,@SWR      ;BRANCH IF SPECIFICATION LINE
        BEQ   10$             ;NOT DESIRED ON EACH ITERATION
        MOVB  @#TSTNUM,R2     ;GET TEST NUMBER
        JSR   PC,OUTSPC       ;OUTPUT TIME RECORDED
        BIT   #SW14,@SWR      ;BRANCH IF CONTINUOUS LOOP
        BEQ   2$              ;NOT DESIRED
        JSR   PC,DELAY         ;DELAY 350 MS
        JSR   PC,RHINIT       ;INIT
        CLRB  @#ERFLG         ;CLEAR ERROR FLAG
        MOV   R5,R1           ;ADDRESS OF 'DS' REG IS IN R1
        ADD   #DS,R1
        MOV   R5,R0           ;ADDRESS OF 'FC' REG IS IN R0
        ADD   #FC,R0
        RTI

        TSTB  @#ERFLG         ;BRANCH IF ERROR FLAG IS SET
        BNE   3$
        MOVB  @#ITCNT,R0      ;GET ITERATION COUNT
        ASL   R0               ;STORE TIME IN TABLE
        MOV   @#ATIME,ATIMTBL(R0)
        INCB  @#ITCNT         ;INCREMENT ITERATION COUNT
        BIT   #SW11,@SWR      ;BRANCH IF SINGLE ITERATION DESIRED
        BNE   4$
        CMPB  #16.,@#ITCNT    ;BRANCH IF ITERATIONS INCOMPLETE
        BNE   1$
        MOV   (SP),@#SCPADR   ;SET SCOPE ADDRESS TO NEXT TEST
        BIT   #SW10,@SWR      ;BRANCH IF NO PUBLICATION DESIRED
        BNE   5$
        JSR   PC,PUBLISH      ;GO PUBLISH TEST DATA
        CLRB  @#ITCNT         ;RESET ITERATION COUNT
        TSTB  @SWR            ;BRANCH IF USER DOES NOT WANT TO
        BEQ   1$              ;HALT ON A SELECTED TEST
        MOV   @SWR,-(SP)      ;GET SWITCHES
        BIC   #177740,(SP)    ;CLEAR ALL BUT TEST #
        DEC   (SP)           ;FORM TEST # -1
        CMPB  @#TSTNUM,(SP)+  ;BRANCH IF NOT AT TEST
        BNE   1$

```

```

1926 004344 000000          HALT
1927 004346 004767 175416 JSR   PC,CKSWR          ;CHECK FOR CNTL G
1928 004352 000705          BR    1$
1929
1930          .SBTTL  TIMER SUBROUTINES
1931
1932          ;SUBROUTINE TO SYNCHRONIZE THE TIMER AND TURN IT ON.
1933          ;REGISTER 4 IS CLEARED, AND THE OSCILLATOR POLARITY IS MONITORED
1934          ;THE ROUTINE IS EXITED WHEN THE OSCILLATOR POLARITY CHANGES WITH R3
1935          ;SET TO INDICATE THE POLARITY OF THE OSCILLATOR.
1936          ;CALL: JSR   PC,TIMON
1937          ;RETURNS: R3 SET TO INDICATE LAST POLARITY (+24/-24=0/1)
1938          ;          R4 = 0
1939
1940 004354 005004          TIMON: CLR   R4          ;CLEAR TIME COUNT
1941 004356 012703 000024   MOV   #24,R3        ;SET POLARITY TO '0' STATE
1942 004362 032765 000100 000024   BIT   #OSC,MR(R5)   ;BRANCH IF POLARITY IS '0'
1943 004370 001405          BEQ   2$
1944 004372 032765 000100 000024 1$: BIT   #OSC,MR(R5)   ;WAIT FOR OSCILLATOR TO RETURN
1945 004400 001374          BNE   1$
1946 004402 000405          BR    4$
1947
1948 004404 005403          2$:  NEG   R3          ;NEGATE PREV POLARITY INDICATOR
1949 004406 032765 000100 000024 3$:  BIT   #OSC,MR(R5)   ;WAIT FOR OSCILLATOR TO RETURN
1950 004414 001774          BEQ   3$          ;TO '1' STATE
1951 004416 000207          4$:  RTS   PC
1952
1953          ;SUBROUTINE TO COUNT TIME
1954          ;EACH TIME THE OSCILLATOR TOGGLES (BIT <06> IN MR REG) REGISTER
1955          ;R4 IS INCREMENTED, AND THE REGISTER R3 IS NEGATED TO INDICATE
1956          ;THE LAST STATE OF THE OSCILLATOR.
1957          ;CALL JMP   TIMER(R3)          ;R3 IS SET BY TIMON ROUTINE
1958          ;R2=RETURN ADDRESS TO CALLER
1959          ;NOTE: THE TIME TO EXECUTE THIS ROUTINE IS VERY CRITICAL. IT MUST BE
1960          ;LESS THAN 40 US.
1961
1962          ;ENTER HERE VIA JMP  TIMER(R3) WHEN R3=-24 (PREV STATE=1)
1963 004420 032765 000100 000024 TIMER1: BIT   #OSC,MR(R5)   ;BRANCH IF CURRENT STATE IS '0'
1964 004426 001406          BEQ   TIMER        ;GO INCREMENT TIME
1965 004430 000112          JMP   (R2)          ;RETURN TO TEST
1966
1967          .=TIMER1+24
1968 004444 005403          TIMER: NEG   R3          ;NEGATE PREV STATE INDICATOR
1969 004446 005204          INC   R4          ;INCREMENT 'TICK' COUNT
1970 004450 100401          BMI   TIMERR       ;BRANCH ON OVERFLOW
1971 004452 000112          JMP   (R2)          ;RETURN TO TEST
1972 004454 000004 014246          TIMERR: TYPE,E.TIMOV  ;TYPE 'TIMER OVERFLOWED'
1973 004460 104400          HLT
1974 004462 000177 174314          JMP   @SCPADR       ;REPORT HARDWARE ERROR
1975          ;RETURN TO BEGINNING OF TEST
1976          .=TIMER+24
1977          ;ENTER HERE VIA JMP  TIMER(R3) WHEN R3=+24 (PREV STATE=0)
1978 004470 032765 000100 000024 TIMERO: BIT   #OSC,MR(R5)   ;BRANCH IF CURRENT STATE = '1'
1979 004476 001362          BNE   TIMER
1980 004500 000112          JMP   (R2)
1981

```

1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990 004502  
1991 004502 004767 176050  
1992 004506 012700 000070  
1993 004512 010401  
1994 004514 005002  
1995 004516 005003  
1996 004520 060002  
1997 004522 005503  
1998 004524 005301  
1999 004526 001374  
2000 004530 010246  
2001  
2002 004532 010346  
2003 004534 012746 000012  
2004 004540 004767 000262  
2005 004544 005726  
2006 004546 012637 001012  
2007 004552 113700 001122  
2008 004556 006300  
2009 004560 006300  
2010 004562 023760 001012 001416  
2011 004570 101004  
2012 004572 023760 001012 001420  
2013 004600 101001  
2014 004602 104401  
2015 004604  
2016 004604 004767 175770  
2017 004610 000207  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028 004612  
2029 004612 004767 175740  
2030 004616 012700 000070  
2031 004622 010401  
2032 004624 005002  
2033 004626 005003  
2034 004630 060002  
2035 004632 005503  
2036 004634 005301  
2037 004636 001374

```

;SUBROUTINE TO CHECK TIME RECORDED BY SUBTEST.
;THIS SUBROUTINE COMPUTES THE ACTUAL TIME (IN MICROSECONDS) AND CHECKS
;THAT THE TIME RECORDED BY THE SUBTEST IS CORRECT BY COMPARING THE TIME
;WITH THE HIGH LIMIT (STIMTBL(RO)) AND THE LOW LIMIT (STIMTBL+2(RO)).
;IF THE TIME IS OUT OF RANGE AN OUT OF RANGE ERROR TYPEOUT RESULTS.
;THE SUBROUTINE IS ENTERED WITH:
;   R4=TICK COUNT
    
```

```

TIMOK:
        JSR     PC,.SAVE           ;SAVE REGISTERS ON THE STACK
        MOV     #56,RO           ;GET TIME PER TICK
        MOV     R4,R1           ;GET TICKS COUNT
        CLR     R2               ;CLEAR SUMMING REGISTERS
        CLR     R3
15:     ADD     RO,R2             ;MULTIPLY TIME PER TICK
        ADC     R3               ;BY TICK COUNT
        DEC     R1
        BNE    1$
        MOV     R2,-(SP)         ;DIVIDE COUNT BY 10.
        MOV     R3,-(SP)
        MOV     #10,-(SP)
        JSR     PC,DIVIDE
        TST     (SP)+           ;DISCARD REMAINDER
        MOV     (SP)+,Q#ATIME   ;STORE QUOTIENT
        MOV     Q#TSTNUM,RO     ;GET TEST #
        ASL     RO
        ASL     RO
        CMP     Q#ATIME,STIMTBL(RO) ;CHECK THAT TIME IS WITHIN
        BHI    2$               ;LIMITS SPECIFIED
        CMP     Q#ATIME,STIMTBL+2(RO)
        BHI    3$
25:     HLT+1                    ;CALL ERROR ROUTINE
35:     JSR     PC,.RESTORE      ;RESTORE REGISTERS FROM THE STACK
        RTS     PC              ;RETURN
    
```

```

;SUBROUTINE TO CHECK INDIVIDUAL GAP TIMES (PRODUCED BY TST021)
;SUBROUTINE COMPUTES THE ACTUAL TIME (IN MICROSECONDS) AND CHECKS
;THAT THE GAP TIME RECORDED BY THE SUBTEST (TST021) BY COMPARING THE
;TIME WITH THE MAX LIMIT (GTIMTBL-GAPTBL(R1)) AND THE MIN LIMIT
;(GTIMTBL+2-GAPTBL(R1)).
;CALL: MOV     #TICK COUNT,R4   ;R4 CONTAINS TICK COUNT
;      MOV     #GAP,Q#GAP       ;LOCATION GAP CONTAINS GAP #
;      JSR     PC,GAPOK
    
```

```

GAPOK:
        JSR     PC,.SAVE           ;SAVE REGISTERS ON THE STACK
        MOV     #56,RO           ;GET TIME PER TICK
        MOV     R4,R1           ;GET TICK COUNT
        CLR     R2               ;CLEAR SUMMING REGISTERS
        CLR     R3
15:     ADD     RO,R2             ;MULTIPLY TICK COUNT
        ADC     R3               ;BY TIME PER TICK
        DEC     R1
        BNE    1$
    
```

```

2038
2039 004640 010246          MOV      R2,-(SP)          ;DIVIDE TIME BY 10.
2040 004642 010346          MOV      R3,-(SP)
2041 004644 012746 000012      MOV      #10,-(SP)
2042 004650 004767 000152      JSR      PC,DIVIDE
2043 004654 005726          TST      (SP)+          ;DISCARD REMAINDER
2044 004656 012637 001012      MOV      (SP)+,@#ATIME  ;STORE QUOTIENT
2045 004662 113703 001120      MOV      @#GAP,R3      ;GET GAP #
2046 004666 006303          ASL      R3            ;MULTPLY BY 4
2047 004670 006303          ASL      R3            ;TO GET AT TABLE ENTRY
2048 004672 023763 001012 001572      CMP      @#ATIME,GTIMTBL(R3) ;CHECK TIME (MAX)
2049 004700 101004          BHI      2$
2050 004702 023763 001012 001574      CMP      @#ATIME,GTIMTBL+2(R3) ;CHECK TIME (MIN)
2051 004710 101002          BHI      3$
2052 004712 104402          HLT+2      2$          ;REPORT OUT OF RANGE ERROR
2053 004714 000406          BR        100$
2054 004716 032777 000400 174054 3$:      BIT      #SW08,@SWR    ;BRANCH IF TIMES NOT WANTED
2055 004724 001402          BEQ      100$
2056 004726 004767 176214          JSR      PC,OUTGAP    ;TYPE GAP TIMES
2057
2058 004732          100$:
2059 004732 004767 175642          JSR      PC,.RESTORE  ;RESTORE REGISTERS FROM THE STACK
2060 004736 000207          RTS      PC          ;RETURN TO TEST
2061
2062          .SBTTL          DELAY SUBROUTINES
2063          ;THIS SUBROUTINE CAUSES A DELAY OF 350 MS.
2064 004740 004767 177410      DELAY: JSR      PC,TIMON
2065 004744 010246          MOV      R2,-(SP)          ;SAVE R2 ON THE STACK
2066 004746 012702 004756          MOV      #2$,R2          ;SET RETURN ADDRESS FOR TIMER
2067 004752          1$:
2068 004752 000163 004444          JMP      TIMER(R3)      ;GO TO TIMER & RETURN VIA R2
2069 004756 032704 004000      2$:      BIT      #4000,R4
2070 004762 001773          BEQ      1$
2071 004764 012602          MOV      (SP)+,R2      ;RESTORE R2
2072 004766 000207          RTS      PC
2073
2074          ;THIS SUBROUTINE ALLOWS A CALLER SPECIFIED DELAY (UP TO 65MS.)
2075          ;CALL: MOV      DELAY TIME,DELTIM    ;LOAD DELAY TIME (IN US)
2076          ;CALL: JSR      PC,DELAYV
2077 004770 005767 174120      DELAYV: TST      DELTIM    ;BRANCH IF 0 DELAY
2078 004774 001413          BEQ      3$
2079 004776 004767 177352          JSR      PC,TIMON    ;TURN TIMER ON
2080 005002 010246          MOV      R2,-(SP)          ;SAVE R2 ON THE STACK
2081 005004 012702 005014          MOV      #2$,R2          ;SET RETURN ADDRESS FROM TIMER
2082 005010          1$:
2083 005010 000163 004444          JMP      TIMER(R3)      ;GO TO TIMER & RETURN VIA R2
2084 005014 023704 001114      2$:      CMP      @#DELTIM,R4
2085 005020 101373          BHI      1$
2086 005022 012602          MOV      (SP)+,R2      ;RESTORE R2
2087 005024 000207          3$:      RTS      PC
2088
2089          .SBTTL          DIVIDE SUBROUTINE
2090          ;THIS SUBROUTINE DIVIDES A DOUBLE PRECISION # AND RETURNS THE RESULT
2091          ;TO THE CALLER ON THE STACK. BOTH DIVIDEND & DIVISOR MUST BE POSITIVE.
2092          ;CALL: MOV      LEAST SIGNIFICANT HALF DIVIDEND,-(SP)
2093          ;CALL: MOV      #MOST SIGNIFICANT HALF DIVIDEND,-(SP)

```

```

2094      :      MOV      #DIVISOR, -(SP)
2095      :      JSR      PC, DIVIDE
2096      : RETURN
2097      :      (SP)=REMAINDER ON STACK
2098      :      2(SP)=QUOTIENT
2099
2100      ;NOTE: THIS SUBROUTINE DESTROYS PREVIOUS CONTENTS OF R0,R1,R2 & R3.
2101
2102      DIVIDE: CLR      -(SP)      ;SAVE LOC FOR SIGNS
2103      MOV      #17, -(SP)      ;SET ITERATION COUNT
2104      MOV      12(SP), R1      ;GET LSH DIVIDEND
2105      MOV      10(SP), R0      ;GET MSH DIVIDEND
2106      MOV      6(SP), R2      ;GET DIVISOR
2107      NEG      R2              ;NEGATE DIVISOR
2108      CLC
2109      BR      2$              ;CLEAR 'C' BIT IN PSW
2110      1$:  ROL      R0          ;ROTATE MSH DIVIDEND
2111      MOV      R0, R3          ;SAVE IN R3
2112      ADD      R2, R3          ;SUBTRACT DIVISOR FROM MSH DIVIDEND
2113      BCC     2$              ;BRANCH IF DIVIDEND > DIVISOR
2114      MOV      R3, R0          ;SAVE REMAINDER IN R0
2115      2$:  ROL      R1          ;ROTATE LSH DIVIDEND
2116      DEC     (SP)            ;DECREMENT ITERATION COUNT
2117      BNE     1$
2118      TST     (SP)+           ;POP ITERATION COUNTER
2119      TST     (SP)+           ;POP SIGN CORRECTION
2120      MOV     R1, 6(SP)        ;PUSH REMAINDER ON STACK
2121      MOV     R0, 4(SP)        ;PUSH QUOTIENT ONTO STACK
2122      MOV     (SP)+, (SP)

```

```

2123 005114 000207          RTS    PC
2124
2125          SBTTL  DRIVE SUBROUTINES
2126          ;SUBROUTINE TO CHECK IF DRIVE IS AVAILABLE
2127          ;CALL:  MOVB  DRIVE#,DRVNUM
2128          JSR    PC,DRVAVA
2129          ;RETURN: 'C' BIT SET IF NOT AVAILABLE
2130 005116 113765 001004 000010 DRVAVA: MOVB  @DRVNUM,CS2(R5)      ;LOAD DRIVE #
2131 005124 032765 040000 000026          BIT    #TAP,DT(R5)          ;CHECK IF TAPE UNIT
2132 005132 001003
2133 005134 004767 000034          BNE   1$
2134 005140 000262          JSR    PC,RHINIT
2135 005142 000207          SEV
2136          ;SET 'V' TO IND NOT AVAIL
2137          1$: RTS    PC          ;RETURN
2138
2139          ;SUBROUTINE TO CHECK IF TE16 SLAVE IS AVAILABLE FOR TEST
2140          ;CALL:  MOVB  DRIVE #,@DRVNUM      ;PASS DRIVE # VIA DRVNUM
2141          MOVB  SLAVE #,@SLVNUM      ;PASS SLAVE # VIA SLVNUM
2142          JSR    PC,SLVAVA
2143          ;CALL SUBROUTINE
2144 005144 113765 001004 000010 SLVAVA: MOVB  @DRVNUM,CS2(R5)      ;LOAD DRIVE #
2145 005152 113765 001005 000032          MOVB  @SLVNUM,TC(R5)          ;AND SLAVE #
2146 005160 032765 002000 000026          BIT    #SPR,DT(R5)          ;BRANCH IF SLAVE PRESENT
2147 005166 001001
2148 005170 000262          BNE   1$
2149 005172 000207          SEV
2150          ;SET 'V' TO INDICATE NO SLAVE
2151          1$: RTS    PC
2152
2153          ;SUBROUTINE TO INITIALIZE RH CONTROLLER
2154          ;CALL:  JSR PC,RHINIT
2155
2156 005174 012765 000040 000010 RHINIT: MOV    #40,CS2(R5)
2157 005202 113765 001004 000010          MOVB  @DRVNUM,CS2(R5)
2158 005210 005046
2159          CLR    -(SP)
2160 005212 113716 001005          MOVB  @SLVNUM,(SP)
2161 005216 012665 000032          MOV    (SP)+,TC(R5)          ;LOAD SLAVE # INTO TC REG
2162 005222 052765 000300 000032          BIS    #NORM1,TC(R5)
2163 005230 000207          RTS    PC
2164
2165          ;SUBROUTINE TO WAIT FOR DRIVE READY (DRY)
2166 005232 005027          WAITRDY:CLR    (PC)+          ;CLEAR WAIT TIMER
2167 005234 000000          WAITTIM: WORD 0
2168 005236 105765 000012          1$:  TSTB  DS(R5)          ;WAIT FOR READY TO SET
2169 005242 100406          BMI   2$
2170 005244 005267 177764          INC   WAITTIM          ;INCREMENT WAIT TIMER
2171 005250 001372          BNE   1$          ;BRANCH IF TIME HAS NOT EXPIRED
2172 005252 000004 014273          TYPE,E.TIMEXP          ;TYPE 'TIME EXPIRED WAITING FOR RDY'
2173 005256 000425          BR    99$          ;TAKE ERROR EXIT
2174 005260 032765 002000 000012 2$: BIT    #EOT,DS(R5)          ;CHECK FOR END OF TAPE
2175 005266 001415          BEQ   3$          ;BRANCH IF NO EOT
2176 005270 000004 013330          TYPE,M.NAM
2177 005274 000004 013656          TYPE,M.EOT          ;TYPE 'END OF TAPE'
2178 005300 004767 000032          JSR    PC,.REWIND          ;REWIND SLAVE
2179 005304 102412          BVS   99$          ;BRANCH IF ERROR ON REWIND
2180 005306 004767 000106          JSR    PC,WRITE          ;WRITE A RECORD
2181 005312 005215          INC   (R5)          ;SET 'GO' BIT
2182 005314 004767 177712          JSR    PC,WAITRDY          ;WAIT FOR READY
2183 005320 000404          BR    99$          ;TAKE ERROR EXIT
2184 005322 032765 040000 000012 3$: BIT    #ERR,DS(R5)          ;CHECK ERROR EXIT

```

C05

DZTUG-B TMO2/TE16 DRIVE FUNCTION TIMER  
DZTUGB.P11 15-JUL-77 12:52

MACY11 30(1046) 15-JUL-77 12:55 PAGE 53  
DRIVE SUBROUTINES

SEQ 0053

2179 005330 001401  
2180 005332 000262  
2181 005334 000207  
2182

99\$: BEQ 100\$  
SEV  
100\$: RTS PC

DZTUG-B TMO2/TE16 DRIVE FUNCTION TIMER  
DZTUGB.P11 15-JUL-77 12:52

MACY11 30(1046) 15-JUL-77 12:55 PAGE 54  
DRIVE SUBROUTINES

SEQ 0054

2183  
2184

;SUBROUTINE TO REWIND A UNIT (DRIVE/SLAVE COMBINATION)  
;CALL MOV8 DRIVE #, @#DRVNUM



E05

DZTUG-B TMO2/TE16 DRIVE FUNCTION TIMER  
DZTUGB.P11 15-JUL-77 12:52

MACY11 30(1046) 15-JUL-77 12:55 PAGE 55  
DRIVE SUBROUTINES

SEQ 0055

```
2185      :      MOVB   SLAVE #,3#SLVNUM
2186      :      JSR    PC,.REWIND
2187      :SUBROUTINE RETURNS TO CALLER WITH SELECTED SLAVE AT 'BOT', & 'V' SET IF
2188      :AN ERROR OCCURS.
2189
2190 005336 004767 177632      .REWIND:JSR    PC,RHINIT      ;INITIALIZE CONTROLLER
```

DZTUG-B TMO2/TE16 DRIVE FUNCTION TIMER  
DZTUGB.P11 15-JUL-77 12:52MACY11 30(1046) 15-JUL-77 12:55 PAGE 56  
DRIVE SUBROUTINES

SEQ 0056

```

2191 005342 004367 000206      JSR      R3,TMCMD      ;GO TO TM COMMAND SUBROUTINE
2192 005346 000000              .WORD    0             ;BUS ADDRESS (NOT USED)
2193 005350 000000              .WORD    0             ;WORD COUNT (NOT USED)
2194 005352 000000              .WORD    0             ;FRAME COUNT (NOT USED)
2195 005354 000006              .WORD    RMD          ;REWIND COMMAND
2196 005356 005215              INC      (R5)          ;SET 'GO' BIT
2197 005360 032765 000002 000012 1$: BIT      #BOT,DS(R5)   ;BRANCH IF 'BOT' SET
2198 005366 001005              BNE     2$            ;
2199 005370 032765 040000 000012  BIT      #ERR,DS(R5)   ;CHECK ERROR BIT
2200 005376 001006              BNE     99$          ;BRANCH IF ERROR BIT SET
2201 005400 000767              BR      1$            ;
2202
2203 005402 032765 020000 000012 2$: BIT      #PIP,DS(R5)   ;WAIT FOR TAPE MOTION TO STOP
2204 005410 001374              BNE     2$            ;
2205 005412 000401              BR      100$         ;
2206 005414 000262              99$:   SEV            ;
2207 005416 000207              100$:  RTS           PC
2208
2209              ;SUBROUTINE TO WRITE 256. WORD RECORD
2210              ;CALL: JSR      PC,WRITE
2211
2212 005420 004367 000130  WRITE: JSR      R3,TMCMD      ;GO TO TM COMMAND SUBROUTINE
2213 005424 015700              .WORD    WTBUF        ;BUS ADDRESS
2214 005426 177600              .WORD    WRDCNT       ;WORD COUNT
2215 005430 177400              .WORD    FRMCNT       ;FRAME COUNT
2216 005432 000060              .WORD    WFWD         ;WRITE FORWARD COMMAND
2217 005434 000207              RTS           PC
2218
2219              ;SUBROUTINE TO READ A 256. WORD RECORD.
2220              ;CALL: JSR      PC,READ
2221
2222 005436 004337 005554  READ:   JSR      R3,#TMCMD
2223 005442 015700              .WORD    RDBUF        ;ADDRESS OF READ BUFFER
2224 005444 177600              .WORD    WRDCNT       ;2'S COMPLEMENT OF WORD COUNT
2225 005446 177400              .WORD    FRMCNT       ;2'S COMPLEMENT OF FRAME COUNT
2226 005450 000070              .WORD    RDFWD        ;READ FORWARD COMMAND
2227 005452 000207              RTS           PC
2228
2229              ;SUBROUTINE TO INITIATE READ REVERSE COMMAND
2230              ;CALL: JSR      PC,REVRO
2231
2232 005454 004367 000074  REVRO: JSR      R3,TMCMD
2233 005460 016300              .WORD    RDBUF+256.  ;ADDRESS OF READ REVERSE BUFFER
2234 005462 177600              .WORD    WRDCNT       ;2'S COMPLEMENT OF WORD COUNT
2235 005464 177400              .WORD    FRMCNT       ;2'S COMPLEMENT OF FRAME COUNT
2236 005466 000076              .WORD    RDREV        ;READ REVERSE COMMAND
2237 005470 000207              RTS           PC
2238
2239              ;SUBROUTINE TO SPACE FORWARD 1 RECORD
2240 005472 012765 177777 000006  FWDSPC: MOV     #-1,FC(R5)  ;LOAD RECORD COUNT
2241 005500 012715 000031              MOV     #SPCFWD+1,(R5) ;LOAD COMMAND
2242 005504 004767 177522              JSR     PC,WAITRDY    ;WAIT FOR READY
2243 005510 000207              RTS     PC            ;RETURN
2244
2245              ;SUBROUTINE TO WRITE A RECORD AND BACK SPACE OVER THE RECORD.
2246 005512 004767 177702  WRT.BK: JSR     PC,WRITE ;WRITE THE RECORD

```

```

2247 005516 005215          INC      (R5)          ;SET 'GO' BIT
2248 005520 004767 177506   JSR      PC, WAITRDY
2249 005524 102412          BVS     2$
2250 005526 012765 177777 000006   MOV     #-1, FC(R5)   ;LOAD RECORD COUNT
2251 005534 012715 000033   MOV     #SPCREV+1, (R5) ;LOAD COMMAND
2252 005540 004767 177466   JSR     PC, WAITRDY
2253 005544 102402          BVS     2$
2254 005546 004767 177166   1$:    JSR     PC, DELAY ;WAIT FOR TAPE MOTION TO STOP
2255 005552 000207          2$:    RTS     PC
2256
2257          ;SUBROUTINE TO LOAD A COMMAND
2258          ;CALL: JSR     R3, TMCMD
2259          .WORD   BUS ADDRESS
2260          .WORD   WORD COUNT (2'S COMPLEMENT)
2261          .WORD   FRAME COUNT (2'S COMPLEMENT)
2262          .WORD   COMMAND
2263
2264 005554 012365 000004   TMCMD: MOV     (R3)+, BA(R5) ;LOAD BUS ADDRESS
2265 005560 012365 000002   MOV     (R3)+, WC(R5) ;LOAD WORD COUNT
2266 005564 012365 000006   MOV     (R3)+, FC(R5) ;LOAD FRAME COUNT
2267 005570 012315   MOV     (R3)+, (R5)   ;LOAD COMMAND
2268 005572 000203   RTS     R3            ;RETURN
2269
2270          ;SUBROUTINE TO PRINT TE16 SERIAL NUMBER
2271          ;JSR     PC, SNPT
2272
2273 005574 016503 000030   SNPT:  MOV     SN(R5), R3
2274 005600 012701 001144   MOV     #ODIGITS, R1
2275 005604 000303   SWAB   R3
2276 005606 006003   ROR    R3
2277 005610 006003   ROR    R3
2278 005612 006003   ROR    R3
2279 005614 006003   ROR    R3          ;GET FIRST DIGIT
2280 005616 042703 177760   BIC     #177760, R3
2281 005622 052703 000260   BIS     #260, R3
2282 005626 110321   MOVB   R3, (R1)+   ;FILL FIRST DIGIT
2283 005630 016503 000030   MOV     SN(R5), R3
2284 005634 000303   SWAB   R3
2285 005636 042703 177760   BIC     #177760, R3
2286 005642 052703 000260   BIS     #260, R3
2287 005646 110321   MOVB   R3, (R1)+   ;GET SECOND DIGIT
2288 005650 016503 000030   MOV     SN(R5), R3
2289 005654 006003   ROR    R3
2290 005656 006003   ROR    R3
2291 005660 006003   ROR    R3
2292 005662 006003   ROR    R3
2293 005664 042703 177760   BIC     #177760, R3
2294 005670 052703 000260   BIS     #260, R3
2295 005674 110321   MOVB   R3, (R1)+   ;GET THIRD DIGIT
2296 005676 016503 000030   MOV     SN(R5), R3
2297 005702 042703 177760   BIC     #177760, R3
2298 005706 052703 000260   BIS     #260, R3
2299 005712 110321   MOVB   R3, (R1)+   ;GET FOURTH DIGIT
2300 005714 105011   CLRB   (R1)
2301 005716 000004 001144   TYPE, ODIGITS      ;TYPE SERIAL NUMBER
2302 005722 000207   RTS     PC          ;RETURN

```

```

2303
2304          .SBTTL  PROGRAM INITIALIZATION
2305 005724 012706 000600  INIT:  MOV      #STKPTR,SP      ;SET STACK PTR
2306
2307 005730 013746 000006  SUSWR: MOV      @#6,-(SP)      ;SAVE VECTORS
2308 005734 013746 000004      MOV      @#4,-(SP)
2309 005740 012737 005760 000004      MOV      @#61$,@#4      ;SET UP FOR TIMEOUT
2310 005746 022777 177777 173024      CMP      #-1,@SWR      ;REFERENCE HARDWARE SWITCH REGISTER
2311 005754 001402
2312 005756 000404
2313 005760 022626      61$:  CMP      (SP)+,(SP)+      ;ADJUST STACK
2314 005762 012767 000176 173010 60$:  MOV      @SWREG,SWR      ;POINT TO SOFTWARE SWITCH REG
2315 005770 012637 000004      62$:  MOV      (SP)+,@#4      ;RESTORE VECTORS
2316 005774 012637 000006      MOV      (SP)+,@#6
2317 006000 022737 000176 001000      CMP      @SWREG,@SWR
2318 006006 001002
2319 006010 004767 174014      JSR      PC,CNTLU
2320 006014 105037 001124      64$:  CLRB     @#PRGFLG      ;CLEAR PROGRAM FLAG
2321 006020 105037 001121      CLRB     @#ITCNT      ;CLEAR ITERATION COUNT
2322 006024 105037 001122      CLRB     @#TSTNUM      ;SET TEST # 0
2323 006030 105037 001123      CLRB     @#ERFLG      ;CLEAR ERROR FLAG
2324 006034 105067 173070      CLRB     @#ASFLG      ;CLEAR ASK FLAG
2325 006040 012737 000006 000004      MOV      @ERRVEC+2,@ERRVEC
2326 006046 012737 000002 000006      MOV      @RTI,@ERRVEC+2      ;CHECK IF 'LP' IS AVAILABLE
2327 006054 005037 001264      2$:  CLR      @#INBUF
2328 006060 000004 001374      TYPE,CRLF
2329 006064 000004 013330      TYPE,M.NAM      ;TYPE TITLE
2330 006070 000004 013376      TYPE,I.REG      ;ASK USER TO TYPE CONT BASE ADRS
2331 006074 004767 175374      JSR      PC,.INPUT      ;GET USER INPUT
2332 006100 004767 175154      4$:  JSR      PC,CNVTA0      ;CONVERT ASCII TO OCTAL
2333 006104 013737 001116 001010      MOV      @#OCTALO,@#TMBASE      ;SET NEW ADDRESS
2334 006112 013705 001010      5$:  MOV      @#TMBASE,R5
2335
2336          ;ROUTINE TO CHECK IF CONTROLLER (RH11) IS AVAILAABLE
2337 006116 000261      SEC      ;SET 'C' IN PSW
2338 006120 005715      TST      (R5)      ;BRANCH IF CONTROLLER AVAIL
2339 006122 103003      BCC     6$
2340 006124 000004 013715      TYPE,E.NCON
2341 006130 000675      BR      INIT
2342 006132 012737 003650 000004 6$:  MOV      @ERRTRP,@ERRVEC      ;SET ERROR TRAP VECTOR
    
```

```

2343          :ROUTINE TO GET TMD2 DRIVES USER DESIRES TO TEST
2344 006140 105037 001123 DRIVES: CLR  @#ERFLG          ;CLEAR ERROR FLAG
2345 006144 012701 001154      MOV  @DRVTBL,R1      ;MARK ALL DRIVES AS NOT TO
2346 006150 012700 000004      MOV  @4,RO          ;BE TESTED. A '0' INDICATES
2347 006154 005021              1$: CLR  (R1)+          ;THAT A DRIVE IS NOT TO BE
2348 006156 005300              DEC  RO            ;TESTED
2349 006160 001375              BNE  1$
2350 006162 000004 013443      TYPE,I.DRVS
2351 006166 004767 175302      JSR  PC,INPUT      ;GET USER INPUT
2352 006172 012700 001264      MOV  @INBUF,RO
2353 006176 122710 000101      CMPB @'A,(RO)     ;AN 'A' SPECIFIES ALL
2354 006202 001013              BNE  3$           ;DRIVES TO BE TESTED
2355 006204 110667 172714      MOV  SP,PRGFLG    ;SET FLAG TO IND ALL DRIVES
2356 006210 012701 001154      MOV  @DRVTBL,R1  ;MARK ALL DRIVES TO BE TESTED
2357 006214 012700 000004      MOV  @4,RO        ;A '-1' INDICATES THAT A DRIVE
2358 006220 012721 177777      2$: MOV  @-1,(R1)+ ;IS TO BE TESTED
2359 006224 005300              DEC  RO
2360 006226 001374              BNE  2$
2361 006230 000417              BR   CHKDRV       ;GO CHECK DRIVE AVAILABILITY
2362
2363          :GET USER SELECTED DRIVES AND MARK EACH DRIVE SELECTED TO BE TESTED
2364 006232 122710 000015      3$: CMPB @CR,(RO)
2365 006236 001414              BEQ  CHKDRV
2366 006240 121027 000054      CMPB (RO),@',    ;CHECK IF 'COMMA'
2367 006244 001001              BNE  4$
2368 006246 105720              TSTB (RO)+        ;STEP PTR PAST 'COMMA'
2369 006250 112001              4$: MOV  (RO)+,R1
2370 006252 042701 177770      BIC  @177770,R1
2371 006256 112761 177777 001154  MOV  @-1,DRVTBL(R1)
2372 006264 000240              NOP
2373 006266 000761              BR   3$
2374
2375          :ASCERTAIN THAT DRIVES (TMD2'S) SPECIFIED ARE AVAILABLE
2376 006270 005000              CHKDRV: CLR  RO    ;A 0/-1 INDICATES THAT THE
2377 006272 105760 001154      1$: TSTB DRVTBL(RO) ;DRIVE IS NOT/IS TO BE TESTED
2378 006276 001005              BNE  3$
2379 006300 005200              2$: INC  RO
2380 006302 122700 000010      CMPB @@.,RO
2381 006306 001371              BNE  1$
2382 006310 000421              BR   4$
2383 006312 110037 001004      3$: MOV  RO,@#DRVNUM
2384 006316 004737 005116      JSR  PC,@#DRVAVA  ;CHECK IF AVAILABLE
2385 006322 102366              BVC  2$          ;'V' BIT SET INDICATES NOT AVAIL
2386 006324 000004 013762      TYPE,E.NDRV
2387 006330 116037 001132 014014  MOV  DIGTAB(RO),@#E.NAVA ;SET DRIVE # IN MESSAGE
2388 006336 000004 014014      TYPE,E.NAVA
2389 006342 110637 001123      MOV  SP,@#ERFLG  ;SET 'ERROR' FLAG
2390 006346 105060 001154      CLR  DRVTBL(RO) ;MARK DRIVE UNAVAILABLE
2391 006352 000752              BR   2$          ;CHECK NEXT DRIVE
2392 006354 105737 001123      4$: TSTB @#ERFLG  ;GO GET SLAVES IF NO ERROR
2393 006360 001403              BEQ  SLAVES
2394 006362 105737 001124      TSTB @#PRGFLG   ;ASK USER TO RETYPE DRIVES IF
2395 006366 001664              BEQ  DRIVES      ;'ALL' NOT SPECIFIED
2396
2397          :ROUTINE TO GET SLAVES (TE16'S) USER DESIRES TO TEST
2398 006370 105037 001123 SLAVES: CLR  @#ERFLG ;CLEAR 'ERROR' FLAG

```



```

2455 006630 005001          3$: CLR R1 ;SET SLAVE # 0
2456 006632 105712          4$: TSTB (R2) ;BRANCH IF DRIVE'S SLAVE IS SEL-
2457 006634 001006          BNE 6$ ;ECTED FOR TEST
2458 006636 005201          5$: INC R1 ;INCREMENT SLAVE #
2459 006640 005202          INC R2 ;STEP PTR TO NEXT SLAVE
2460 006642 022701 000010  CMP #8.,R1 ;GO TO 4$ IF ALL SLAVES NOT
2461 006646 001371          BNE 4$ ;CHECKED
2462 006650 000760          BR 2$ ;OTHERWISE GO TO 2$ ABOVE
2463
2464 006652 110037 001004  6$: MOVB RO, @#DRVNUM ;PASS DRIVE & SLAVE #
2465 006656 110137 001005  MOVB R1, @#SLVNUM
2466 006662 004737 005144  JSR PC, @#SLVAVA ;AND CHECK IF AVAILABLE
2467 006666 102363          BVC 5$ ;'V' BIT SET ON RETURN IND-
2468 006670 116037 001132 014004  MOVB DIGTAB(RO), @#E.DRV ;ICATES ERROR. PREPARE ERROR
2469 006676 116137 001132 014014  MOVB DIGTAB(R1), @#E.NAVA ;MESSAGE
2470 006704 000004 013776  TYPE, E.NSLV
2471 006710 110637 001123  MOVB SP, @#ERFLG ;SET ERROR INDICATOR
2472 006714 105012          CLRB (R2) ;CLEAR SLAVE TABLE ENTRY
2473 006716 000747          BR 5$ ;GET NEXT SLAVE
2474
2475 006720 105737 001123  7$: TSTB @#ERFLG ;BRANCH IF NO ERROR
2476 006724 001403          BEQ 100$
2477 006726 105737 001124  TSTB @#PRGFLG ;BRANCH IF NOT 'ALL'
2478 006732 001616          BEQ SLAVES ;ASK USER TO RETYPE SLAVES
2479 006734 012737 003650 000004  100$: MOV @#ERRTRP, @#ERRVEC
2480
2481 ;SCAN DIVE AND SLAVE TABLE FOR DRIVE/SLAVE COMBINATION TO TEST
2482 006742 105037 001004  CLRB @#DRVNUM ;SET DRIVE AND SLAVE # 0
2483 006746 105037 001005  CLRB @#SLVNUM
2484 006752 012737 001164 001006  MOV @#SLVTBL, @#SLVPTR ;SET PTR TO SLAVE TABLE
2485 006760 105037 001125  CLRB @#UNTFND ;CLEAR 'UNIT FOUND' IND.
2486
2487 006764 113700 001004  BEGIN: MOVB @#DRVNUM, RO ;GET DRIVE #
2488 006770 113701 001005  MOVB @#SLVNUM, R1 ;AND SLAVE #

```

DZTUG-B TH02/TE16 DRIVE FUNCTION TIMER  
DZTUG8.P11 15-JUL-77 12:52

MACY11 30(1046) 15-JUL-77 12:55 PAGE 62  
PROGRAM INITIALIZATION

SEQ 0062

2489	006774	013702	001006		MOV	2#SLVPTR,R2		;GET SLAVE PTR
2490	007000	105760	001154	1S:	TSTB	DRVTBL(R0)		;BRANCH IF DRIVE AVAIL TO TEST
2491	007004	001011			BNE	3S		
2492	007006	005001			CLR	R1		;CLEAR SLAVE #
2493	007010	062702	000010		ADD	#8.,R2		;AND STEP PTR TO NEXT DRIVE'S
2494	007014	005200		2S:	INC	R0		;SLAVES AND INCREMENT DRIVE #
2495	007016	022700	000010		CMP	#8.,R0		;EXIT TEST IF ALL DRIVES
2496	007022	001366			BNE	1S		;CHECKED OTHERWISE CONTINUE
2497	007024	000137	012732		JMP	2#END		;SCAN FOR NEXT 'UNIT'
2498								
2499	007030	105712		3S:	TSTB	(R2)		;BRANCH IF SLAVE ON DRIVE IS
2500	007032	001007			BNE	4S		;AVAILABLE THERWISE STEP
2501	007034	005202			INC	R2		;PTR TO NEXT SLAVE
2502	007036	005201			INC	R1		;INCREMENT SLAVE #
2503	007040	122701	000010		CMPB	#8.,R1		;UNTIL ALL SLAVES CHECKED
2504	007044	001371			BNE	3S		;WHEN ALL SLAVES CHECKED
2505	007046	005001			CLR	R1		;SET SLAVE # 0
2506	007050	000761			BR	2S		;AND CONTINUE SCAN
2507								
2508	007052	110637	001125	4S:	MOVB	SP,2#UNTFND		;INDICATE THAT A 'UNIT' IS FOUND
2509	007056	110037	001004		MOVB	RO,2#DRVNUM		;SET DRIVE 3



2510	007062	110137	001005		MOVB	R1, @SLVNUM		;SET SLAVE #
2511	007066	010237	001006		MOV	R2, @SLVPTR		;SAVE SLAVE PTR
2512								
2513	007072	105737	001130		5S:	TSTB	@ASFLG	
2514	007076	001044				BNE	7S	
2515	007100	112767	000001	172022		MOVB	@1, ASFLG	
2516								
2517	007106	105037	001124			CLRB	@PRGFLG	;CLEAR PROGRAM INDICATOR
2518	007112	000004	013564			TYPE, I.SKEW		;ASK USER IF HE WANTS TO RUN SKEW TESTS
2519	007116	004767	174352			JSR	PC, .INPUT	;GET USER INPUT
2520	007122	012703	001264			MOV	@INBUF, R3	;GET REPLY
2521	007126	122713	000060			CMPB	@'0', (R3)	;BRANCH IF 'NO' (0)
2522	007132	001406				BEQ	6S	
2523	007134	122713	000061			CMPB	@'1', (R3)	;CHECK IF 'YES' (1)
2524	007140	001354				BNE	5S	;NEITHER SO ASK AGAIN
2525	007142	111337	001124			MOVB	(R3), @PRGFLG	;SET INDICATOR
2526	007146	000420				BR	7S	
2527								
2528	007150	105037	001127		6S:	CLRB	@NRZFLG	;CLEAR NRZ INDICATOR
2529	007154	000004	013625			TYPE, I.NRZ		;ASK USER IF DRIVE 'NRZ' ONLY
2530	007160	004767	174310			JSR	PC, .INPUT	;GET USER INPUT
2531	007164	012703	001264			MOV	@INBUF, R3	;GET REPLY
2532	007170	122713	000060			CMPB	@'0', (R3)	;BRANCH IF 'NO' (0)
2533	007174	001405				BEQ	7S	
2534	007176	122713	000061			CMPB	@'1', (R3)	;CHECK IF 'YES' (1)
2535	007202	001362				BNE	6S	;ASK AGAIN IF NEITHER
2536	007204	111337	001127			MOVB	(R3), @NRZFLG	;SET INDICATOR
2537	007210				7S:			
2538								
2539	007210	052737	000100	177560	TYPHDR:	BIS	@100, @TKS	;SET KEYBOARD IE BIT
2540	007216	000004	014344			TYPE, L.HDR1		
2541	007222	116037	001132	014524		MOVB	DIGTAB(R0), @L.DRV	;SET DRIVE #
2542	007230	116137	001132	014536		MOVB	DIGTAB(R1), @L.SLV	;AND SLAVE #
2543	007236	112737	000071	014541		MOVB	@'9', @L.CHAN	;GET SLAVES CHANNEL TYPE
2544	007244	032765	010000	000026		BIT	@CH7, DT(R5)	
2545	007252	001403				BEQ	1S	
2546	007254	112737	000067	014541		MOVB	@'7', @L.CHAN	;SET 7 CHANNEL
2547	007262	000004	014457		1S:	TYPE, L.HDR2		
2548	007266	004767	176302			JSR	PC, SNPT	;GO PRINT SERIAL NUMBER
2549	007272	000004	014560			TYPE, L.HDR3		
2550	007276	012737	007336	001002		MOV	@TST001, @SCPADR	;SET 'SCOPE' ADDRESS FOR FIRST TEST
2551	007304	010500				MOV	R5, R0	
2552	007306	062700	000006			ADD	@FC, R0	;R0 CONTAINS ADDRESS OF FC REG
2553	007312	010501				MOV	R5, R1	
2554	007314	062701	000012			ADD	@DS, R1	;R1 CONTAINS ADDRESS OF DS REG
2555	007320	012703	004444			MOV	@TIMER, R3	;SET JUMP ADDRESS TO TIMER
2556	007324	105737	001124			TSTB	@PRGFLG	;BRANCH IF NOT SKEW TESTS
2557	007330	001402				BEQ	TST001	
2558	007332	000137	012764			JMP	@SKEWTST	

```

2559          SBTTL START OF TESTS
2560          ;TEST 001 - WRITE FROM BOT
2561          ;THIS TEST WILL MEASURE ACCELERATION DELAY REQUIRED TO
2562          ;MOVE THE TAPE APPROXIMATELY SEVEN (7) INCHES FORWARD
2563          ;FROM DEAD STOP BEFORE STARTING TO TRANSFER DATA.
2564
2565          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2566          007336 112737 000001 001122 TST001: MOVB #1, @TSTNUM ;SET TEST #
2567          007344 012702 007370      MOV #1$, R2 ;SET RETURN PC FROM TIMER
2568          007350 004767 175762      JSR PC, .REWIND ;REWIND SLAVE
2569          007354 102420      BVS 99$ ;BRANCH IF ERROR ON REWIND
2570          007356 004767 176036      JSR PC, WRITE ;GO SETUP WRITE COMMAND
2571          007362 004767 174766      JSR PC, TIMON ;TURN TIMER ON
2572          007366 005215      INC (R5) ;SET 'GO' BIT
2573
2574          007370 005765 000032      1$: TST TC(R5) ;BRANCH WHEN 'ACCL'=0
2575          007374 100002      BPL 2$
2576          007376 000163 004444      JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2577
2578          007402 004767 175624      2$: JSR PC, WAITRDY ;WAIT FOR COMMAND TO FINISH
2579          007406 102403      BVS 99$ ;BRANCH IF ERROR
2580          007410 004767 175066      JSR PC, TIMOK ;GO CHECK TIME
2581          007414 000401      BR 100$
2582          007416 104400      99$: HLT
2583          007420 104000      100$: SCOPE
2584
2585          ;TEST 002 - WRITE START
2586          ;THIS TST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2587          007422 112737 000002 001122 TST002: MOVB #2, @TSTNUM ;SET TEST # 2
2588          007430 004767 175764      JSR PC, WRITE ;INITIATE WRITE COMMAND
2589          007434 012702 007446      MOV #1$, R2 ;SET RETURN PC FROM TIMER
2590          007440 004767 174710      JSR PC, TIMON
2591          007444 005215      INC (R5) ;SET 'GO' BIT
2592
2593          007446 005765 000032      1$: TST TC(R5) ;BRANCH WHEN 'ACCL'=0
2594          007452 100002      BPL 2$
2595          007454 000163 004444      JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2596
2597          007460 004767 175546      2$: JSR PC, WAITRDY ;WAIT FOR READY
2598          007464 102403      BVS 99$ ;BRANCH IF ERROR
2599          007466 004767 175010      JSR PC, TIMOK ;GO CHECK TIME RECORDED
2600          007472 000401      BR 100$ ;EXIT VIA SCOPE
2601
2602          007474 104400      99$: HLT ;REPORT ERROR
2603          007476 104000      100$: SCOPE
2604
2605          ;TEST 003- WRITE SHUTDOWN
2606          ;THIS TEST MEASURES TIME FROM 'FC REG'=0 TO 'SWDN'=1.
2607          007500 112737 000003 001122 TST003: MOVB #3, @TSTNUM ;SET TEST#3
2608          007506 004767 175706      JSR PC, WRITE ;INITIATE WRITE COMMAND
2609          007512 005215      INC (R5) ;SET 'GO' BIT
2610
2611          007514 005710      1$: TST (R0) ;BRANCH WHEN WRITING FINISHED
2612          007516 001404      BEQ 2$
2613          007520 032711 040000      BIT #ERR, (R1) ;MONITOR ERROR BIT
2614          007524 001017      BNE 99$
    
```

```

2615 007526 000772          BR      1$
2616
2617 007530          2$:
2618 007530 004767 174620      JSR    PC,TIMON          ;TURN TIMER ON
2619 007534 010702          MOV    PC,R2            ;LOAD RETURN PC FROM TIMER
2620 007536 032711 000020      BIT    #SDWN,(R1)       ;BRANCH WHEN DS <SDWN> SETS
2621 007542 001002          BNE   4$
2622 007544 000163 004444      JMP    TIMER(R3)        ;GO TO TIMER & RETURN VIA R2
2623
2624 007550 004767 175456      4$:   JSR    PC,WAITRDY   ;WAIT FOR READY
2625 007554 102403          BVS   99$
2626 007556 004767 174720      JSR    PC,TIMOK        ;GO CHECK TIME RECORDED
2627 007562 000401          BR    100$
2628 007564 104400          99$:  HLT
2629 007566 104000          100$: SCOPE            ;REPORT ERROR
2630
2631
2632          ;TEST 004 - WRITE SETTLEDOWN
2633          ;THIS TEST MEASURES TIME FROM 'SDWN'=1 TO 'SDWN'=0.
2633 007570 112737 000004 001122  TST004: MOVB   #4,#TSTNUM
2634 007576 004767 175616      JSR    PC,WRITE
2635 007602 005215          INC    (R5)            ;SET 'GO' BIT
2636
2637 007604 005710          1$:   TST    (R0)        ;BRANCH WHEN WRITING FINISHED
2638 007606 001404          BEQ   2$
2639 007610 032711 040000      BIT    #ERR,(R1)       ;CHECK ERROR BIT
2640 007614 001026          BNE   99$
2641 007616 000772          BR    1$
2642
2643 007620 032711 000020      2$:   BIT    #SDWN,(R1)   ;WAIT FOR ASSERTION OF 'SDWN'
2644 007624 001004          BNE   3$
2645 007626 032711 040000      BIT    #ERR,(R1)       ;MONITOR ERROR BIT
2646 007632 001017          BNE   99$
2647 007634 000771          BR    2$
2648
2649 007636          3$:
2650 007636 004767 174512      JSR    PC,TIMON          ;TURN TIMER ON
2651 007642 010702          MOV    PC,R2            ;SET RETURN PC FROM TIMER
2652 007644 032711 000020      BIT    #SDWN,(R1)       ;BRANCH WHEN SDWN CLEARS
2653 007650 001402          BEQ   5$
2654 007652 000163 004444      JMP    TIMER(R3)        ;GO TO TIMER & RETURN VIA R2
2655
2656 007656 004767 175350      5$:   JSR    PC,WAITRDY   ;WAIT FOR READY
2657 007662 102403          BVS   99$
2658 007664 004767 174612      JSR    PC,TIMOK
2659 007670 000401          BR    100$
2660
2661 007672 104400          99$:  HLT
2662 007674 104000          100$: SCOPE
2663
2664          ;TEST 005 - READ FROM BOT
2665          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2665 007676 112737 000005 001122  TST005: MOVB   #5,#TSTNUM
2666 007704 004767 175426      JSR    PC,REWIND        ;SET TEST #5
2667          ;REWIND SLAVE
2668 007710 102422          BVS   99$              ;BRANCH IF ERROR ON REWIND
2669 007712 004767 175520      JSR    PC,READ
2670 007716 012702 007730      MOV    #1$,R2          ;SET RETURN PC FROM TIMER

```

```

2671 007722 004767 174426          JSR    PC,TIMON          ;TURN TIMER ON
2672 007726 005215          INC    (R5)             ;SET 'GO' BIT
2673
2674 007730 005765 000032          1$:   TST    TC(R5)          ;BRANCH WHEN 'ACCL' RESETS
2675 007734 100002          BPL    2$              ;
2676 007736 000163 004444          JMP    TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
2677
2678 007742 004767 175264          2$:   JSR    PC,WAITRDY     ;WAIT FOR READY
2679 007746 102403          BVS    99$            ;BRANCH IF ERROR
2680 007750 004767 174526          JSR    PC,TIMOK        ;CHECK RECORDED TIME
2681 007754 000401          BR     100$           ;
2682
2683 007756 104400          99$:   HLT
2684 007760 104000          100$:  SCOPE
2685
2686          ;TEST 006 - READ START
2687          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2688 007762 112737 000006 001122 1$T006: MOV#    #6,#TSTNUM          ;SET TEST #6
2689 007770 004767 175516          JSR    PC,WRT.BK       ;WRITE A RECORD & BACK SPACE
2690 007774 102422          BVS    99$            ;
2691 007776 004767 175434          JSR    PC,READ         ;
2692 010002 012702 010014          MOV    #1,R2           ;SET RETURN PC FROM TIMER
2693 010006 004767 174342          JSR    PC,TIMON        ;TURN TIMER ON
2694 010012 005215          INC    (R5)           ;SET 'GO' BIT
2695
2696 010014 005765 000032          1$:   TST    TC(R5)          ;BRANCH WHEN 'ACCL' RESETS
2697 010020 100002          BPL    2$              ;
2698 010022 000163 004444          JMP    TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
2699
2700 010026 004767 175200          2$:   JSR    PC,WAITRDY     ;
2701 010032 102403          BVS    99$            ;
2702 010034 004767 174442          JSR    PC,TIMOK        ;
2703 010040 000401          BR     100$           ;
2704
2705 010042 104400          99$:   HLT
2706 010044 104000          100$:  SCOPE
2707
2708          ;TEST 007 - READ SHUTDOWN
2709          ;THIS TEST MEASURES TIME FROM 'FC REG'=FRAME COUNT TO 'SDWN'=1.
2710 010046 112737 000007 001122 1$T007: MOV#    #7,#TSTNUM          ;SET TEST #7
2711 010054 004767 175432          JSR    PC,WRT.BK       ;WRITE A RECORD & BACK SPACE
2712 010060 102430          BVS    99$            ;BRANCH IF ERROR
2713 010062 004767 175350          JSR    PC,READ         ;
2714 010066 005215          INC    (R5)           ;SET 'GO' BIT
2715
2716 010070 022710 000400          1$:   CMP    #-FRMCNT,(R0)    ;WAIT FOR FRAME COUNT TO
2717 010074 001404          BEQ    2$              ;= # OF FRAMES WRITTEN
2718 010076 032711 040000          BIT    #ERR,(R1)       ;MONITOR ERROR BIT
2719 010102 001017          BNE    99$            ;
2720 010104 000771          BR     1$             ;
2721
2722 010106          2$:   JSR    PC,TIMON        ;TURN TIMER ON
2723 010106 004767 174242          MOV    PC,R2           ;SET RETURN PC FROM TIMER
2724 010112 010702          BIT    #SDWN,(R1)      ;BRANCH WHEN SDWN SETS
2725 010114 032711 000020          BNE    3$              ;
2726 010120 001002          3$:

```

```

2727 010122 000163 004444          JMP      TIMER(R3)          ;GO TO TIMER & RETURN VIA R2
2728
2729 010126 004767 175100          3$:     JSR      PC, WAITRDY
2730 010132 102403                    BVS     99$
2731 010134 004767 174342          JSR     PC, TIMOK
2732 010140 000401                    BR      100$
2733
2734 010142 104400          99$:    HLT
2735 010144 104000          100$:   SCOPE          ;REPORT ERROR
2736
2737          ;TEST 010 - READ SETTLEDOWN
2738          ;THIS TEST MEASURES TIME FROM 'SDWN'=1 TO 'SDWN'=0.
2739 010146 112737 000010 001122  TST010: MOVB   #10, @TSTNUM          ;SET TEST #10
2740 010154 012702 010232          MOV     #4$, R2          ;SET RETURN PC FROM TIMER
2741 010160 004767 175326          JSR     PC, WRT.BK       ;WRITE A RECORD & BACK SPACE
2742 010164 102436                    BVS     99$
2743 010166 004767 175244          JSR     PC, READ
2744 010172 005215                    INC     (R5)          ;SET 'GO' BIT
2745
2746 010174 105711          1$:     TSTB   (R1)          ;WAIT FOR READY
2747 010176 100404                    BMI     2$
2748 010200 032711 040000          BIT     #ERR, (R1)      ;BRANCH WHEN SET
2749 010204 001026                    BNE     99$            ;CHECK ERROR BIT
2750 010206 000772                    BR      1$
2751
2752 010210 032711 000020          2$:     BIT     #SDWN, (R1) ;WAIT FOR ASSERTION OF 'SDWN'
2753 010214 001004                    BNE     3$
2754 010216 032711 040000          BIT     #ERR, (R1)      ;MONITOR ERROR BIT
2755 010222 001017                    BNE     99$
2756 010224 000771                    BR      2$
2757
2758 010226          3$:
2759 010226 004767 174122          JSR     PC, TIMON          ;TURN TIMER ON
2760 010232 032765 000020 000012  4$:     BIT     #SDWN, DS(R5) ;WAIT FOR NEGATION OF SDWN
2761 010240 001402                    BEQ     5$
2762 010242 000163 004444          JMP     TIMER(R3)        ;GO TO TIMER & RETURN VIA R2
2763
2764 010246 004767 174760          5$:     JSR     PC, WAITRDY
2765 010252 102403                    BVS     99$
2766 010254 004767 174222          JSR     PC, TIMOK
2767 010260 000401                    BR      100$
2768
2769 010262 104400          99$:    HLT
2770 010264 104000          100$:   SCOPE
2771
2772
2773          ;TEST 011-READ REVERSE START
2774          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2775 010266 112737 000011 001122  TST011: MOVB   #11, @TSTNUM
2776 010274 012702 010332          MOV     #1$, R2          ;SET RETURN PC FROM TIMER
2777 010300 004767 175114          JSR     PC, WRITE        ;WRITE A RECORD
2778 010304 005215                    INC     (R5)            ;SET 'GO' BIT
2779 010306 004767 174720          JSR     PC, WAITRDY
2780 010312 102422                    BVS     99$
2781 010314 004767 174420          JSR     PC, DELAY        ;WAIT FOR TAPE MOTION TO STOP
2782 010320 004767 175130          JSR     PC, REVRD
    
```

```

2783 010324 004767 174024          JSR    PC,TIMON          ;TURN TIMER ON
2784 010330 005215                   INC    (R5)              ;SET 'GO' BIT
2785                                     ;
2786 010332 005765 000032          1$:   TST    TC(R5)          ;BRANCH WHEN 'ACCL' = 0
2787 010336 100002                   BPL    2$                ;
2788 010340 000163 004444          JMP    TIMER(R3)        ;GO TO TIMER & RETURN VIA R2
2789                                     ;
2790 010344 004767 174662          2$:   JSR    PC,WAITRDY     ;
2791 010350 102403                   BVS    99$              ;BRANCH IF ERROR
2792 010352 004767 174124          JSR    PC,TIMOK
2793 010356 000401                   BR     100$
2794                                     ;
2795 010360 104400          99$:   HLT
2796 010362 104000          100$:  SCOPE
2797                                     ;
2798                                     ;TEST 012-READ REVERSE SHUTDOWN
2799                                     ;THIS TEST MEASURES TIME FROM 'FC REG' = FRAME COUNT TO 'SDWN'=1.
2800 010364 112737 000012 001122  TST012: MOVB    #12,#TSTNUM
2801 010372 012702 010442          MOV    #3$,R2          ;SET RETURN PC FROM TIMER
2802 010376 004767 175016          JSR    PC,WRITE        ;WRITE A RECORD
2803 010402 005215                   INC    (R5)              ;SET 'GO' BIT
2804 010404 004767 174622          JSR    PC,WAITRDY
2805 010410 102427                   BVS    99$
2806 010412 004767 175036          JSR    PC,REVRD
2807 010416 005215                   INC    (R5)              ;SET 'GO' BIT
2808                                     ;
2809 010420 022710 000400          1$:   CMP    #-FRMCNT,(R0)  ;BRANCH WHEN FRAME COUNT
2810 010424 001404                   BEQ    2$                ;= # OF RECORD WRITTEN
2811 010426 032711 040000          BIT    #ERR,(R1)        ;MONITOR ERROR BIT IN 'DS' REG
2812 010432 001016                   BNE    99$
2813 010434 000771                   BR     1$
2814                                     ;
2815 010436                                     2$:
2816 010436 004767 173712          JSR    PC,TIMON          ;TURN TIMER ON
2817 010442 032711 000020          3$:   BIT    #SDWN,(R1)    ;BRANCH WHEN SDWN SETS
2818 010446 001002                   BNE    4$
2819 010450 000163 004444          JMP    TIMER(R3)        ;GO TO TIMER & RETURN VIA R2
2820                                     ;
2821 010454 004767 174552          4$:   JSR    PC,WAITRDY     ;WAIT FOR READY
2822 010460 102403                   BVS    99$
2823 010462 004767 174014          JSR    PC,TIMOK
2824 010466 000401                   BR     100$
2825                                     ;
2826 010470 104400          99$:   HLT
2827 010472 104000          100$:  SCOPE
2828                                     ;
2829                                     ;TEST 013-READ REVERSE SETTLEDOWN
2830                                     ;THIS TEST MEASURES TIME FROM 'SDWN'=1 TO 'SDWN'=0.
2831 010474 112737 000013 001122  TST013: MOVB    #13,#TSTNUM
2832 010502 012702 010566          MOV    #4$,R2          ;SET RETURN PC FROM TIMER
2833 010506 004767 174706          JSR    PC,WRITE        ;WRITE A RECORD
2834 010512 005215                   INC    (R5)              ;SET 'GO' BIT
2835 010514 004767 174512          JSR    PC,WAITRDY
2836 010520 102435                   BVS    99$
2837 010522 004767 174726          JSR    PC,REVRD
2838 010526 005215                   INC    (R5)              ;SET 'GO' BIT

```

```

2839
2840 010530 105711          1$:   TSTB   (R1)           ;BRANCH WHEN
2841 010532 100404          BMI     2$           ;READY SETS
2842 010534 032711 040000  BIT     #ERR, (R1)
2843 010540 001025          BNE     99$
2844 010542 000772          BR      1$
2845
2846 010544 032711 000020  2$:   BIT     #SDWN, (R1)
2847 010550 001004          BNE     3$
2848 010552 032711 040000  BIT     #ERR, (R1)
2849 010556 001016          BNE     99$
2850 010560 000771          BR      2$
2851
2852 010562
2853 010562 004767 173566  3$:   JSR     PC, TIMON           ;TURN TIMER ON
2854 010566 032711 000020  4$:   BIT     #SDWN, (R1)       ;BRANCH WHEN SWDN = 0
2855 010572 001402          BEQ     5$
2856 010574 000163 004444  JMP     TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
2857
2858 010600 004767 174426  5$:   JSR     PC, WAITRDY        ;WAIT FOR READY
2859 010604 102403          BVS     99$
2860 010606 004767 173670  JSR     PC, TIMOK
2861 010612 000401          BR      100$
2862
2863 010614 104400          99$:  HLT
2864 010616 104000          100$: SCOPE
2865
2866          ;REWIND DRIVE
2867 010620          A:
2868 010620 004767 174512  JSR     PC, .REWIND       ;REWIND SLAVE
2869 010624 102401          BVS     99$           ;BRANCH IF ERROR ON REWIND
2870 010626 102002          BVC     100$
2871 010630 104400          99$:  HLT
2872 010632 000772          BR      A
2873 010634          100$:
2874
2875          ;TEST 014-TURN AROUND DELAY (FORWARD-REVERSE)
2876          ;THIS TEST MEASURES TIME FROM 'GO'=1 (READ REVERSE) TO 'ACCL'=0
2877 010634 112737 000014 001122  TST014: MOVB   #14, #TSTNUM
2878 010642 012702 010674  MOV     #25, R2           ;SET RETURN PC FROM TIMER
2879 010646 004767 174546  JSR     PC, WRITE        ;WRITE A RECORD
2880 010652 005215          INC     (R5)           ;SET 'GO' BIT
2881 010654 004767 174352  JSR     PC, WAITRDY
2882 010660 102420          BVS     99$
2883
2884 010662 004767 174566  1$:   JSR     PC, REVRD        ;READ THE RECORD (REVERSE)
2885 010666 004767 173462  JSR     PC, TIMON        ;TURN TIMER ON
2886 010672 005215          INC     (R5)           ;SET 'GO' BIT
2887
2888 010674 005765 000032  2$:   TST     TC(R5)         ;WAIT FOR 'ACCL' = 0
2889 010700 100002          BPL     3$
2890 010702 000163 004444  JMP     TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
2891
2892 010706 004767 174320  3$:   JSR     PC, WAITRDY
2893 010712 102403          BVS     99$
2894 010714 004767 173562  JSR     PC, TIMOK

```

DZTUG-B TMO2/TE16 DRIVE FUNCTION TIMER  
DZTUGB.P11 15-JUL-77 12:52MACY11 30(1046) 15-JUL-77 12:55 PAGE 70  
START OF TESTS

SEQ 0070

```

2895 010720 000401          BR      100$
2896
2897 010722 104400          99$:  HLT
2898 010724 104000          100$: SCOPE
2899
2900          ;TEST 015- TURN AROUND DELAY (REVERSE-FORWARD)
2901          ;THIS TEST MEASURES TIME FROM 'GO'=1 (READ) TO 'ACCL'=0.
2902 010726 112737 000015 001122 †ST015: MOVB  #15, @STNUM
2903 010734 012702 011002          MOV  #2$, R2          ;SET RETURN PC FROM TIMER
2904 010740 004767 174454          JSR  PC, WRITE        ;WRITE A RECORD
2905 010744 005215          INC  (R5)            ;SET 'GO' BIT
2906 010746 004767 174260          JSR  PC, WAITRDY     ;WAIT FOR READY
2907 010752 102426          BVS  99$
2908 010754 004767 174474          JSR  PC, REVRD       ;READ A RECORD IN THE
2909 010760 005215          INC  (R5)            ;SET 'GO' BIT
2910
2911 010762 004767 174244          JSR  PC, WAITRDY
2912 010766 102420          BVS  99$
2913
2914 010770 004767 174442          1$:  JSR  PC, READ      ;READ RECORD FORWARD
2915 010774 004767 173354          JSR  PC, TIMON       ;TURN TIMER ON
2916 011000 005215          INC  (R5)            ;SET 'GO' BIT
2917
2918 011002 005765 000032          2$:  TST  TC(R5)       ;WAIT FOR 'ACCL' = 0
2919 011006 100002          BPL  3$
2920 011010 000163 004444          JMP  TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
2921
2922 011014 004767 174212          3$:  JSR  PC, WAITRDY
2923 011020 102403          BVS  99$
2924 011022 004767 173454          JSR  PC, TIMOK
2925 011026 000401          BR      100$
2926
2927 011030 104400          99$:  HLT
2928 011032 104000          100$: SCOPE
2929
2930          ;TEST 016-GAP SIZE (STOP HALF)
2931 011034 112737 000016 001122 †ST016: MOVB  #16, @STNUM
2932 011042 012702 011100          MOV  #1$, R2          ;SET RETURN PC FROM TIMER
2933 011046 004767 174346          JSR  PC, WRITE        ;WRITE A RECORD
2934 011052 005215          INC  (R5)            ;SET 'GO' BIT
2935 011054 004767 174152          JSR  PC, WAITRDY
2936 011060 102421          BVS  99$
2937 011062 004767 173652          JSR  PC, DELAY       ;DELAY 350 MS
2938 011066 004767 174362          JSR  PC, REVRD       ;READ REVERSE RECORD
2939 011072 004767 173256          JSR  PC, TIMON       ;TURN TIMER ON
2940 011076 005215          INC  (R5)            ;SET 'GO' BIT
2941
2942 011100 005710          1$:  TST  (R0)           ;WAIT FOR FRAME COUNT > 0
2943 011102 001002          BNE  2$
2944 011104 000163 004444          JMP  TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
2945
2946 011110 004767 174116          2$:  JSR  PC, WAITRDY   ;WAIT FOR READY BIT TO SET
2947 011114 102403          BVS  99$
2948 011116 004767 173360          JSR  PC, TIMOK       ;CHECK TIME
2949 011122 000401          BR      100$
2950

```



```

2951 011124 104400          99$:   HLT
2952 011126 104000          100$:  SCOPE
2953
2954
2955
2956
2957 011130 112737 000017 001122 ;TEST 017-GAP SIZE (START HALF)
2958 011136 012702 011210 †TST017: MOVB #17,0#TSTNUM
2959 011142 004767 174252      MOV #15,R2 ;SET RETURN PC FROM TIMER
2960 011146 005215          JSR PC,WRITE ;WRITE A RECORD
2961 011150 004767 174056      INC (R5) ;SET 'GO' BIT
2962 011154 102427          JSR PC,WAITRDY ;WAIT FOR READY
2963 011156 004767 174272      BVS 99$
2964 011162 005215          JSR PC,REVRD ;READ REVERSE THE RECORD
2965 011164 004767 174042      INC (R5) ;SET 'GO' BIT
2966 011170 102421          JSR PC,WAITRDY ;WAIT FOR READY
2967 011172 004767 173542      BVS 99$ ;BRANCH ON ERROR
2968 011176 004767 174234      JSR PC,DELAY ;WAIT FOR TAPE MOTION TO STOP
2969 011202 004767 173146      JSR PC,READ ;READ RECORD
2970 011206 005215          JSR PC,TIMON ;TURN TIMER ON
2971          INC (R5) ;SET 'GO' BIT
2972
2973
2974 011210 005710          1$:   TST (R0) ;WAIT FOR FRAME COUNT > 0
2975 011212 001002          BNE 2$
2976 011214 000163 004444      JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2977
2978
2979 011220 004767 174006      2$:   JSR PC,WAITRDY ;WAIT FOR READY
2980 011224 102403          BVS 99$
2981 011226 004767 173250      JSR PC,TIMOK ;CHECK TIME
2982 011232 000401          BR 100$
2983
2984
2985 011234 104400          99$:   HLT
2986 011236 104000          100$:  SCOPE
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006

```

```

;TEST 020- GAP SIZE (INTERRECORD)
;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'FC REG' >0.

```

```

†TST020: MOVB #20,0#TSTNUM
          MOV #15,R2 ;SET RETURN PC FROM TIMER
          JSR PC,WRITE ;WRITE A RECORD
          INC (R5) ;SET 'GO' BIT
          JSR PC,WAITRDY ;WAIT FOR READY
          BVS 99$
          JSR PC,WRITE ;WRITE SECOND RECORD
          INC (R5) ;SET 'GO' BIT
          JSR PC,WAITRDY ;WAIT FOR READY
          BVS 99$
          JSR PC,REVRD ;READ REVERSE SECOND RECORD
          INC (R5) ;SET 'GO' BIT
          JSR PC,WAITRDY ;WAIT FOR READY
          BVS 99$
          JSR PC,REVRD ;READ REVERSE FIRST RECORD
          JSR PC,TIMON ;TURN TIMER ON
          INC (R5) ;SET 'GO' BIT
          TST (R0) ;WAIT FOR FRAME COUNT > 0
          BNE 2$
          JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
          JSR PC,WAITRDY ;WAIT FOR READY

```

3007 011344 102403  
3008 011346 004767 173130  
3009 011352 000401  
3010  
3011 011354 104400  
3012 011356 104000  
3013  
3014  
3015  
3016  
3017  
3018  
3019  
3020  
3021  
3022  
3023  
3024  
3025

BVS 99\$  
JSR PC,TIMOK  
BR 100\$  
99\$: HLT  
100\$: SCOPE

```

:TEST 021- GAP CONSISTANCY
:THIS TEST MEASURES TIME FROM 'GO'=1 TO 'FC REG' > 0.
:THE TEST REWINDS THE TAPE, WRITES 17 RECORDS WITH A DELAY FROM 1-16 MS
:BEFORE EACH WRITE COMMAND. AFTER THE 17. RECORDS ARE WRITTEN THE
:PROGRAM READ REVERSES 16 RECORDS. AT THIS POINT THE TAPE IS STOPPED BE-
:TWEEN THE FIRST AND SECOND RECORD. A READ COMMAND IS EXECUTED TO READ
:THE 16 RECORDS WITH THE TIME BETWEEN GO=1 TO FC > 0 STORED IN 'CAPTBL'
:FOR EACH RECORD READ. AFTER 16 RECORDS HAVE BEEN READ THE TIME IS VER-
:IFIED FOR EACH READ. AFTER ALL RECORD TIMES ARE VERIFIED THEY ARE AVER-
:AGED AND PLACED IN THE 'ATIMTBL' (BY SCOPE). THE ABOVE PROCESS IS RE-
:PEATED FOR EACH ITERATION.
    
```

3026 011360 112737 000021 001122  
3027 011366 012702 011524  
3028 011372 004767 173740  
3029 011376 102530  
3030 011400 005067 167510  
3031 011404 012700 000021  
3032 011410 004767 174004  
3033 011414 005215  
3034 011416 004767 173610  
3035 011422 102516  
3036 011424 004767 173340  
3037 011430 062767 000022 167456  
3038 011436 005300  
3039 011440 001363  
3040  
3041 011442 012700 000021  
3042 011446 004767 174002  
3043 011452 005215  
3044 011454 004767 173552  
3045 011460 102477  
3046 011462 005300  
3047 011464 001370  
3048  
3049 011466 012700 000020  
3050 011472 012701 001054  
3051 011476 004767 173734  
3052 011502 005215  
3053  
3054 011504 004767 173522  
3055 011510 102463  
3056 011512 004767 173720  
3057 011516 004767 172632  
3058 011522 005215  
3059  
3060 011524 005765 000006  
3061 011530 001002  
3062 011532 000163 004444

```

TST021: MOV      #21, R2          ;SET RETURN PC FROM TIMER
        MOV      #45, R2         ;REWIND SLAVE
        JSR      PC, REWIND      ;BRANCH IF ERROR ON REWIND
        BVS     99$             ;CLEAR VARIABLE DELAY TIME
        CLR      DELTIM          ;SET # OF RECORDS TO WRITE
        MOV      #17, R0         ;WRITE 17. RECORDS
1$:     JSR      PC, WRITE        ;SET 'GO' BIT
        INC      (R5)           ;WAIT FOR READY
        JSR      PC, WAITRDY
        BVS     99$
        JSR      PC, DELAYV      ;DELAY BEFORE WRITING NEXT REC.
        ADD      #18., DELTIM    ;SET NEXT DELAY TIME
        DEC      R0             ;DECREMENT RECORDS WRITTEN COUNT
        BNE     1$
        MOV      #17, R0        ;SET # OF RECS. TO REVERSE READ
2$:     JSR      PC, REVRD       ;REVERSE READ 17. RECORDS
        INC      (R5)           ;SET 'GO' BIT
        JSR      PC, WAITRDY    ;WAIT FOR READY
        BVS     99$            ;DECREMENT RECORD COUNT
        DEC      R0
        BNE     2$
        MOV      #16, R0        ;SET # OF RECORDS TO READ
        MOV      #CAPTBL, R1     ;SET PTR TO GAP TABLE FOR TEST
        JSR      PC, READ        ;READ A RECORD
        INC      (R5)           ;SET 'GO' BIT
        JSR      PC, WAITRDY    ;WAIT FOR READY
        BVS     99$
        JSR      PC, READ        ;READ NEXT RECORD
        JSR      PC, TIMON      ;TURN TIMER ON
        INC      (R5)           ;SET 'GO' BIT
        TST     FC(R5)          ;WAIT FOR FRAME COUNT > 0
        BNE     5$
        JMP     TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
    
```

```

3063
3064 011536 004767 173470      5$:  JSR      PC, WAITRDY      ;WAIT FOR READY
3065 011542 102446                BVS      99$
3066 011544 010421                MOV      R4, (R1)+      ;STORE TIME IN GAP TBL
3067 011546 005300                DEC      R0              ;DECREMENT # OF RECORDS READ
3068 011550 001355                BNE      3$
3069
3070 011552 105037 001120        CLR      2#GAP          ;SET GAP # 0
3071 011556 012700 000020        MOV      #16., R0
3072 011562 012701 001054        MOV      #GAP TBL, R1
3073
3074 011566 012104                6$:  MOV      (R1)+, R4      ;GET GAP TICK COUNT
3075 011570 004767 173016        JSR      PC, GAP OK     ;CHECK TIME
3076 011574 105237 001120        INCB    2#GAP          ;INCREMENT GAP #
3077 011600 122737 000020 001120  CMPB    #16., 2#GAP     ;BRANCH IF ALL GAPS NOT CHECKED
3078 011606 001367                BNE      6$
3079
3080 011610 012700 000020        MOV      #16., R0      ;SETUP TO AVERAGE GAP SIZES
3081 011614 012701 001054        MOV      #GAP TBL, R1  ;SET PTR TO TABLE
3082 011620 005002                CLR      R2            ;CLEAR 'SUM' REGISTERS
3083 011622 005003                CLR      R3
3084 011624 062102                7$:  ADD      (R1)+, R2      ;ADD ALL GAP SIZES TOGETHER
3085 011626 005503                ADC      R3
3086 011630 005300                DEC      R0
3087 011632 001374                BNE      7$
3088 011634 012700 000004        MOV      #4, R0        ;NOW DIVIDE BY 16.
3089 011640 006203                8$:  ASR      R3            ;BY SHIFTING 4 PLACES RIGHT
3090 011642 006002                ROR      R2
3091 011644 005300                DEC      R0
3092 011646 001374                BNE      8$
3093 011650 010204                MOV      R2, R4        ;MOVE AVERAGED TIMES TO R4
3094 011652 004767 172624        JSR      PC, TIMOK     ;CHECK AVERAGED TIMES
3095 011656 000401                BR      100$
3096
3097 011660 104400                99$:  HLT
3098 011662 104000                100$: SCOPE
3099
3100 ;TEST 022-DUMMY TEST
3101 ;THIS TEST MEASURES NOTHING
3102 011664 112737 000022 001122  TST02: MOVB    #22, 2#TSTNUM
3103
3104 ;TEST 023-DATA TIME (200BPI)
3105 ;THIS TEST MEASURES TIME FROM 'FC REG' CHANGES TO 'RDY'=1.
3106 011672 112737 000023 001122  TST023: MOVB   #23, 2#TSTNUM
3107 011700 012702 011752                MOV      #3$, R2      ;SET RETURN PC FROM TIMER
3108 011704 004767 173426                JSR      PC, REWIND    ;REWIND SLAVE
3109 011710 102437                BVS      99$          ;BRANCH IF ERROR ON REWIND
3110 011712 004367 173636                JSR      R3, TMCMD     ;WRITE 800 WORD RECORD
3111 011716 015700                .WORD   WTBUF         ;SET WRITE BUFFER ADDRESS
3112 011720 176340                .WORD   -800.         ;WORD COUNT
3113 011722 174700                .WORD   -1600.        ;FRAME COUNT
3114 011724 000060                .WORD   WFWD          ;WRITE COMMAND
3115 011726 005215                INC      (R5)         ;SET 'GO' BIT
3116
3117 011730 022710 174700                1$:  CMP      #-1600., (R0)  ;WAIT FOR FRAME COUNT TO CHANGE
3118 011734 001004                BNE      2$

```

3119	011736	032711	040000		BIT	#ERR, (R1)		;MONITOR ERROR BIT
3120	011742	001022			BNE	99\$		
3121	011744	000771			BR	1\$		
3122								
3123	011746			2\$:	JSR	PC, TIMON		;TURN TIMER ON
3124	011746	004767	172402		TSTB	(R1)		;WAIT FOR READY TO SET
3125	011752	105711		3\$:	BMI	4\$		
3126	011754	100402			JMP	TIMER(R3)		;GO TO TIMER & RETURN VIA R2
3127	011756	000163	004444		MOV	#3, R0		;SET TO DIVIDE BY 8
3128	011762	012700	000003	4\$:	ASR	R4		;BY SHIFTING RIGHT 3 PLACES
3129	011766	006204		5\$:	DEC	R0		
3130	011770	005300			BNE	5\$		
3131	011772	001375			JSR	PC, WAITRDY		
3132	011774	004767	173232		BVS	99\$		
3133	012000	102403			JSR	PC, TIMOK		;CHECK TIME
3134	012002	004767	172474		BR	100\$		
3135	012006	000401						
3136								
3137	012010	104400		99\$:	HLT			
3138	012012	104000		100\$:	SCOPE			
3139								
3140								
3141	012014	112737	000024	001122	:TEST 024-DATA TIME (556BPI)			
3142	012022	012702	012102		TST024: MOV	#24, #TSTNUM		
3143	012026	004767	173304		MOV	#3\$, R2		;SET RETURN PC FROM TIMER
3144	012032	102442			JSR	PC, .REWIND		;REWIND SLAVE
3145	012034	052765	000700	000032	BVS	99\$		;BRANCH IF ERROR ON REWIND
3146	012042	004367	173506		BIS	#BPI556+NORM11, TC(R5)		;LOAD TAPE CONTROL REGISTER
3147	012046	015700			JSR	R3, TMCMD		;WRITE 2224. WORD RECORD
3148	012050	173520			.WORD	WTBUF		
3149	012052	167240			.WORD	-2224.		
3150	012054	000060			.WORD	-4448.		
3151	012056	005215			.WORD	WFWD		
3152					INC	(R5)		;SET 'GO' BIT
3153	012060	022710	167240	1\$:	CMP	#-4448., (R0)		;BRANCH WHEN WRITING BEGINS
3154	012064	001004			BNE	2\$		
3155	012066	032711	040000		BIT	#ERR, (R1)		;MONITOR ERROR BIT
3156	012072	001022			BNE	99\$		
3157	012074	000771			BR	1\$		
3158								
3159	012076			2\$:	JSR	PC, TIMON		;TURN TIMER ON
3160	012076	004767	172252		TSTB	(R1)		;BRANCH WHEN READY SETS
3161	012102	105711		3\$:	BMI	4\$		
3162	012104	100402			JMP	TIMER(R3)		;GO TO TIMER & RETURN VIA R2
3163	012106	000163	004444		MOV	#3, R0		;SET SHIFT COUNT
3164				4\$:	ASR	R4		
3165	012112	012700	000003	5\$:	DEC	R0		
3166	012116	006204			BNE	5\$		
3167	012120	005300			JSR	PC, WAITRDY		
3168	012122	001375			BVS	99\$		
3169	012124	004767	173102		JSR	PC, TIMOK		;CHECK TIME
3170	012130	102403			BR	100\$		
3171	012132	004767	172344					
3172	012136	000401						
3173								
3174	012140	104400		99\$:	HLT			

```

3175 012142 104000      100$: SCOPE
3176
3177
3178 012144 112737 000025 001122 :TEST 025-DATA TIME (800BPI)
3179 012152 012702 012232          TST025: MOVB #025,2#TSTNUM
3180 012156 004767 173154          MOV #35,R2 ;SET RETURN PC FROM TIMER
3181 012162 102442          JSR PC,REWIND ;REWIND SLAVE
3182 012164 052765 001300 000032 BVS 99$ ;BRANCH IF ERROR ON REWIND
3183 012172 004367 173356          BIS #8PI800+NORM11,TC(R5) ;SET 800 BPI
3184 012176 015700          JSR R3,TMCMO ;WRITE 3200. WORD RECORD
3185 012200 171600          .WORD WTBUF
3186 012202 163400          .WORD -3200.
3187 012204 000060          .WORD -6400.
3188 012206 005215          .WORD WFWO
3189          INC (R5) ;SET 'GO' BIT
3190 012210 022710 163400      1$: CMP #-6400.,(R0) ;WAIT FOR WRITING TO START
3191 012214 001004          BNE 2$
3192 012216 032711 040000          BIT #ERR,(R1) ;MONITOR ERROR BIT
3193 012222 001022          BNE 99$
3194 012224 000771          BR 1$
3195
3196 012226          2$:
3197 012226 004767 172122          JSR PC,TIMON ;TURN TIMER ON
3198 012232 105711          3$: TSTB (R1) ;BRANCH WHEN READY SETS
3199 012234 100402          BMI 4$
3200 012236 000163 004444          JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3201
3202 012242 012700 000003          4$: MOV #3,R0 ;SET SHIFT COUNT
3203 012246 006204          5$: ASR R4
3204 012250 005300          DEC R0
3205 012252 001375          BNE 5$
3206 012254 004767 172752          JSR PC,WAITRDY
3207 012260 102403          BVS 99$
3208 012262 004767 172214          JSR PC,TIMOK ;CHECK TIME
3209 012266 000401          BR 100$
3210
3211 012270 104400          99$: HLT
3212 012272 104000          100$: SCOPE
3213
3214
3215 012274 112737 000026 001122 :TEST 026-DATA TIME (1600BPI)
3216 012302 105737 001127          TST026: MOVB #026,2#TSTNUM
3217 012306 001046          TSTB 2#NRZFLG ;BRANCH IF DRIVE 'NRZ ONLY'
3218 012310 012702 012370          BNE TST027
3219 012314 004767 173016          MOV #35,R2 ;SET RETURN PC FROM TIMER
3220 012320 102437          JSR PC,REWIND ;REWIND SLAVE
3221 012322 052765 002300 000032 BVS 99$ ;BRANCH IF ERROR ON REWIND
3222 012330 004367 173220          BIS #PE1600+NORM11,TC(R5) ;SET 1600 BPI
3223 012334 015700          JSR R3,TMCMO ;WRITE 3200. WORD RECORD
3224 012336 171600          .WORD WTBUF
3225 012340 163400          .WORD -3200.
3226 012342 000060          .WORD -6400.
3227 012344 005215          .WORD WFWO
3228          INC (R5) ;SET 'GO' BIT
3229 012346 022710 163400      1$: CMP #-6400.,(R0) ;BRANCH WHEN WRITING STARTS
3230 012352 001004          BNE 2$

```

```

3231 012354 032711 040000          BIT      #ERR, (R1)          ;MONITOR ERROR BIT
3232 012360 001017                    BNE      99$
3233 012362 000771                    BR       1$
3234
3235 012364                    2$:
3236 012364 004767 171764          JSR      PC, TIMON          ;TURN TIMER ON
3237 012370 105711                    TSTB    (R1)              ;BRANCH WHEN READY SETS
3238 012372 100402                    BMI     4$
3239 012374 000163 004444          JMP      TIMER(R3)        ;GO TO TIMER & RETURN VIA R2
3240
3241 012400 006204                    4$:  ASR      R4              ;DIVIDE TIME BY 4
3242 012402 006204                    ASR      R4
3243 012404 004767 172622          JSR      PC, WAITRDY
3244 012410 102403                    BVS     99$
3245 012412 004767 172064          JSR      PC, TIMOK        ;CHECK TIME
3246 012416 000401                    BR       100$
3247
3248 012420 104400                    99$:  HLT
3249 012422 104000                    100$: SCOPE
3250
3251                    ;TEST 027-ERASE
3252                    ;THIS TST MEASURES TIME FROM 'GO'=1 TO 'RDY'=1.
3253 012424 112737 000027 001122  TST027: MOVB   #27, @TSTNUM
3254 012432 012702 012460          MOV      #1$, R2          ;SET RETURN PC FROM TIMER
3255 012436 004337 005554          JSR      R3, @TMCMD
3256 012442 000000                    .WORD   0
3257 012444 000000                    .WORD   0
3258 012446 000000                    .WORD   0
3259 012450 000024                    .WORD   ERASE
3260 012452 004767 171676          JSR      PC, TIMON          ;TURN TIMER ON
3261 012456 005215                    INC      (R5)              ;SET 'GO' BIT
3262
3263 012460 105711                    1$:  TSTB    (R1)              ;BRANCH WHEN READY SETS
3264 012462 100402                    BMI     2$
3265 012464 000163 004444          JMP      TIMER(R3)        ;GO TO TIMER & RETURN VIA R2
3266
3267 012470 004767 172536                    2$:  JSR      PC, WAITRDY
3268 012474 102403                    BVS     99$
3269 012476 004767 172000          JSR      PC, TIMOK
3270 012502 000401                    BR       100$
3271
3272 012504 104400                    99$:  HLT
3273 012506 104000                    100$: SCOPE
3274
3275                    ;TEST-030 TAPE MARK
3276                    ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'RDY'=1.
3277 012510 112737 000030 001122  TST030: MOVB   #30, @TSTNUM
3278 012516 012702 012560          MOV      #1$, R2          ;SET RETURN PC FROM TIMER
3279 012522 004767 172672          JSR      PC, WRITE        ;WRITE A RECORD
3280 012526 005215                    INC      (R5)              ;SET 'GO' BIT
3281 012530 004767 172476          JSR      PC, WAITRDY
3282 012534 102423                    BVS     99$
3283 012536 004337 005554          JSR      R3, @TMCMD
3284 012542 000000                    .WORD   0
3285 012544 000000                    .WORD   0
3286 012546 000000                    .WORD   0

```

3287	012550	000026		WORD	WFMK	
3288	012552	004767	171576	JSR	PC, TIMON	;TURN TIMER ON
3289	012556	005215		INC	(R5)	;SET 'GO' BIT
3290						
3291	012560	105711		15: TSTB	(R1)	;BRANCH WHEN READY SETS
3292	012562	100402		BMI	25	
3293	012564	000163	004444	JMP	TIMER(R3)	;GO TO TIMER & RETURN VIA R2
3294						
3295	012570	004767	172436	25: JSR	PC, WAITRDY	
3296	012574	102403		BVS	99\$	
3297	012576	004767	171700	JSR	PC, TIMOK	
3298	012602	000401		BR	100\$	
3299						
3300	012604	104400		99\$: HLT		
3301	012606			100\$:		
3302	012606	004767	172524	JSR	PC, .REWIND	;REWIND SLAVE
3303	012612	102774		BVS	99\$	;BRANCH IF ERROR ON REWIND
3304	012614	104000		SCOPE		
3305						

DZTUG-B TMO2/TE16 DRIVE FUNCTION TIMER  
DZTUGB.P11 15-JUL-77 12:52

MACY11 30(1046) 15-JUL-77 12:55 PAGE 78  
START OF TESTS

SEQ 0078

3306	012616	012700	000012		FINISH: MOV	#10.,R0		;SET LINE FEED COUNT
3307	012622	000004	001374		1S: TYPE,CRLF			
3308	012626	005300			DEC	R0		
3309	012630	001374			BNE	1S		
3310	012632	032777	000100	166140	BIT	#SW06,2SWR		
3311	012640	001410			BEQ	2S		
3312	012642	113700	001004		MOVB	@DRVNUM,R0		
3313	012646	113701	001005		MOVB	@SLVNUM,R1		
3314	012652	113702	001006		MOVB	@SLVPTR,R2		
3315	012656	000137	007210		JMP	@TYPHDR		
3316	012662	105237	001005		2S: INCB	@SLVNUM		;SET NEXT SLAVE #
3317	012666	005237	001006		INC	@SLVPTR		;AND ITS POINTER
3318	012672	122737	000010	001005	CMPB	#8.,@SLVNUM		;BRANCH IF LAST SLAVE (7)
3319	012700	001402			BEQ	3S		
3320	012702	000137	006764		JMP	@BEGIN		;BEGIN TEST ON NEXT SLAVE
3321	012706	105037	001005		3S: CLRB	@SLVNUM		;SET SLAVE #0
3322	012712	105237	001004		INCB	@DRVNUM		;AND INCREMENT DRIVE #
3323	012716	122737	000010	001004	CMPB	#8.,@DRVNUM		;AND CHECK IF LAST DRIVE
3324	012724	001402			BEQ	END		
3325	012726	000137	006764		JMP	@BEGIN		
3326								
3327	012732	105737	001125		END: TSTB	@UNTFND		;BRANCH IF A UNIT WAS FOUND
3328	012736	001004			BNE	1S		
3329	012740	000004	014047		TYPE,E.UNIT			
3330	012744	000137	005724		JMP	@INIT		
3331	012750	000000			1S: HALT			
3332	012752	004767	167012		JSR	PC,CKSWR		;CHECK FOR CNTL G
3333	012756	000005			RESET			
3334	012760	000137	005724		JMP	@INIT		;RESTART



```

3335 ;SKEW TAPE TIMING TESTS
3336 ;THE FOLLOWING TESTS REQUIRE A SPECIALLY WRITTEN 800 BPI SKEW TAPE
3337 012764 012737 012772 001002 SKEWTST:MOV #TST031,#SCPADR ;SET SCOPE POINTER
3338
3339 ;TEST 031- SKEW TAPE SPEED TEST-FORWARD
3340 ;THIS TEST READS 32" OF TAPE (26400.-800. = 25600. FRAMES), THEN
3341 ;DIVIDES TIME BY 32. TO GET TIME TO READ 1" (800. FRAMES) OF TAPE.
3342 012772 112737 000031 001122 TST031: MOV #31,#TSTNUM
3343 013000 012702 013056 MOV #25,R2 ;SET RETURN PC FROM TIMER
3344 013004 004767 172326 JSR PC,REWIND ;REWIND SLAVE
3345 013010 102441 BVS 99$ ;BRANCH IF ERROR ON REWIND
3346 013012 052765 001300 000032 BIS #BPI800+NORM11,TC(R5) ;SET 800 BPI
3347 013020 052765 000010 000010 BIS #BAI,CS2(R5) ;INHIBIT BUS ADDRESS INCREMENT
3348 013026 004337 005554 JSR R3,#TMCMD ;READ 32" OF TAPE-FORWARD
3349 013032 015700 .WORD R0BUF
3350 013034 177777 .WORD -1.
3351 013036 063440 10$: .WORD 26400. ;FRAME COUNT
3352 013040 000070 .WORD R0FWD
3353 013042 005215 INC (R5) ;SET 'GO' BIT
3354
3355 013044 022710 001440 1$: CMP #800.,(R0) ;WAIT FOR FIRST 800 FRAMES
3356 013050 101375 BHI 1$ ;TO BE READ
3357
3358 013052 004767 171276 JSR PC,TIMON ;TURN TIMER ON
3359 013056 023710 013036 2$: CMP #10$, (R0) ;WAIT FOR READING TO FINISH
3360 013062 103402 BLO 3$
3361 013064 000163 004444 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3362
3363 013070 012700 000005 3$: MOV #5,R0 ;DIVIDE TIME BY 32.
3364 013074 006204 4$: ASR R4
3365 013076 005300 DEC R0
3366 013100 001375 BNE 4$
3367 013102 004767 172066 JSR PC,RHINIT ;INIT DRIVE
3368 013106 004767 171370 JSR PC,TIMOK ;CHECK TIME
3369 013112 000401 BR 100$
3370
3371 013114 104400 99$: HLT
3372 013116 104000 100$: SCOPE
3373
3374 ;TEST 032-SKEW TAPE SPEED TEST-REVERSE
3375 ;THIS TEST READS FORWARD 40" (32000. FRAMES) OF TAPE, THEN READS REVERSE
3376 ;32" (26400.-800. = 25600. FRAMES) OF TAPE. THE TIME IS THEN DIVIDED BY
3377 ;32. TO GET TIME TO READ 1" (800. FRAMES) OF TAPE.
3378 013120 112737 000032 001122 TST032: MOV #32,#TSTNUM
3379 013126 012702 013254 MOV #35,R2 ;SET RETURN PC FROM TIMER
3380 013132 004767 172200 JSR PC,REWIND ;REWIND SLAVE
3381 013136 102465 BVS 99$ ;BRANCH IF ERROR ON REWIND
3382 013140 052765 001300 000032 BIS #BPI800+NORM11,TC(R5)
3383 013146 052765 000010 000010 BIS #BAI,CS2(R5)
3384 013154 004337 005554 JSR R3,#TMCMD ;READ FORWARD 32000. FRAMES
3385 013160 015700 .WORD R0BUF
3386 013162 177777 .WORD -1. ;WORD COUNT
3387 013164 076400 10$: .WORD 32000. ;FRAME COUNT
3388 013166 000070 .WORD R0FWD ;READ FORWARD
3389 013170 005215 INC (R5) ;SET 'GO' BIT
3390

```

3391	013172	023710	013164	15:	CMP	#105,(R0)	
3392	013176	101375			BHI	15	
3393							
3394	013200	004767	171770		JSR	PC,RHINIT	; INIT DRIVE
3395	013204	004767	171530		JSR	PC,DELAY	; WAIT FOR TAPE MOTION TO STOP
3396	013210	052765	001300	000032	BIS	#BPI800+NORM11,TC(R5)	; SET 800 BPI
3397	013216	052765	000010	000010	BIS	#BAI,CS2(R5)	; INHIBIT BUS ADDRESS INCREMENT
3398	013224	004337	005554		JSR	R3,#TMCMD	; READ REVERSE 32" OF TAPE
3399	013230	015700			.WORD	RDBUF	; READ BUFFER
3400	013232	177777			.WORD	-1.	; WORD COUNT
3401	013234	063440		115:	.WORD	26400.	; FRAME COUNT
3402	013236	000076			.WORD	RDREV	; READ REVERSE
3403	013240	005215			INC	(R5)	; SET 'GO' BIT
3404							
3405	013242	022710	001440	25:	CMP	#800.,(R0)	; WAIT FOR FIRST 800 FRAMES
3406	013246	101375			BHI	25	; TO BE READ
3407							
3408	013250	004767	171100		JSR	PC,TIMON	; TURN TIMER ON
3409	013254	023710	013234	35:	CMP	#115,(R0)	; WAIT FOR ALL FRAMES TO BE READ
3410	013260	103402			BLO	45	
3411	013262	000163	004444		JMP	TIMER(R3)	; GO TO TIMER & RETURN VIA R2
3412							
3413	013266	012700	000005	45:	MOV	#5,R0	; DIVIDE TIME BY 32.
3414	013272	006204		55:	ASR	R4	
3415	013274	005300			RO	R0	
3416	013276	001375			BNE	55	
3417	013300	004767	171670		JSR	PC,RHINIT	
3418	013304	004767	171172		JSR	PC,TIMOK	
3419	013310	000401			BR	1005	
3420							
3421	013312	104400		995:	HLT		
3422	013314			1005:			
3423	013314	004767	172016		JSR	PC,.REWIND	; REWIND SLAVE
3424	013320	102774			BVS	995	; BRANCH IF ERROR ON REWIND
3425	013322	104000			SCOPE		
3426							
3427	013324	000137	012616		JMP	#FINISH	
3428							
3429							
3430							

3431				
3432				
3433	013330	005015	042524	033061
3434	013336	042040	044522	042526
3435	013344	043040	047125	052103
3436	013352	047511	020116	044524
3437	013360	042515	020122	042050
3438	013366	052132	043525	041055
3439	013374	000051		
3440	013376	005015	054524	042520
3441	013404	043040	051111	052123
3442	013412	040440	042104	042522
3443	013420	051523	047440	020106
3444	013426	047503	052116	047522
3445	013434	046114	051105	020040
3446	013442	000		
3447	013443	124	050131	020105
3448	013450	046524	031060	042040
3449	013456	044522	042526	021440
3450	013464	051447	052040	020117
3451	013472	042502	052040	051505
3452	013500	042524	020104	000
3453	013505	106	051117	052040
3454	013512	030115	020062	051104
3455	013520	053111	020105	
3456	013524	026460	052040	050131
3457	013532	020105	046123	053101
3458	013540	020105	023443	020123
3459	013546	047524	041040	020105
3460	013554	042524	052123	042105
3461	013562	000040		
3462	013564	050123	042505	020104
3463	013572	042524	052123	020123
3464	013600	047117	054514	020077
3465	013606	054450	051505	047057
3466	013614	020117	020075	027461
3467	013622	024460	000	
3468	013625	116	055122	047440
3469	013632	046116	037531	024040
3470	013640	042531	027523	047516
3471	013646	036440	030440	030057
3472	013654	000051		
3473	013656	005015	047105	020104
3474	013664	043117	052040	050101
3475	013672	006505	000012	
3476				
3477				
3478	013676	005015	051124	050101
3479	013704	042520	020104	047524
3480	013712	032040	000	
3481	013715	116	020117	047503
3482	013722	052116	047522	046114
3483	013730	051105	040440	020124
3484	013736	042101	051104	051505
3485	013744	020123	050123	041505
3486	013752	043111	042511	006504

```

.SBTTL PROGRAM MESSAGES
:OPERATOR INSTRUCTIONS
M.NAM: .ASCIZ <CR><LF>'TE16 DRIVE FUNCTION TIMER (DZTUG-B)'

I.REG: .ASCIZ <CR><LF>'TYPE FIRST ADDRESS OF CONTROLLER '

I.DRVS: .ASCIZ %TYPE TMO2 DRIVE #'S TO BE TESTED %

I.SLVS: .ASCII 'FOR TMO2 DRIVE '

I.DRV: .ASCIZ %- TYPE SLAVE #'S TO BE TESTED %

I.SKEW: .ASCIZ 'SPEED TESTS ONLY? (YES/NO = 1/0)'

I.NRZ: .ASCIZ 'NRZ ONLY? (YES/NO = 1/0)'

M.EOT: .ASCIZ <CR><LF>'END OF TAPE'<CR><LF>

:ERROR MESSAGES
E.TRP4: .ASCIZ <CR><LF>'TRAPPED TO 4'

E.NCON: .ASCIZ 'NO CONTROLLER AT ADDRESS SPECIFIED'<CR><LF>

```

3487	013760	000012			
3488	013762	046524	031060	042040	E.NDRV: .ASCIZ 'TMO2 DRIVE '
3489	013770	044522	042526	000040	
3490	013776	051104	053111	020105	E.NSLV: .ASCII 'DRIVE '
3491	014004	020060	046123	053101	E.DRV: .ASCII 'O SLAVE '
3492	014012	020105			
3493	014014	020060	047516	020124	E.NAVA: .ASCIZ 'O NOT AVAILABLE FOR TEST'<CR><LF>
3494	014022	053101	044501	040514	
3495	014030	046102	020105	047506	
3496	014036	020122	042524	052123	
3497	014044	005015	000		
3498	014047	116	020117	046524	E.UNIT: .ASCIZ 'NO TMO2/TE16 UNIT FOUND TO TEST'<CR><LF>
3499	014054	031060	052057	030505	
3500	014062	020066	047125	052111	
3501	014070	043040	052517	042116	
3502	014076	052040	020117	042524	
3503	014104	052123	005015	000	
3504	014111	123	043117	020124	E.SFT: .ASCIZ 'SOFT ERROR (DATA)'<CR><LF>
3505	014116	051105	047522	020122	
3506	014124	042050	052101	024501	
3507	014132	005015	000		
3508	014135	124	051505	020124	E.HDR: .ASCIZ 'TEST # '
3509	014142	020043	000		
3510	014145	040	042504	044526	E.HDR1: .ASCII ' DEVICE ERROR'<CR><LF>
3511	014152	042503	042440	051122	
3512	014160	051117	005015		
3513	014164	051503	004461	041527	.ASCIZ 'CS1'<HT>'WC'<HT>'BA'<HT>'FC'<HT>'CS2'<HT>'DS'<HT>'ER'<HT>'TC'<CR><LF>
3514	014172	041011	004501	041506	
3515	014200	041411	031123	042011	
3516	014206	004523	051105	052011	
3517	014214	006503	000012		
3518	014220	047440	052125	047440	E.HDR2: .ASCIZ ' OUT OF RANGE ERROR'<CR><LF>
3519	014226	020106	040522	043516	
3520	014234	020105	051105	047522	
3521	014242	006522	000012		
3522	014246	005015	044524	042515	E.TIMOV: .ASCIZ <CR><LF>'TIMER OVERFLOWED'<CR><LF>
3523	014254	020122	053117	051105	
3524	014262	046106	053517	042105	
3525	014270	005015	000		
3526	014273	015	052012	046511	E.TIMEX: .ASCIZ <CR><LF>'TIME EXPIRED WAITING FOR RDY'<CR><LF>
3527	014300	020105	054105	044520	
3528	014306	042522	020104	040527	
3529	014314	052111	047111	020107	
3530	014322	047506	020122	042122	
3531	014330	006531	000012		
3532	014334	043440	050101	021440	E.GAP: .ASCIZ ' GAP # '
3533	014342	000040			
3534					
3535					
3536	014344	025052	025052	025052	;TIME DOCUMENT LINES
3537	014352	025052	025052	025052	L.HDR1: .ASCIZ '*****'
3538	014360	025052	025052	025052	
3539	014366	025052	025052	025052	
3540	014374	025052	025052	025052	
3541	014402	025052	025052	025052	
3542	014410	025052	025052	025052	

3543	014416	025052	025052	025052	
3544	014424	025052	025052	025052	
3545	014432	025052	025052	025052	
3546	014440	025052	025052	025052	
3547	014446	025052	025052	025052	
3548	014454	005015	000		
3549	014457	052	052040	030115	L.HDR2: .ASCII '* TMO2 DRIVE FUNCTION TIMES- DRIVE # '
3550	014464	020062	051104	053111	
3551	014472	020105	052506	041516	
3552	014500	044524	047117	052040	
3553	014506	046511	051505	020055	
3554	014514	051104	053111	020105	
3555	014522	020043			
3556	014524	020060	046123	053101	L.DRV: .ASCII 'D SLAVE # '
3557	014532	020105	020043		
3558	014536	020060	040		L.SLV: .ASCII 'D '
3559	014541	071	041440	040510	L.CHAN: .ASCIZ '9 CHAN. SER # '
3560	014546	027116	051440	051105	
3561	014554	021440	000040		
3562	014560	006440	025012	005015	L.HDR3: .ASCII ' <CR><LF>' * <CR><LF>
3563	014566	020052	052506	041516	.ASCIZ '* FUNCTION' <HT><HT>' TIME(SPECIFICATION)' <HT>' TIME(ACTUAL)' <CR><LF>
3564	014574	044524	047117	004411	
3565	014602	044524	042515	051450	
3566	014610	042520	044503	044506	
3567	014616	040503	044524	047117	
3568	014624	004451	044524	042515	
3569	014632	040450	052103	040525	
3570	014640	024514	005015	000	
3571					
3572	014645	122	047101	042507	L.RNG: .ASCIZ 'RANGE=<'
3573	014652	036075	000		
3574	014655	101	052103	040525	L.ACT: .ASCIZ 'ACTUAL='
3575	014662	036514	000		
3576					
3577					:TEST DESCRIPTOR HEADERS
3578	014665	052	053440	044522	A.T001: .ASCIZ '* WRITE FROM BOT' <HT>
3579	014672	042524	043040	047522	
3580	014700	020115	047502	004524	
3581	014706	000			
3582	014707	052	053440	044522	A.T002: .ASCIZ '* WRITE START' <HT><HT>
3583	014714	042524	051440	040524	
3584	014722	052122	004411	000	
3585	014727	052	053440	044522	A.T003: .ASCIZ '* WRITE SHUTDOWN' <HT>
3586	014734	042524	051440	052510	
3587	014742	042124	053517	004516	
3588	014750	000			
3589	014751	052	053440	044522	A.T004: .ASCIZ '* WRITE SETTLEDOWN' <HT>
3590	014756	042524	051440	052105	
3591	014764	046124	042105	053517	
3592	014772	004516	000		
3593	014775	052	051040	040505	A.T005: .ASCIZ '* READ FROM BOT' <HT><HT>
3594	015002	020104	051106	046517	
3595	015010	041040	052117	004411	
3596	015016	000			
3597	015017	052	051040	040505	A.T006: .ASCIZ '* READ START' <HT><HT>
3598	015024	020104	052123	051101	

3599	015032	004524	000011		
3600	015036	020052	042522	042101	A.T007: .ASCIZ '* READ SHUTDOWN' <HT><HT>
3601	015044	051440	052510	042124	
3602	015052	053517	004516	000011	
3603	015060	020052	042522	042101	A.T010: .ASCIZ '* READ SETTLEDOWN' <HT>
3604	015066	051440	052105	046124	
3605	015074	042105	053517	004516	
3606	015102	000			
3607	015103	052	051040	040505	A.T011: .ASCIZ '* READ REV START' <HT>
3608	015110	020104	042522	020126	
3609	015116	052123	051101	004524	
3610	015124	000			
3611	015125	052	051040	040505	A.T012: .ASCIZ '* READ REV SHUTDOWN' <HT>
3612	015132	020104	042522	020126	
3613	015140	044123	052125	047504	
3614	015146	047127	000011		
3615	015152	020052	042522	042101	A.T013: .ASCIZ '* READ REV SETTLEDOWN' <HT>
3616	015160	051040	053105	051440	
3617	015166	052105	046124	042105	
3618	015174	053517	004516	000	
3619	015201	052	052040	051125	A.T014: .ASCIZ '* TURN AROUND DELAY F-R' <HT>
3620	015206	020116	051101	052517	
3621	015214	042116	042040	046105	
3622	015222	054501	043040	051055	
3623	015230	000011			
3624	015232	020052	052524	047122	A.T015: .ASCIZ '* TURN AROUND DELAY R-F' <HT>
3625	015240	040440	047522	047125	
3626	015246	020104	042504	040514	
3627	015254	020131	026522	004506	
3628	015262	000			
3629	015263	052	043440	050101	A.T016: .ASCIZ '* GAP SIZE-STOP HALF' <HT>
3630	015270	051440	055111	026505	
3631	015276	052123	050117	044040	
3632	015304	046101	004506	000	
3633	015311	052	043440	050101	A.T017: .ASCIZ '* GAP SIZE-START HALF' <HT>
3634	015316	051440	055111	026505	
3635	015324	052123	051101	020124	
3636	015332	040510	043114	000011	
3637	015340	020052	040507	020120	A.T020: .ASCIZ '* GAP SIZE-INTERRECORD' <HT>
3638	015346	044523	042532	044455	
3639	015354	052116	051105	042522	
3640	015362	047503	042122	000011	
3641	015370	020052	040507	020120	A.T021: .ASCIZ '* GAP CONSISTANCY' <HT>
3642	015376	047503	051516	051511	
3643	015404	040524	041516	004531	
3644	015412	000			
3645	015413	052	042040	052101	A.T023: .ASCIZ '* DATA TIME-200BPI' <HT>
3646	015420	020101	044524	042515	
3647	015426	031055	030060	050102	
3648	015434	004511	000		
3649	015437	052	042040	052101	A.T024: .ASCIZ '* DATA TIME-556BPI' <HT>
3650	015444	020101	044524	042515	
3651	015452	032455	033065	050102	
3652	015460	004511	000		
3653	015463	052	042040	052101	A.T025: .ASCIZ '* DATA TIME-800BPI' <HT>
3654	015470	020101	044524	042515	

3655	015476	034055	030060	050102	
3656	015504	004511	000		
3657	015507	052	042040	052101	A.T026: .ASCIZ '* DATA TIME-1600BPI' <HT>
3658	015514	020101	044524	042515	
3659	015522	030455	030066	041060	
3660	015530	044520	000011		
3661	015534	020052	051105	051501	A.T027: .ASCIZ '* ERASE GAP TIME' <HT>
3662	015542	020105	040507	020120	
3663	015550	044524	042515	000011	
3664	015556	020052	051127	052111	A.T030: .ASCIZ '* WRITE FILE MARK' <HT>
3665	015564	020105	044506	042514	
3666	015572	046440	051101	004513	
3667	015600	000			
3668	015601	052	052040	050101	A.T031: .ASCIZ '* TAPE SPEED-FWD' <HT>
3669	015606	020105	050123	042505	
3670	015614	026504	053506	004504	
3671	015622	000			
3672	015623	052	052040	050101	A.T032: .ASCIZ '* TAPE SPEED-REV' <HT>
3673	015630	020105	050123	042505	
3674	015636	026504	042522	004526	
3675	015644	000			
3676					
3677	015645	015	057012	000107	L.CNTG: .ASCIZ <CR><LF>'↑G'
3678	015652	005015	053523	036522	L.SWR: .ASCIZ <CR><LF>'SWR='
3679	015660	000			
3680	015661	040	047040	053505	L.NEW: .ASCIZ ' NEW= '
3681	015666	020075	000		
3682	015671	015	037412	005015	L.QUEST: .ASCIZ <CR><LF>'?' <CR><LF>
3683	015676	000			
3684		015700			.EVEN
3685		015700			RDBUF=.
3686		015700			WTBUF=.
3687	015700	000200			.BLKW 128.
3688		000001			.END

A	010620	CNTRLO=	000017	E.HDR	014135	L.HDR3	014560	RESVEC=	000010
ACCL	= 100000	CNTRLU=	000025	E.HDR1	014145	L.NEW	015661	REVRD	005454
ANGTAB	001412	CNVDEC	002730	E.HDR2	014220	L.QUES	015671	RHINIT	005174
AS	= 000016	CNVVCT	002622	E.NAVA	014014	L.RNG	014645	RMR	= 000004
ASFLG	001130	CNVTAO	003260	E.NCON	013715	L.SLV	014536	RMD	= 000006
ATA	= 100000	CNVTD	002742	E.NDRV	013762	L.SWR	015652	RMDOFF=	000002
ATIME	001012	CNVTO	002634	E.NSLV	013776	MCPE	= 020000	R10	=%000000
ATIMTB	001014	COUNT	001764	E.SFT	014111	MDPE	= 000400	R11	=%000001
A.T001	014665	CR	= 000015	E.TIME	014273	MMVEC	= 000250	R12	=%000002
A.T002	014707	CRLF	001374	E.TIMO	014246	MOL	= 010000	R13	=%000003
A.T003	014727	CSITM	= 002000	E.TRP4	013676	MR	= 000024	R14	=%000004
A.T004	014751	CS1	= 000000	E.UNIT	014047	MXF	= 001000	R15	=%000005
A.T005	014775	CS2	= 000010	FC	= 000006	M.EOT	013656	SC	= 100000
A.T006	015017	DASH	001405	FCE	= 001000	M.NAM	013330	SCOPE	= 104000
A.T007	015036	DB	= 000022	FINISH	012616	NAMPTR	001672	SCPADR	001002
A.T010	015060	DCONST	003036	FMT	= 000020	NED	= 010000	SDWN	= 000020
A.T011	015103	DELAY	004740	FPEVEC=	000244	NEF	= 004000	SKENTS	012764
A.T012	015125	DELAYV	004770	FRMCNT=	177400	NEM	= 004000	SLA	= 000001
A.T013	015152	DELTIM	001114	FWDSPC	005472	NOP	= 000000	SLAVES	006370
A.T014	015201	DIGTAB	001132	GAP	001120	NORM11=	000300	SLR	= 177774
A.T015	015232	DIVIDE	005026	GAPOK	004612	NRZFLG	001127	SLVAVA	005144
A.T016	015263	DLT	= 100000	GAPTBL	001054	NSG	= 000400	SLVNUM	001005
A.T017	015311	DPR	= 000400	GO	= 000001	OCTALO	001116	SLVPTR	001006
A.T020	015340	DRIVES	006140	GTIMTB	001572	ODIGIT	001144	SLVTBL	001164
A.T021	015370	DRVAVA	005116	HLT	= 104400	OPI	= 020000	SN	= 000030
A.T023	015413	DRVNUM	001004	HT	= 000011	OR	= 000200	SNPT	005574
A.T024	015437	DRVTL	001154	IDB	= 000010	OSC	= 000100	SPACE	001410
A.T025	015463	DRY	= 000200	IE	= 000100	OUT	002240	SPACE2	001407
A.T026	015507	DRYCLR=	000010	ILF	= 000001	OUTBUF=	005724	SPCFWD=	000030
A.T027	015534	DS	= 000012	ILR	= 000002	OUTGAP	003146	SPCREV=	000032
A.T030	015556	DT	= 000026	INBUF	001264	OUTSPC	003052	SPR	= 002000
A.T031	015601	DTE	= 010000	INCVAE=	000100	PARVEC=	000114	SSC	= 000100
A.T032	015623	DVA	= 004000	INIT	005724	PAT	= 000020	STIMTB	001416
A16	= 000400	DVO	= 000000	IOTVEC=	000020	PEFLRC=	000200	STKPTR=	000600
A17	= 001000	DV1	= 000001	IR	= 000100	PES	= 000040	SUSWR	005730
BA	= 000004	DV2	= 000002	ITCNT	001121	PE1600=	002000	SWR	001000
BAI	= 000010	DV3	= 000003	I.DRV	013524	PFVEC	= 000024	SWREG	000176
BEGIN	006764	DV4	= 000004	I.DRVS	013443	PGE	= 002000	SW06	= 000100
BELL	001403	DV5	= 000005	I.NRZ	013625	PIP	= 020000	SW07	= 000200
BKSLSH	001377	DV6	= 000206	I.REG	013376	PIRQ	= 177772	SW08	= 000400
BOT	= 000002	DV7	= 000007	I.SKEW	013564	PIRVEC=	000240	SW09	= 001000
BPI200=	000000	ECHO	001401	I.SLVS	013505	PLKCSR=	172540	SW10	= 002000
BPI556=	000400	EMTVEC=	000030	LF	= 000012	PLKVEC=	000104	SW11	= 004000
BPI800=	001000	END	012732	LKS	= 177546	PRGFLG	001124	SW13	= 020000
BPTVEC=	000014	EOT	= 002000	LKVEC	= 000100	PSEL	= 002000	SW14	= 040000
CDM11	= 000320	ER	= 000014	LPB	= 177516	PSW	= 177776	SW15	= 100000
CHKDRV	006270	ERASE	= 000024	LPS	= 177514	PUBLIS	003346	TAP	= 040000
CHKSLV	006574	ERFLG	001123	L.ACT	014655	RDBUF	= 015700	TBITVE=	000014
CH7	= 010000	ERR	= 040000	L.CHAN	014541	RDFW	= 000070	TC	= 000032
CKSWR	001770	ERRTRP	003650	L.CNTG	015645	ROREV	= 000076	TCRLF	002360
CLR	= 000040	ERRVEC=	000004	L.DRV	014524	RDSW	001766	TEMPST	001762
CNTLU	002030	E.DRV	014004	L.HDR1	014344	RDY	= 000200	TIB	001760
CNTRLC=	000003	E.GAP	014334	L.HDR2	014457	READ	005436	TIMER	004444



K07

TIMERR 004454	TSTNUM 001122	TST023 011672	TYPOCT 002630	SCNTRL 002325
TIMERO 004470	TST001 007336	TST024 012014	UBREAK= 177770	SCRFLF 002326
TIMER1 004420	TST002 007422	TST025 012144	UNS = 040000	SFILL 002315
TIMOK 004502	TST003 007500	TST026 012274	UNTFND 001125	SHT = 000011
TIMON 004354	TST004 007570	TST027 012424	UPE = 020000	SNULL 002314
TKB = 177562	TST005 007676	TST030 012510	WAITRD 005232	STKFLG 002317
TKISR 003606	TST006 007762	TST031 012772	WAITTI 005234	STPB 002322
TKS = 177560	TST007 010046	TST032 013120	WC = 000002	STPFLG 002316
TKVEC = 000060	TST010 010146	TTIN 002242	WCE = 040000	STPS = 002320
TMBASE 001010	TST011 010266	TTIN1 002262	WCHKF = 000050	. = 016300
TMCHD 005554	TST012 010364	TTIN2 002276	WCHKR = 000056	.HLT 003652
TMCSI = 172440	TST013 010474	TYPDEC 002736	WFMK = 000026	.INPUT 003474
TMK = 000004	TST014 010634	TYPE = 000004	WFD = 000060	.RESTO 002600
TPB = 177566	TST015 010726	TYPE1 002352	WRDCNT= 177600	.REWIND 005336
TPS = 177564	TST016 011034	TYPE2 002400	WRITE 005420	.SAVE 002556
TPVEC = 000064	TST017 011130	TYPE3 002406	WRL = 004000	.SCOPE 004126
TRAPVE= 000034	TST020 011240	TYPE4 002412	WRT.BK 005512	.TYPE 002332
TRE = 040000	TST021 011360	TYPFLG 001126	WTBUF = 015700	
TRTVEC= 000014	TST022 011664	TYPHDR 007210	SCHARC 002324	

. ABS. 016300 000

ERRORS DETECTED: 0

DZTUGB, DZTUGB. SEQ/SOL/NL: TOC/DOC+DZTUGB.P11

RUN-TIME: 47.3 SECONDS

RUN-TIME RATIO: 179/12=14.7

CORE USED: 7K (13 PAGES)

DOCUMENT PAGES: 87

EOF1DZTUGBSEQ

00010000

770804

PDP10 411