

# TJU16

DRIVE FUNCTION TIMER  
MD-11-DZTUG-A

EP DZTUG A DL A  
COPYRIGHT 1977  
FICHE 1 OF 1

MAR 1977  
**digital**  
MADE IN USA

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

Small technical diagram or table in the bottom right corner.

40910ZTUGASEG

00010000

770224

801  
PDP10 411

IDENTIFICATION

SEQ 0001

Product Code: MAINDEC-11-DZTUG-A-D  
Product Name: TMO2/TU16J DRIVE FUNCTION TIMER  
Date Created: JANUARY 1977  
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 EQUIPMEN CORPORATION

TABLE OF CONTENTS

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

ABSTRACT

Program DZTUG measures the time required and GAP sizes produced by the TMO2/TU16J 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.

CHAPTER 1  
REQUIREMENTS

PDP-11 Family Central Processor with 4K memory with up to 64 TMO2/TU16J  
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

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 CHAPTER 3 FOR OPTIONS)

Program will request DRIVE (TMO2) and SLAVE (TU16J) 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.

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.

SW15 (100000)	HALT ON ERROR	This switch when set will halt the processor when an error is detected. The PC+2 and PSW at the time of the error is stored on the stack. Pressing continue will cause the error to be typed (if selected) and further testing resumed.
SW14 (040000)	LOOP SUBTEST	This switch when set loops the current subtest regardless of error condition.
SW13 (020000)	INHIBIT ERROR TYPEOUT	This switch when set inhibits error typeout.
SW11 (004000)	INHIBIT SUB- TEST ITERATION	This switch when set causes each subtest to be executed only once. (Initial Startup Only).
SW10 (002000)	INHIBIT FUNCTION TIME PUBLICATION	This switch when set will inhibit the printing of the function times. (See Chapter 8.)
SW09 (001000)	RING BELL ON ERROR	This switch when set will ring the bell on the TTY when an error is detected.
SW07 (000200)	HALT AFTER SELECTED TEST	This switch when set will cause the program to HALT after the test selected in SW05-SW00 is executed.
SW06 (000100)	CONTINUOUS CYCLE	THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO RUN CONTINUOUSLY UNTIL STOPPED BY THE OPERATOR.
SW5-0	TEST SELECT	THE PROGRAM WILL HALT AFTER EXECUTION OF THE TEST SELECTED WHEN SW07 IS SET.



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 AS SOFT ERRORS AND HAVE NO EFFECT ON TIME.

TEST # XXXXXX DEVICE ERROR

CS1	WE	BA	FC	CS2	DS	ER1
aaaaaa	bbbbbb	cccccc	dddddd	eeeeee	ffffff	gggggg

where:

XXXXXX = Test Number

AAAAAA-IIIIII = Contents of Tape Register 172440-172454

4.2 ERROR TYPEOUT FORMAT (FUNCTION TIME OUT OF RANGE)

TEST # XXXXXX OUT OF RANGE ERROR

RANGE = (AAAAAA-BBBBBB) ACTUAL = CCCCCC

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.

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 <↑G> BY DOING A JSR PC,CKSWR  
(REFER TO CHAPTER 3 FOR DESCRIPTION)>

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.

CHAPTER 7  
PROGRAM DESCRIPTION

7.1 SAMPLE TIME DOCUMENT

TYPE FIRST ADDRESS OF CONTROLER 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=<176000-172000>	ACTUAL=174740
* 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=<045000-041000>	ACTUAL=043580
* 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

TM02 DRIVE FUNCTION TIMER

7.1.1 SAMPLE TIME DOCUMENT FOR TAPE SPEED TESTS

TYPE FIRST ADDRESS OF CONTROLLER 172440

TYPE TM02 DRIVE #'S TO BE TESTED 0

FOR TM02 DRIVE 0- TYPE SLAVE #'S TO BE TESTED 7

SPEED TESTS ONLY? (YES/NO = 1/0) 1

\*\*\*\*\*

\*TM02 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

TMC2 DRIVE FUNCTION TIMER

SEQ 0014

7.2 TEST SEQUENCE WITH RELATED ADJUSTMENTS AND ASSOCIATED HARDWARE

TEST NO./NAME	RELATED ADJUSTMENTS	ASSOCIATED HARDWARE
1. WRITE FROM BOT	*NONE	*M8911 ROM*M8903 ACCL CNTR
2. WRITE START	* "	* " * "
3. WRITE SHUTDOWN	* "	* " * "
4. WRITE SETTLEDOWN	* "	*M8910 SETTLEDOWN ONE SHOT
5. READ FROM BOT	* "	*M8911 ROM*M8903 ACCL CNTR
6. READ START	* "	* " * "
7. READ SHUTDOWN	* "	* " * "
10. READ SETTLEDOWN	* "	*M8910 SETTLEDOWN ONE SHOT
11. READ REVERSE START	* "	*M8911 ROM*M8903 ACCL CNTR
12. READ REVERSE SHUTDOWN	* "	* " * "
13. READ REVERSE SETTLEDOWN	* "	*M8910 SETTLEDOWN ONE SHOT
14. TURN AROUND F-R	* "	*M8911 ROM*M8903 ACCL CNTR
15. TURN AROUND R-F	* "	* " * "
16. GAP SIZE-STOP HALF	*FWRD/REV SPEED-START/STOP-RAMPS	*CAPSTAN SERVO LOOP
17. GAP SIZE-START HALF	*SAME AS IN TEST 16	* " " "
20. GAP SIZE INTERRECORD	*FWRD/REV SPEED	* " " "

TMD2 DRIVE FUNCTION TIMER

SEQ 0015

21. GAP CONSISTENCY	*SAME AS IN TEST 16	*WRITE CLOCK
*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	* "	*M8911 ROM*M8903 ACCL CNTR
30. WRITE FILE MARK	* "	* " " * " " "
31. TAPE SPEED-FORWARD	*FWD SPEED	*CAPSTAN SERVO LOOP
32. TAPE SPEED-REVERSE	*REVERSE SPEED	*CAPSTAN SERVO LOOP

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



## 7.3 TEST DESCRIPTIONS:

THE FIRST THIRTEEN (13) TESTS (T1 - T15) ARE CHECKS OF THE ROM CIRCUITS IN THE TU16J (M9811), THE ACCL COUNTER IN THE TM02 (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 INTERRECORD 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

## T4. WRITE SETTLEDOWN:

SEQ 0017

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

## 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

## TMD2 DRIVE FUNCTION TIMER

## 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

## TMO2 DRIVE FUNCTION TIMER

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.

## GAP MEASUREMENTS:

THE PREVIOUS THIRTEEN (13) TESTS WERE MEASUREMENTS OF LOGIC DELAYS PERFORMED BY THE TMO2 OR TU16J 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.
7. STOP

## T21. GAP CONSISTENCY:

NOW THAT WE HAVE ESTABLISHED THAT THE INTERRECORD 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\*\*

## TM02 DRIVE FUNCTION TIMER

## 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



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.

M02

DZTUG-A TMO2/TU16J DRIVE FUNCTION TIMER MACY11 27(1906) 09-DEC-76 10:45  
DZTUGA.P11 08-DEC-76 12:38 TABLE OF CONTENTS

SEQ 0025

7	STARTING INSTRUCTIONS
33	MACRO DEFINITIONS
57	REGISTER ASSIGNMENTS
86	TMO2/TU16 REGISTER BITS
271	TIME SPECIFICATION TABLE
308	GAP TIME SPECIFICATION TABLE
332	TEST HEADER POINTERS
432	PROGRAM SUBROUTINES
433	TYPE SUBROUTINE
465	OCTAL TO ASCII & TYPE ROUTINE
502	OCTAL TO DECIMAL & TYPE ROUTINE
535	TYPE SPECIFIED TIMES ROUTINE
566	TYPE GAP TIMES SUBROUTINE
597	ASCII TO OCTAL CONVERT SUBROUTINE
619	PUBLISH SUBROUTINE
660	INPUT SUBROUTINE
706	ERROR SERVICE ROUTINES
771	SCOPE SUBROUTINE
831	TIMER SUBROUTINES
959	DELAY SUBROUTINES
984	DIVIDE SUBROUTINE
1021	DRIVE SUBROUTINES
1206	PROGRAM INITIALIZATION
1469	START OF TESTS
2319	PROGRAM MESSAGES

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60

```

.NLIST MC
.LIST ME
.ABS
.MCALL $CPVEC,$CPREG,$SCATCH,$STYPE
.TITLE DZTUG-A TMO2/TU16J DRIVE FUNCTION TIMER
.SBTTL STARTING INSTRUCTIONS
:LOADING AND STARTING PROCEEDURE
:LOAD PROGRAM USING ABS LOADER
:LOAD ADDRESS 200
:SET SWITCH OPTIONS
:PRESS START

:GENERAL REGISTER USAGE
:R0=ADDRESS OF 'PC' REGISTER (SET BY SCOPE)
:R1=ADDRESS OF 'DS' REGISTER (SET BY SCOPE)
:R2=RETURN PC FROM TIMER (SET BY EACH TEST)
:R3=INDEX INDICATING PREVIOUS OSCILLATOR POLARITY (SET BY TIMER)
:R4=CONTAINS 'TICK' COUNT WHEN TIMER IS RUNNING (SET BY TIMER)
:R5=ADDRESS OF CS1 (SET BY SCOPE)

:SWITCH REGISTER SWITCH ASSIGNMENTS
SW15= 100000 ;HALT ON ERROR
SW14= 040000 ;LOOP SUBTEST
SW13= 020000 ;INHIBIT ERROR TYPE OUT
SW11= 004000 ;INHIBIT SUBTEST ITERATION
SW10= 002000 ;INHIBIT PUBLICATION OF FUNCTION TIMES
SW09= 001000 ;RING BELL ON ERROR
SW08= 000400 ;TYPE LINE ITEM AFTER EACH ITERATION
SW07= 000200 ;HALT ON TEST SELECTED IN SW05-SW00
SW06= 000100 ;CONTINUOUS CYCLE

.SBTTL MACRO DEFINITIONS
.MACRO SAVE
JSR PC,SAVE ;SAVE REGISTERS ON THE STACK
.ENDM
.MACRO RESTORE
JSR PC,RESTORE ;RESTORE REGISTERS FROM THE STACK
.ENDM
.MACRO INPUT
JSR PC,INPUT ;GET USER INPUT
.ENDM
.MACRO REWIND
JSR PC,REWIND ;REWIND SLAVE
BVS 99$ ;BRANCH IF ERROR ON REWIND
.ENDM
.MACRO TIMEON
JSR PC,TIMON ;TURN TIMER ON
.ENDM
.MACRO TIMCHK
JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
.ENDM
.MACRO SETGO
INC (R5) ;SET 'GO' BIT
.ENDM

```

100000  
040000  
020000  
004000  
002000  
001000  
000400  
000200  
000100



71  
72 000000  
73 000002  
74 000004  
75 000006  
76 000010  
77 000012  
78 000014  
79 000016  
80 000022  
81 000024  
82 000026  
83 000030  
84 000032  
85  
86  
87  
88 000001  
89 000000  
90 000002  
91 000006  
92 000010  
93 000026  
94 000024  
95 000030  
96 000032  
97 000050  
98 000056  
99 000060  
100 000070  
101 000076  
102 000100  
103 000200  
104 000400  
105 001000  
106 002000  
107 004000  
108 020000  
109 040000  
110 100000  
111  
112 000000  
113 000001  
114 000002  
115 000003  
116 000004  
117 000005  
118 000006  
119 000007  
120 000010  
121 000020  
122 000040  
123 000100  
124 000200  
125 000400  
126 001000

;TM02/TU16 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/TU16 REGISTER BITS  
;RHCS1-CS1(R5)

GO= 1  
NOP= 0  
RWDOFF= 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= 5  
DV7= 7  
BAI= 10  
PAT= 20  
CLR= 40  
IR= 100  
OK= 200  
MDPE= 400  
MXF= 1000

127	002000	PGE=	2000
128	004000	NEM=	4000
129	010000	NED=	10000
130	020000	UPE=	20000
131	040000	WCE=	40000
132	100000	DLT=	100000
133		;RHDS-DS(R5)	
134	000001	SLA=	1
135	000002	BOT=	2
136	000004	TMK=	4
137	000010	IDB=	10
138	000020	SDWN=	20
139	000040	PES=	40
140	000100	SSC=	100
141	000200	DRY=	200
142	000400	DPR=	400
143	002000	EOT=	2000
144	004000	WRL=	4000
145	010000	MOL=	10000
146	020000	PIP=	20000
147	040000	ERR=	40000
148	100000	ATA=	100000
149		;RHER-ER(R5)	
150	000001	ILF=	1
151	000002	ILR=	2
152	000004	RMR=	4
153			
154	000020	FMT=	20
155	000100	INCVAE=	100
156	000200	PEFLRC=	200
157	000400	NSG=	400
158	001000	FCE=	1000
159	002000	CSITM=	2000
160	004000	NEF=	4000
161	010000	DTE=	10000
162	020000	OPI=	20000
163	040000	UNS=	40000
164			
165		;RHMR-MR(R5)	
166	000100	OSC=	100
167			
168		;RHDT-DT(R5)	
169	002000	SPR=	2000
170	010000	CH7=	10000
171	040000	TAP=	40000
172			
173		;RHTC-TC(R5)	
174	000300	NORM11=	300
175	000320	CDM11=	320
176	000000	BPI200=	0
177	000400	BPI556=	000400
178	001000	BPI800=	001000
179	002000	PE1600=	002000
180	100000	ACCL=	100000
181			
182		;INSTRUCTION EQUATES	

183	104400	HLT=	TRAP	
184	104000	SCOPE=	EMT	
185	000004	TYPE=	IOT	
186				
187				
188	005724	: MISCELLANEOUS EQUATES		
189	177400	OUTBUF=	INIT	: OUTPUT BUFFER START AT BEG OF PROGRAM
190	177600	FRMCNT=	-256.	: FRAME COUNT
191		WRDCNT=	-128.	: WORD COUNT
192		: ASCII EQUATES		
193	000003	CNTRLC=	3	: ASCII CODE FOR CONTROL C (↑C)
194	000011	HT=	11	: ASCII CODE FOR HORIZONTAL TAB
195	000012	LF=	12	: ASCII CODE FOR LINE FEED
196	000015	CR=	15	: ASCII CODE FOR CARRIAGE RETURN
197	000017	CNTRLO=	17	: ASCII CODE FOR CONTROL O (↑O)
198	000025	CNTRLU=	25	: ASCII CODE FOR CONTROL U (↑U)

```

200 ;SETUP TRAP VECTORS
201 ;=TBITVEC
202 000014 000016 .WORD +2 ;SET 'T' TRAP TO TIMER ROUTINE
203 000016 000000 .WORD HALT ;PRIORITY LEVEL 7
204 000020 002332 .WORD TYPE ;SET IOT TRAP TO .TYPE ROUTINE
205 000022 000000 .WORD 0 ;PRIORITY LEVEL 0
206 000024 000026 .WORD PFVEC+2 ;POWER FAIL TRAP TO HALT
207 000026 000000 .WORD HALT ;AT PFVEC+2
208 000030 004126 .WORD SCOPE ;SET EMT TRAP TO .SCOPE ROUTINE
209 000032 000340 .WORD 340 ;PRIORITY LEVEL 7
210 000034 003652 .WORD HLT ;SET TRAP TRAP TO .HLT ROUTINE
211 000036 000340 .WORD 340 ;PRIORITY LEVEL 7
212 ;=TKVEC
213 000060 003606 .WORD TKISR
214 000062 000340 .WORD 340
215
216 ;SOFTWARE SWITCH REGISTER LOC. 176
217 ;=176
218 000176 000000 SWREG: 0 ;SOFTWARE SWITCH REGISTER
219
220 ;=200
221 000200 000137 005724 JMP @#INIT ;GO TO START OF PROGRAM
222
223 ;=500
224 000500 000600 STKPTR= 600 ;STACK
225
226 ;=1000
227 ;PROGRAM TAGS
228 001000 177570 SWR: 177570 ;SWITCH REGISTER
229 001002 000000 SCPADR: .WORD 0
230 001004 000 DRVNUM: .BYTE 0 ;TMO2 DRIVE UNDER TEST
231 001005 000 SLVNUM: .BYTE 0 ;TU16 SLAVE UNDER TEST
232 001006 000000 SLVPTR: .WORD 0 ;POINTER TO SLAVE TABLE (SLVTBL) BELOW
233 001010 172440 TMBASE: .WORD TMCS1 ;BASE ADDRESS OF TMO2/TU16 REGISTERS
234 001012 000000 ATIME: .WORD 0 ;CONTAINS 'TICK' COUNT
235 001014 000020 ATIMTBL: .BLKW 16. ;EACH ENTRY CONTAINS TIME FOR FUNCTION
236 ;ENTRIES ARE MADE BY 'SCOPE' ROUTINE
237 001054 000020 GAPTBL: .BLKW 16. ;TIMES RECORDED BY 'GAP CONSISTANCY' TEST
238 001114 000000 DELTIM: .WORD 0 ;VARIABLE DELAY
239 001116 000000 OCTALO: .WORD 0
240 001120 000 GAP: .BYTE 0 ;CONTAINS GAP # (USED FOR TST 021)
241 001121 000 ITCNT: .BYTE 0 ;ITERATION COUNT
242 001122 000 TSTNUM: .BYTE 0 ;TEST #
243 001123 000 ERFLG: .BYTE 0 ;ERROR FLAG
244 001124 000 PRGFLG: .BYTE 0 ;PROGRAM FLAG
245 001125 000 UNTFND: .BYTE 0 ;UNIT FOUND INDICATOR
246 001126 000 TYPFLG: .BYTE 0
247 001127 000 NRZFLG: .BYTE 0 ;INDICATES IF DRIVE IS NRZ ONLY.
248 001130 000 ASFLG: .BYTE 0 ;1/0 = YES/NO.
249 ;EVEN
250 001132 030460 DIGTAB: "01
251 001134 031462 "23
252 001136 032464 "45
253 001140 033466 "67
254 001142 034470 "89
255 001144 000006 DDIGITS: .BLKB 6 ;RESERVE SPACE FOR CONVERTED DIGITS
    
```



G03

DZTUG-A TMO2/TU16J DRIVE FUNCTION TIMER MACY11 27(1006) 09-DEC-76 10:45 PAGE 1-6  
DZTUGA.P11 08-DEC-76 12:39 TMO2/TU16 REGISTER BITS

SEQ 0032

256	001152	000		.BYTE	0	:TERMINATOR
257		001154		.EVEN		
258	001154	000010		DRVTBL: .BLKB	8.	:A 0/-1 = DRIVE NOT TO BE/TO BE TESTED
259	001164	000100		SLVTBL: .BLKB	64.	:A 0/-1 = SLAVE NOT TO BE/TO BE TESTED
260	001264	000110		INBUF: .BLKB	72.	:TELETYPE INPUT BUFFER
261	001374	005015	000	CRLF: .ASCIZ	<CR><LF>	:MISCELLANEOUS ASCII CHARACTERS
262	001377	134	000	BKSLSH: .ASCIZ	'\'	
263	001401	060	000	ECHO: .ASCIZ	'0'	
264	001403	007	000	BELL: .ASCIZ	<7>	
265	001405	055	000	DAS: .ASCIZ	'-'	
266	001407	040		SPACE2: .ASCII	' '	
267	001410	000040		SPACE: .ASCIZ	' '	
268	001412	004476	000	ANGTAB: .ASCIZ	'>><HT>	
269		001416		.EVEN		

271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306

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 #

.WORD	MAX,MIN	TIME IN MS	FUNCTION	TEST #
STIMTBL: .WORD 0,0		: SPARE		
.WORD 17600.,17200.		: 176.0-172.0	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 04500.,04100.		: 45.0-41.0	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 SHUTDOWN	TST012
.WORD 01350.,00810.		: 13.5-8.1	RD REV STLDOWN	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.

308  
 309  
 310  
 311  
 312  
 313  
 314  
 315  
 316  
 317  
 318  
 319  
 320  
 321  
 322  
 323  
 324  
 325  
 326  
 327  
 328  
 329  
 330

.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 (TSTC21).  
 ;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

Line	Address	Value	Description
332			.SBTTL TEST HEADER POINTERS
333			; THE BELOW TABLE CONTAINS POINTERS TO EACH TEST'S DESCRIPTOR
334	001672	000000	NAMPTR: .WORD 0
335	001674	014667	.WORD A.T001
336	001676	014711	.WORD A.T002
337	001700	014731	.WORD A.T003
338	001702	014753	.WORD A.T004
339	001704	014777	.WORD A.T005
340	001706	015021	.WORD A.T006
341	001710	015040	.WORD A.T007
342	001712	015062	.WORD A.T010
343	001714	015105	.WORD A.T011
344	001716	015127	.WORD A.T012
345	001720	015154	.WORD A.T013
346	001722	015203	.WORD A.T014
347	001724	015234	.WORD A.T015
348	001726	015265	.WORD A.T016
349	001730	015313	.WORD A.T017
350	001732	015342	.WORD A.T020
351	001734	015372	.WORD A.T021
352	001736	000000	.WORD 0 ;DUMMY TEST
353	001740	015415	.WORD A.T023
354	001742	015441	.WORD A.T024
355	001744	015465	.WORD A.T025
356	001746	015511	.WORD A.T026
357	001750	015536	.WORD A.T027
358	001752	015560	.WORD A.T030
359	001754	015603	.WORD A.T031
360	001756	015625	.WORD A.T032

```

362
363 ;CHECK SWITCH REGISTER ROUTINE. CHECKS FOR ↑G TO ALLOW CHANGING
364 ;OF LOC.176.
365 ;CALL IS BY WAY OF JSR PC,CKSWP
366 ;LOCATIONS USED:
367 001760 000000 TIB: .WORD 0
368 001762 000000 TEMPST: .WORD 0
369 001764 000000 COUNT: .WORD 0
370 001766 000000 RDSW: .WORD 0
371
372 001770 022767 000176 177002 CKSWR: CMP #SWREG,SWR ;SOFTWARE SWITCH REG PRESENT
373 001776 001120 BNE OUT ;NO GET OUT
374 002000 016767 175556 177752 MOV TKB,TIB ;AND STRIP OFF
375 002006 042767 177600 177744 BIC #177600,TIB ;THE GARBAGE
376 002014 022767 000007 177736 CMP #7,TIB ;IS IT A <↑G>
377 002022 001106 BNE OUT
378 002024 000004 015647 TYPE,L.CNTG
379 002030 000004 015654 CNTLU: TYPE,L.SWR
380 002034 017702 176740 MOV #SWR,R2
381 002040 004767 000564 JSR PC,TYPOCT
382 002044 000004 015663 TYPE,L.NEW
383
384 002050 005067 177706 CLR TEMPST
385 002054 012767 000007 177702 MOV #7,COUNT
386 002062 004767 000154 1$: JSR PC,TTIN ;GO READ A CHARACTER
387 002066 042767 177600 177664 BIC #177600,TIB ;STRIP OFF GARBAGE
388 002074 122767 000025 177656 CMPB #25,TIB ;IS IT A ↑U?
389 002102 001001 BNE 2$ ;BRANCH IF NOT
390 002104 000751 3$: BR CNTLU ;START OVER
391 002106 122767 000015 177644 2$: CMPB #15,TIB ;IS IT A <CR>?
392 002114 001012 BNE 4$ ;BRANCH IF NOT
393 002116 012767 000200 177642 MOV #200,RDSW
394 002124 004767 000230 JSR PC,TCRLF ;ECHO IT WITH <LF>
395 002130 022767 000007 177626 CMP #7,COUNT ;WAS IT FIRST CHARACTER
396 002136 001034 BNE 7$ ;CHANGE SWR IF NOT FIRST ONE
397 002140 000437 8$: BR OUT ;GET OUT
398 002142 122767 000060 177610 4$: CMPB #60,TIB
399 002150 003004 BGT 5$
400 002152 122767 000067 177600 CMPB #67,TIB
401 002160 002003 BGE 6$
402 002162 000004 015673 5$: TYPE,L.QUEST
403 002166 000746 BR 3$ ;START OVER IF NOT LEGAL CHARACTER
404 002170 006367 177566 6$: ASL TEMPST
405 002174 006367 177562 ASL TEMPST
406 002200 006367 177556 ASL TEMPST
407 002204 142767 000060 177546 BICB #60,TIB ;GET NITTY-GRITTY
408 002212 156767 177542 177542 BISB TIB,TEMPST
409 002220 005367 177540 DEC COUNT ;ONLY WANT 6 DIGITS
410 002224 001756 BEQ 5$
411 002226 000715 BR 1$
412 002230 016777 177526 176542 7$: MOV TEMPST,#SWR ;CHANGE SWITCH REGISTER CONTENTS
413 002236 000740 BR 8$
414 002240 000207 CUT: RTS PC

```



```

(1) 002420 004767 000026      JSR    PC,5$      ;; TYPE CHARACTER
(1) 002424 122726 000012      3$:    CMPB    #12,(SP)+  ;; CHECK IF CHARACTER WAS A LINE FEED
(1) 002430 001350              BNE    TYPE1      ;; BRANCH IF NOT LINE FEED
(1) 002432 016746 177656      MOV    $NULL,-(SP) ;; GET # OF FILLERS REQUIRED AND FILLER
(1)                               ;; CHARACTER.
(1)
(1) 002436 105366 000001      4$:    DECB    1(SP)   ;; DECREMENT FILLERS REQ. COUNT
(1) 002442 002770              BLT    3$         ;; BRANCH IF NO MORE FILLERS ARE REQUIRED
(1) 002444 004767 000002      JSR    PC,5$     ;; TYPE FILLER CHARACTER
(1) 002450 000772              BR
(1)
(1) 002452 105777 177642      5$:    TSTB    @STPS    ;; WAIT FOR OUTPUT DEVICE
(1) 002456 100375              BPL    -4
(1) 002460 122737 000017 002325  CMPB    #17,@$CNTRL0 ;; CHECK IF CONTROL 0 WAS TYPED
(1) 002466 001403              BEQ    6$        ;; STOP TYPING MESSAGE IF 10 WAS TYPED
(1) 002470 116677 000002 177624  MOVB    2(SP),@STPB  ;; OUTPUT CHARACTER
(1) 002476 122766 000015 000002  6$:    CMPB    #15,2(SP) ;; BRANCH IF NOT <CR>
(1) 002504 001003              BNE    7$
(1) 002506 105067 177612      CLRB    $CHARCNT  ;; CLEAR CHARACTERS TYPED COUNT
(1) 002512 000406              BR
(1) 002514 122766 000012 000002  7$:    CMPB    #12,2(SP)  ;; BRANCH IF <LF> OR 'NULL'
(1) 002522 002002              BGE    8$
(1) 002524 105267 177574      INCB    $CHARCNT  ;; INCREMENT CHARACTER TYPED COUNT
(1) 002530 000207      8$:    RTS    PC
(1)
(1)                               ;; HORIZONTAL TAB <HT> PROCESSER
(1) 002532 112716 000040      9$:    MOVB    #40,(SP) ;; LOAD 'SPACE'
(1) 002536 004767 177710      10$:   JSR    PC,5$      ;; TYPE 'SPACE'
(1) 002542 132767 000007 177554  BITB    #7,$CHARCNT ;; TYPE SPACES UNTIL A MULTIPLE
(1) 002550 001372              BNE    10$       ;; OF 8 CHARACTERS HAVE BEEN TYPED
(1) 002552 105726              TSTB    (SP)+    ;; POP SPACE
(1) 002554 000676              BR    TYPE1      ;; GET NEXT CHARACTER
435
436                               ; SUBROUTINE TO SAVE GENERAL REGISTERS ON THE STACK
437                               ; CALL:  SAVE
438 002556 010546      .SAVE:  MOV    R5,-(SP) ;SAVE REGISTERS ON THE STACK
439 002560 010446      MOV    R4,-(SP)
440 002562 010346      MOV    R3,-(SP)
441 002564 010246      MOV    R2,-(SP)
442 002566 010146      MOV    R1,-(SP)
443 002570 010046      MOV    R0,-(SP)
444 002572 016646 000014      MOV    14(SP),-(SP) ;GET RETURN PC
445 002576 000207      RTS    PC        ;RETURN
446
447                               ; SUBROUTINE TO RESTORE GENERAL REGISTERS FROM THE STACK
448                               ; CALL:  RESTORE
449 002600 012666 000014      .RESTORE:MOV (SP)+,14(SP) ;MOVE RETURN PC
450 002604 012600      MOV    (SP)+,R0  ;RESTORE REGISTERS
451 002606 012601      MOV    (SP)+,R1
452 002610 012602      MOV    (SP)+,R2
453 002612 012603      MOV    (SP)+,R3
454 002614 012604      MOV    (SP)+,R4
455 002616 012605      MOV    (SP)+,R5
456 002620 000207      RTS    PC        ;RETURN
457
458                               ; SUBROUTINE TO CONVERT OCTAL DATA TO ASCII

```

```

459      ;CALL: MOV    NUMBER,R2      ;MOVE NUMBER TO R2
460      ;      JSR    PC,CNVOC      ;
461      ;
462      002622 110667 176300      CNVOC: MOVB   SP,TYPFLG      ;SET DO NOT TYPE FLAG
463      002626 000402              BR      CNVTO
464      ;
465      .SBTTL      OCTAL TO ASCII & TYPE ROUTINE
466      ;SUBROUTINE TO CONVERT OCTAL NUMBER TO ASCII AND TYPE IT OUT
467      ;CALL: MOV    NUMBER,R2      ;PUT # IN R2
468      ;      JSR    PC,TYPEOCT     ;CALL ROUTINE
469      ;
470      002630 105037 001126      TYPEOCT: CLRB  @#TYPFLG      ;SET TYPE FLAG
471      002634      CNVTO:
472      (1) 002634 004767 177716      JSR    PC,.SAVE      ;SAVE REGISTERS ON THE STACK
473      002640 012704 001144      MOV    #ODIGITS,R4   ;SET PTR TO OUTPUT
474      002644 005003              CLR    R3            ;R3 WILL CONTAIN OCTAL DIGIT
475      002646 010201              MOV    R2,R1        ;GET # TO BE TYPED
476      002650 006302      1$: ASL    R2          ;SHIFT #
477      002654 012700 000006      ROL    R3
478      002660 000404              MOV    #6,R0        ;SET DIGIT COUNTER
479      ;
480      002662 006302      2$: ASL    R2          ;SHIFT # 3 PLACES LEFT
481      002664 006103              ROL    R3
482      002666 005301              DEC    R1
483      002670 001374              BNE    2$
484      002672 012701 000003      3$: MOV    #3,R1      ;SET SHIFT COUNTER
485      002676 116324 001132      MOVB   DIGTAB(R3),(R4)+ ;MOVE ASCII EQUIV TO OUTPUT
486      002702 005003              CLR    R3
487      002704 005300              DEC    R0            ;DECREMENT DIGIT COUNT
488      002706 001365              BNE    2$            ;GET NEXT DIGIT
489      002710 105737 001126      TSTB   @#TYPFLG      ;BRANCH IF ASCII IS
490      002714 001002              BNE    4$            ;NOT TO BE TYPED
491      002716 000004 001144      TYPE,ODIGITS.
492      002722      4$:
493      (1) 002722 004767 177652      JSR    PC,.RESTORE   ;RESTORE REGISTERS FROM THE STACK
494      002726 000207      RTS    PC
495      ;
496      ;SUBROUTINE TO CONVERT OCTAL DATA TO DECIMAL ASCII
497      ;CALL: MOV    NUMBER,R2      ;MOVE NUMBER TO R2
498      ;      JSR    PC,CNVDEC      ;
499      ;
500      002730 110637 001126      CNVDEC: MOVB   SP,@#TYPFLG  ;SET DO NOT TYPE FLAG
501      002734 000402              BR      CNVTD
502      ;
503      .SBTTL      OCTAL TO DECIMAL & TYPE ROUTINE
504      ;THIS ROUTINE CONVERTS AN OCTAL # TO DECIMAL ASCII AND TYPES IT OUT
505      ;CALL: MOV    NUMBER,R2      ;PUT # IN R2
506      ;      JSR    PC,TYPEDEC     ;CALL ROUTINE
507      002736 105037 001126      TYPEDEC: CLRB  @#TYPFLG   ;SET TYPE FLAG
508      002742      CNVTD:
509      (1) 002742 004767 177610      JSR    PC,.SAVE      ;SAVE REGISTERS ON THE STACK
510      002746 005000              CLR    R0            ;R0 IS INDEX TO DECIMAL CONSTANT
511      002750 012704 001144      MOV    #ODIGITS,R4   ;SET OUTPUT PTR
512      002754 005003      1$: CLR    R3            ;R3 CONTAINS DECIMAL DIGIT

```



```

512 002756 166002 003036 2$: SUB DCONST(R0),R2 ;SUBTRACT DECIMAL CONSTANT UNTIL
513 002762 103402 BLO 3$ ;INPUT * GOES NEGATIVE
514 002764 005203 INC R3 ;KEEPING TRACK OF SUBTRACTIONS
515 002766 000773 BR 2$
516 002770 066002 003036 3$: ADD DCONST(R0),R2 ;ADD BACK CONSTANT WHEN NEGATIVE
517 002774 116324 001132 MOVB DIGTAB(R3),(R4)+ ;MOVE ASCII EQUIVALENT
518 003000 062700 000002 ADD #2,R0 ;NEXT CONSTANT
519 003004 005760 003036 TST DCONST(R0) ;UNTIL ALL CONSTANTS DONE
520 003010 001361 BNE 1$
521 003012 112724 000060 MOVB #'0,(R4)+ ;LAST DIGIT IS 0
522 003016 105737 001126 TSTB @#TYPFLG ;BRANCH IF ASCII IS
523 003022 001002 BNE 4$ ;NOT TO BE TYPED
524 003024 000004 001144 TYPE,ODIGITS
525 003030
(1) 003030 004767 177544 4$: JSR PC,.RESTORE ;RESTORE REGISTERS FROM THE STACK
526 003034 000207 RTS PC
527
528 003036 023420 DCONST: .WORD 10000.
529 003040 001750 .WORD 1000.
530 003042 000144 .WORD 100.
531 003044 000012 .WORD '0.
532 003046 000001 .WORD 1.
533 003050 000000 .WORD 0 ;TERMINATOR

```

```

534
535 .SBTTL TYPE SPECIFIED TIMES ROUTINE
536 ;THIS SUBROUTINE OUTPUTS THE TIME SPECIFICATIONS FOR THE TEST
537 ;AND ALSO THE ACTUAL TIME RECORDED (ATIME)
538 ;FORMAT OF LINE TYPED
539 ;RANGE=<AAAAAA-BBBBBB> ACTUAL=CCCCC
540 ;WHERE: AAAAAA IS MAXIMUM TIME FOR TEST (STIMTBL(TSTNUMX4)).
541 ;BBBBBB IS MINIMUM TIME FOR TEST (STIMTBL(TSTNUMX4+2)).
542 ;CCCCC IS ACTUAL TIME RECORDED BY TEST (ATIME).
543 ;CALL: MOVB TEST NUMBER,R2 ;LOAD TEST NUMBER
544 ;MOV #TIME,@#ATIME ;MOVE TIME TO ATIME
545 ;JSR PC,OUTSPC
546 003052 010246 OUTSPC: MOV R2,-(SP) ;SAVE R2 & R3 ON THE STACK
547 003054 010346 MOV R3,-(SP)
548 003056 006302 ASL R2 ;MULTIPLY TEST # TIMES 4
549 003060 006302 ASL R2 ;TO FORM INDEX INTO STIMTBL
550 003062 010203 MOV R2,R3 ;R3 CONTAINS INDEX INTO TABLE
551 003064 000004 014647 TYPE,L.RNG
552 003070 016302 001416 MOV STIMTBL(R3),R2 ;GET MAXIMUM SPEC TIME
553 003074 004767 177636 JSR PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
554 003100 000004 001405 TYPE,DASH
555 003104 016302 001420 MOV STIMTBL+2(R3),R2 ;GET MINIMUM TIME
556 003110 004767 177622 JSR PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
557 003114 000004 001412 TYPE,ANGTAB
558 003120 000004 014657 TYPE,L.ACT
559 003124 013702 001012 MOV @#ATIME,R2 ;GET ACTUAL TIME
560 003130 004767 177602 JSR PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
561 003134 000004 001374 TYPE,CRLF
562 003140 012603 MOV (SP)+,R3
563 003142 012602 MOV (SP)+,R2
564 003144 000207 RTS PC ;RETURN

```

.SBTTL TYPE GAP TIMES SUBROUTINE

```

567 ; THIS SUBROUTINE IS USED TO TYPE THE SPECIFIED GAP SIZES (RECORDED IN
568 ; TST021). IT IS CALLED BY THE GAPOK ROUTINE IF THE GAP SIZE IS OUT OF
569 ; RANGE VIA THE HLT ROUTINE (HLT+2).
570 ; CALL:  MOVB  #GAP,GAP      ; LOAD GAP # INTO GAP
571         MOV   #TIME,ATIME   ; LOAD ACTUAL TIME INTO ATIME
572         JSR   PC,OUTGAP
573
574 OUTGAP: MOV   R2,-(SP)      ; SAVE R2 AND R3
575         MOV   R3,-(SP)
576         MOVB  GAP,R3      ; GET GAP #
577         ASL   R3
578         ASL   R3
579         TYPE,L.RNG
580         MOV   GTIMTBL(R3),R2 ; GET MAX TIME
581         JSR   PC,TYPDEC    ; CONVERT TO DECIMAL & TYPE
582         TYPE,DASH
583         MOV   GTIMTBL+2(R3),R2 ; GET MIN TIME
584         JSR   PC,TYPDEC    ; CONVERT TO DECIMAL & TYPE
585         TYPE,ANGTAB
586         TYPE,L.ACT
587         MOV   @#ATIME,R2    ; GET ACTUAL TIME
588         JSR   PC,TYPDEC    ; CONVERT TO DECIMAL & TYPE
589         TYPE,E.GAP
590         MOVB  @#GAP,R2     ; GET GAP #
591         JSR   PC,TYPOCT    ; TYPE GAP #
592         TYPE,CRLF
593         MOV   (SP)+,R3     ; RESTORE R3 AND R2
594         MOV   (SP)+,R2
595         RTS   PC
596
597 .SBTTL      ASCII TO OCTAL CONVERT SUBROUTINE
598 ; SUBROUTINE TO CONVERT ASCII DATA TO OCTAL. CONVERTED OCTAL DATA
599 ; IS LEFT IN OCTALO <15-00>.
600 CNVTAO:
601         JSR   PC,SAVE      ; SAVE REGISTERS ON THE STACK
602         MOV   #INBUF,R0    ; SET PTR TO ASCII DATA
603         MOV   #OCTALO,R1   ; GET ADDRESS OF OCTAL DATA
604         CLR   (R1)         ; CLEAR OUT OLD OCTAL DATA
605         CLR   2(R1)
606 1$:      CMPB  #CR,(R0)    ; <CR> TERMINATES INPUT
607         BEQ   3$
608         MOVB (R0)+,R2     ; GET 'OCTAL' DATA
609         BIC  #177770,R2   ; STRIP UNUSED BITS
610         MOV  #3,R3        ; SET SHIFT COUNT
611 2$:      ASL  (R1)         ; SHIFT LAST
612         ROL  2(R1)       ; OCTAL DIGIT
613         DEC  R3
614         BNE  2$
615         BIS  R2,(R1)     ; AND INSERT THIS DIGIT
616         BR   1$         ; GO GET NEXT DIGIT
617 3$:      JSR   PC,.RESTORE ; RESTORE REGISTERS FROM THE STACK
618         RTS   PC        ; RETURN
619
620 .SBTTL      PUBLISH SUBROUTINE
        ; THE PUBLISH SUBROUTINE AVERAGES THE RECORDED TIMES FOR EACH TEST IT-
    
```

```

621 ;ERATION (IF 16. ITERATIONS) AND PLACES THE AVERAGE RESULT IN 'ATIME'.
622 ;IT TYPES THE NAME OF THE FUNCTION THAT WAS TIMED,THE TIME SPEC-
623 ;IFICATION AND THE ACTUAL TIME .
624
625 PUBLISH:
(1) 003346 004767 177204 JSR PC,SAVE ;SAVE REGISTERS ON THE STACK
626 003352 012700 001014 MOV #ATIMTBL,R0 ;GET TABLE ADDRESS CONTAINING TIMES
627 003356 113701 001121 MOVB @#ITCNT,R1 ;GET # OF ENTRIES (GIVEN BY ITERATION COUNT)
628 003362 122701 000001 CMPB #1,R1 ;BRANCH IF SINGLE ITERATION
629 003366 001423 BEQ 4$
630 003370 005002 CLR R2 ;CLEAR 'SUM' REGISTERS
631 003372 005003 CLR R3
632 003374 122701 000020 CMPB #16.,R1 ;BRANCH IF 16. ITERATIONS
633 003400 001402 BEQ 1$
634 003402 000000 HALT ;ITERATION COUNT MUST BE 1 OR 16.
635 003404 000777 BR . ;DO NOT CHANGE POSIT OF SW11
636 ;WHEN TEST IS RUNNING.
637
638 003406 062002 1$: ADD (R0)+,R2 ;SUM INDIVIDUAL TIMES
639 003410 005503 ADC R3
640 003412 005301 DEC R1
641 003414 001374 BNE 1$
642
643 003416 012700 000004 2$: MOV #4,R0
644 003422 006203 3$: ASR R3 ;SHIFT TIME IN R3 & R2 4 PLACES
645 003424 006002 ROR R2 ;RIGHT = DIVIDE BY 16.
646 003426 005300 DEC R0
647 003430 001374 BNE 3$
648 003432 010237 001012 MOV R2,@#ATIME ;MOVE AVERAGED TIMES
649
650 003436 113700 001122 4$: MOVB @#TSTNUM,R0 ;GET TEST #
651 003442 006300 ASL R0
652 003444 016067 001672 000002 MOV NAMPTR(R0),5$ ;GET TEST NAME STRING ADDRESS
653 003452 000004 TYPE
654 003454 000000 5$: .WORD 0
655 003456 113702 001122 MOVB @#TSTNUM,R2 ;GET TEST #
656 003462 004767 177364 JSR PC,OUTSPC ;OUTPUT TIMES
657 003466 004767 177106 JSR PC,.RESTORE ;RESTORE REGISTERS FROM THE STACK
658 003472 000207 RTS PC
659
660 .SBTTL INPUT SUBROUTINE
661 ;SUBROUTINE TO GET TTY INPUT
662 ;CALL: JSR PC,.INPUT
663 ;INPUT DATA IS RETURNED IN BUFFER BEGINNING AT INBUF.
664
665 003474 010046 .INPUT: MOV R0,-(SP) ;SAVE R0 ON THE STACK
666 003476 012700 001264 1$: MOV #INBUF,R0
667 003502 105737 177560 2$: TSTB @#TKS
668 003506 100375 BPL 2$
669
670 003510 113746 177562 MOVB @#TKB,-(SP) ;GET CHARACTER
671 003514 042716 000200 BIC #200,(SP)
672 003520 122716 000177 CMPB #177,(SP) ;CHECK RUBOUT
673 003524 001004 BNE 3$
674 003526 124026 CMPB -(R0),(SP)+ ;REMOVE CHARACTER FROM INPUT
675 003530 000004 001377 TYPE,BKSLSH
    
```

```

676 003534 000762
677 003536 122716 000025      3$:  BR      2$          ;WAIT FOR NEXT CHARACTER
678 003542 001004          ;CHECK CONTROL U (↑U)
679 003544 005726          BNE     4$
680 003546 000004 001374      TST     (SP)+
681 003552 000751          TYPE,CRLF
682 003554 111637 001401      4$:  BR      1$
683 003560 111620          MOVB    (SP),@#ECHO
684 003562 122726 000015      MOVB    (SP),(RO)+
685 003566 001403          CMPB    #CR,(SP)+
686 003570 000004 001401      BEQ     5$
687 003574 000742          TYPE,ECHO
688 003576 000004 001374      5$:  BR      2$
689 003602 012600          TYPE,CRLF
690 003604 000207          MOV     (SP)+,RO
691
692
693 003606 113746 177562      ;KEYBOARD INTERRUPT SERVICE ROUTINE
694 003612 042716 000200      TKISR: MOVB   @#TKB,-(SP) ;GET TYPED CHARACTER
695 003616 122716 000017      BIC     #200,(SP) ;STRIP PARITY BIT
696 003622 001003          CMPB    #CNTRL0,(SP) ;BRANCH IF NOT CONTROL 0 (↑0)
697 003624 112667 176475      BNE     1$
698 003630 000002          MOVB    (SP)+,#CNTRL0 ;SET CONTROL 0 INDICATOR IN TYPE ROUTINE
699
700 003632 122726 000003          RTI ;EXIT
701 003636 001003          1$:  CMPB    #3,(SP)+ ;BRANCH IF NOT CONTROL C (↑C)
702 003640 000005          BNE     2$
703 003642 000137 005724          RESET
704 003646 000002          2$:  JMP     @#INIT ;RESTART PROGRAM
          RTI ;EXIT

```

```

706 .SBTTL ERROR SERVICE ROUTINES
707 ;ROUTINE TO PROCESS ERROR TRAPS (TRAPS TO 4)
708 003650 000000 ERRTRP: HALT
709
710 ;ERROR SERVICE ROUTINE
711 ;THIS ROUTINE PROCESSES TWO TYPES OF ERRORS (OUT OF RANGE AND HARDWARE)
712 ;THE CALLS FOR AN OUT OF RANGE ERROR ARE <HLT+1>,<HLT+2> AND, FOR A
713 ;HARDWARE ERROR THE CALL IS <HLT>.
714
715 003652 004767 176112 .HLT: JSR PC,CKSWR ;CHECK FOR CNTL G
716 003656 004767 176674 JSR PC,.SAVE ;SAVE REGISTERS ON THE STACK
717 003662 110637 001123 1$: MOVB SP,@#ERFLG ;SET ERROR FLAG
718 003666 032777 020000 175104 BIT #SW13,@SWR ;BRANCH IF NO TYP0UT
719 003674 001075 BNE 4$
720 003676 000004 014137 TYPE,E.HDR
721 003702 113702 001122 MOVB @#TSTNUM,R2 ;GET TEST #
722 003706 004767 176716 JSR PC,TYPOCT ;AND TYPE IT
723 003712 016600 000016 MOV 16(SP),R0 ;GET RETURN PC
724 003716 162700 000002 SUB #2,R0 ;NOW PC OF HLT CALL
725 003722 111000 MOVB (R0),R0 ;NOW HLT CALL ITSELF
726 003724 001417 BEQ 2$ ;BRANCH IF HLT
727 003726 000004 014222 TYPE,E.HDR2
728 003732 122700 000002 CMPB #2,R0 ;BRANCH IF NOT HLT+2
729 003736 001005 BNE 10$
730 003740 004767 177202 JSR PC,OUTGAP ;TYPE GAP SPECIFIED TIMES
731 003744 000004 001374 TYPE,CRLF
732 003750 000447 BR 4$
733 003752 004767 177074 10$: JSR PC,OUTSPC ;TYPE SPECIFIED TIMES
734 003756 000004 001374 TYPE,CRLF
735 003762 000442 BR 4$
736 003764 016500 000014 2$: MOV ER(R5),R0
737 003770 032765 002000 000032 BIT #PE1600,TC(R5)
738 003776 001403 BEQ 20$
739 004000 042700 102100 BIC #102100,R0
740 004004 000402 BR 21$
741 004006 042700 102300 20$: BIC #102300,R0
742 004012 005700 21$: TST R0
743 004014 001003 BNE 22$
744 004016 000004 014113 TYPE,E.SFT ;TYPE SOFT ERROR MESSAGE
745 004022 000434 BR 6$
746
747 004024 000004 014147 22$: TYPE,E.HDR1
748 004030 010500 MOV R5,R0 ;GET FIRST ADDRESS OF REGS.
749 004032 012701 000007 MOV #7,R1 ;TYPE FIRST 7 REGS.
750 004036 012002 3$: MOV (R0)+,R2 ;GET REG CONTENTS
751 004040 004767 176564 JSR PC,TYPOCT ;AND TYPE IT
752 004044 000004 001407 TYPE,SPACE2
753 004050 005301 DEC R1
754 004052 001371 BNE 3$
755 004054 016502 000032 MOV TC(R5),R2 ;GET CONTENTS OF TC REGISTER
756 004060 004767 176544 JSR PC,TYPOCT
757 004064 000004 001374 TYPE,CRLF
758
759 004070 032777 001000 174702 4$: BIT #SW09,@SWR ;BRANCH IF NO RING THE BELL
760 004076 001402 BEQ 5$
761 004100 000004 001403 TYPE,BELL
    
```

G04

DZTLG-A TMO2/TU16J DRIVE FUNCTION TIMER MACY11 27(1006) 09-DEC-76 10:45 PAGE 2-8  
DZTCGR.P11 08-DEC-76 12:39 ERROR SERVICE ROUTINES

SEQ 0045

762	004104	005777	174670	5\$:	TST	QSWR	:HALT ON ERRCP?
763	004110	100001			BPL	6\$	
764	004112	000000			HALT		
765	004114	0047F7	175650	5\$:	JSR	PC.CKSWR	.CHECK FOR CNTL G
766	004120	004767	176454		JSR	PC..RESTORE	:RESTORE REGISTERS FROM THE STACK
767	004124	000002			RTI		:RETURN

```

771          SBTTL          SCOPE SUBROUTINE
772          :SCOPE ROUTINE
773          :THIS ROUTINE IS ENTERED UPON COMPLETION OF EACH SUBTEST
774          :THE SCOPE ROUTINE:
775          :   OUTPUTS TIME SPEC ON EACH ITERATION IF SW08 IS SET
776          :   REPEATS TEST IF SW14 IS SET
777          :   STORES ACTUAL TIME FOR FUNCTION IN TIME TABLE (ATIMTBL)
778          :   PUBLISHES TIME IF SW10=0
779          :   UPDATES ITERATION COUNT AND IF ITERATIONS COMPLETE CONTINUES
780          :   TO NEXT TEST, OTHERWISE REPEATS TEST.
781          :   DELAYS BEFORE CONTINUING OR REPEATING TEST.
782          :   INITIALIZES DRIVE
783          :RETURNS:      R5=BASE ADDRESS OF TMO2 REGISTERS (ADDRESS OF CS1)
784          :              R1='DS' REG ADDRESS
785          :              R0='FC' REG ADDRESS
786
787          .SCOPE: JSR      PC,CKSWR          ;CHECK FOR CNTL G
788                  MOV      @#TMBASE,R5      ;SET R5 TO FIRST TM REG
789                  BIT      #SW08,@SWR       ;BRANCH IF SPECIFICATION LINE
790                  BEQ      10$              ;NOT DESIRED ON EACH ITERATION
791                  MOVB     @#TSTNUM,R2      ;GET TEST NUMBER
792                  JSR      PC,OUTSPC        ;OUTPUT TIME RECORDED
793                  BIT      #SW14,@SWR       ;BRANCH IF CONTINUOUS LOOP
794                  BEQ      2$              ;NOT DESIRED
795                  JSR      PC,DELAY         ;DELAY 350 MS
796                  JSR      PC,RHINIT        ;INIT
797                  CLRB     @#ERFLG         ;CLEAR ERROR FLAG
798                  MOV      SCPADR,(SP)
799                  MOV      R5,R1
800                  ADD      #DS,R1          ;ADDRESS OF 'DS' REG IS IN R1
801                  MOV      R5,R0
802                  ADD      #FC,R0          ;ADDRESS OF 'FC' REG IS IN R0
803                  RTI
804
805                  TSTB     @#ERFLG          ;BRANCH IF ERROR FLAG IS SET
806                  BNE      3$
807                  MOVB     @#ITCNT,R0      ;GET ITERATION COUNT
808                  ASL      R0              ;STORE TIME IN TABLE
809                  MOV      @#ATIME,ATIMTBL(R0)
810                  INCB     @#ITCNT          ;INCREMENT ITERATION COUNT
811                  BIT      #SW11,@SWR       ;BRANCH IF SINGLE ITERATION DESIRED
812                  BNE      4$
813                  CMPB     #16,@#ITCNT     ;BRANCH IF ITERATIONS INCOMPLETE
814                  BNE      1$
815                  MOV      (SP),@#SCPADR   ;SET SCOPE ADDRESS TO NEXT TEST
816                  BIT      #SW10,@SWR       ;BRANCH IF NO PUBLICATION DESIRED
817                  BNE      5$
818                  JSR      PC,PUBLISH       ;GO PUBLISH TEST DATA
819                  CLRB     @#ITCNT         ;RESET ITERATION COUNT
820                  TSTB     @SWR            ;BRANCH IF USER DOES NOT WANT TO
821                  BEQ      1$              ;HALT ON A SELECTED TEST
822                  MOV      @SWR,-(SP)      ;GET SWITCHES
823                  BIC      #177740,(SP)    ;CLEAR ALL BUT TEST #
824                  DEC      (SP)           ;FORM TEST # -1
825                  CMPB     @#TSTNUM,(SP)+  ;BRANCH IF NOT AT TEST
826                  BNE      1$

```

```

827 004344 000000          HALT
828 004346 004767 175416    JSR      PC,CKSWR          ;CHECK FOR CNTL G
829 004352 000705          BR       1$
830
831          .SBTTL  TIMER SUBROUTINES
832
833          ;SUBROUTINE TO SYNCHRONIZE THE TIMER AND TURN IT ON.
834          ;REGISTER 4 IS CLEARED, AND THE OSCILLATOR POLARITY IS MONITORED
835          ;THE ROUTINE IS EXITED WHEN THE OSCILLATOR POLARITY CHANGES WITH R3
836          ;SET TO INDICATE THE POLARITY OF THE OSCILLATOR.
837          ;CALL:  JSR      PC,TIMON
838          ;RETURNS: R3 SET TO INDICATE LAST POLARITY (+24/-24=0/1)
839          ;          R4 = 0
840
841 004354 005004          TIMON: CLR      R4          ;CLEAR TIME COUNT
842 004356 012703 000024    MOV      #24,R3         ;SET POLARITY TO '0' STATE
843 004362 032765 000100 000024  BIT      #OSC,MR(R5)    ;BRANCH IF POLARITY IS '0'
844 004370 001405          BEQ      2$
845 004372 032765 000100 000024  1$: BIT      #OSC,MR(R5)    ;WAIT FOR OSCILLATOR TO RETURN
846 004400 001374          BNE      1$
847 004402 000405          BR       4$
848
849 004404 005403          2$:  NEG      R3          ;NEGATE PREV POLARITY INDICATOR
850 004406 032765 000100 000024  3$:  BIT      #OSC,MR(R5)    ;WAIT FOR OSCILLATOR TO RETURN
851 004414 001774          BEQ      3$          ;TO '1' STATE
852 004416 000207          4$:  RTS      PC
853
854          ;SUBROUTINE TO COUNT TIME
855          ;EACH TIME THE OSCILLATOR TOGGLES (BIT <06> IN MR REG) REGISTER
856          ;R4 IS INCREMENTED, AND THE REGISTER R3 IS NEGATED TO INDICATE
857          ;THE LAST STATE OF THE OSCILLATOR.
858          ;CALL  JMP      TIMER(R3)          ;R3 IS SET BY TIMON ROUTINE
859          ;          R2=RETURN ADDRESS TO CALLER
860          ;NOTE: THE TIME TO EXECUTE THIS ROUTINE IS VERY CRITICAL. IT MUST BE
861          ;LESS THAN 40 US.
862
863          ;ENTER HERE VIA JMP  TIMER(R3) WHEN R3=-24 (PREV STATE=1)
864 004420 032765 000100 000024  TIMER1: BIT      #OSC,MR(R5)    ;BRANCH IF CURRENT STATE IS '0'
865 004426 001406          BEQ      TIMER          ;GO INCREMENT TIME
866 004430 000112          JMP      (R2)          ;RETURN TO TEST
867
868          .=TIMER1+24
869 004444 005403          TIMER: NEG      R3          ;NEGATE PREV STATE INDICATOR
870 004446 005204          INC      R4          ;INCREMENT 'TICK' COUNT
871 004450 100401          BMI      TIMERR        ;BRANCH ON OVERFLOW
872 004452 000112          JMP      (R2)          ;RETURN TO TEST
873 004454 000004 014250    TIMERR: TYPE,E.TIMOV      ;TYPE 'TIMER OVERFLOWED'
874 004460 104400          HLT
875 004462 000177 174314    JMP      @SCPADR        ;REPORT HARDWARE ERROR
876
877          .=TIMER+24
878          ;ENTER HERE VIA JMP  TIMER(R3) WHEN R3=+24 (PREV STATE=0)
879 004470 032765 000100 000024  TIMER0: BIT      #OSC,MR(R5)    ;BRANCH IF CURRENT STATE = '1'
880 004476 001362          BNE      TIMER          ;GO INCREMENT TIME
881 004500 000112          JMP      (R2)
882

```



```

883 ;SUBROUTINE TO CHECK TIME RECORDED BY SUBTEST.
884 ;THIS SUBROUTINE COMPUTES THE ACTUAL TIME (IN MICROSECONDS) AND CHECKS
885 ;THAT THE TIME RECORDED BY THE SUBTEST IS CORRECT BY COMPARING THE TIME
886 ;WITH THE HIGH LIMIT (STIMTBL(RO)) AND THE LOW LIMIT (STIMTBL+2(RO)).
887 ;IF THE TIME IS OUT OF RANGE AN OUT OF RANGE ERROR TYPEOUT RESULTS.
888 ;THE SUBROUTINE IS ENTERED WITH:
889 ; R4=TICK COUNT
890
891 TIMOK:
(1) 004502 004767 176050 JSR PC, .SAVE ;SAVE REGISTERS ON THE STACK
892 004506 012700 000070 MOV #56, R0 ;GET TIME PER TICK
893 004512 010401 MOV R4, R1 ;GET TICKS COUNT
894 004514 005002 CLR R2 ;CLEAR SUMMING REGISTERS
895 004516 005003 CLR R3
896 004520 060002 1$: ADD R0, R2 ;MULTIPLY TIME PER TICK
897 004522 005503 ADC R3 ;BY TICK COUNT
898 004524 005301 DEC R1
899 004526 001374 BNE 1$
900 004530 010246 MOV R2, -(SP) ;DIVIDE COUNT BY 10.
901
902 004532 010346 MOV R3, -(SP)
903 004534 012746 000012 MOV #10, -(SP)
904 004540 004767 000262 JSR PC, DIVIDE
905 004544 005726 TST (SP)+ ;DISCARD REMAINDER
906 004546 012637 001012 MOV (SP)+, @#ATIME ;STORE QUOTIENT
907 004552 113700 001122 MOVB @#TSTNUM, R0 ;GET TEST #
908 004556 006300 ASL R0
909 004560 006300 ASL R0
910 004562 023760 001012 001416 CMP @#ATIME, STIMTBL(RO) ;CHECK THAT TIME IS WITHIN
911 004570 101004 BHI 2$ ;LIMITS SPECIFIED
912 004572 023760 001012 001420 CMP @#ATIME, STIMTBL+2(RO)
913 004600 101001 BHI 3$
914 004602 104401 2$: HLT+1 ;CALL ERROR ROUTINE
915 3$:
(1) 004604 004767 175770 JSR PC, .RESTORE ;RESTORE REGISTERS FROM THE STACK
916 004610 000207 RTS PC ;RETURN
917
918 ;SUBROUTINE TO CHECK INDIVIDUAL GAP TIMES (PRODUCED BY TST021)
919 ;SUBROUTINE COMPUTES THE ACTUAL TIME (IN MICROSECONDS) AND CHECKS
920 ;THAT THE GAP TIME RECORDED BY THE SUBTEST (TST021) BY COMPARING THE
921 ;TIME WITH THE MAX LIMIT (GTIMTBL-GAPTBL(R1)) AND THE MIN LIMIT
922 ;((GTIMTBL+2-GAPTBL(R1))).
923 ;CALL: MOV #TICK COUNT, R4 ;R4 CONTAINS TICK COUNT
924 ; MOVB #GAP, @#GAP ;LOCATION GAP CONTAINS GAP #
925 ; JSR PC, GAPOK
926
927 GAPOK:
(1) 004612 004767 175740 JSR PC, .SAVE ;SAVE REGISTERS ON THE STACK
928 004616 012700 000070 MOV #56, R0 ;GET TIME PER TICK
929 004622 010401 MOV R4, R1 ;GET TICK COUNT
930 004624 005002 CLR R2 ;CLEAR SUMMING REGISTERS
931 004626 005003 CLR R3
932 004630 060002 1$: ADD R0, R2 ;MULTIPLY TICK COUNT
933 004632 005503 ADC R3 ;BY TIME PER TICK
934 004634 005301 DEC R1
935 004636 001374 BNE 1$
    
```

```

936
937 004640 010246      MOV      R2,-(SP)      ;DIVIDE TIME BY 10.
938 004642 010346      MOV      R3,-(SP)
939 004644 012746 000012  MOV      #10,-(SP)
940 004650 004767 000152  JSR      PC,DIVIDE
941 004654 005726      TST      (SP)+        ;DISCARD REMAINDER
942 004656 012637 001012  MOV      (SP)+,a#ATIME ;STORE QUOTIENT
943 004662 113703 001120  MOV      a#GAP,R3     ;GET GAP #
944 004666 006303      ASL      R3           ;MULTPLY BY 4
945 004670 006303      ASL      R3           ;TO GET AT TABLE ENTRY
946 004672 023763 001012 001572  CMP      a#ATIME,GTIMTBL(R3) ;CHECK TIME (MAX)
947 004700 101004      BHI      2$
948 004702 023763 001012 001574  CMP      a#ATIME,GTIMTBL+2(R3) ;CHECK TIME (MIN)
949 004710 101002      BHI      3$
950 004712 104402      HLT+2          ;REPORT OUT OF RANGE ERROR
951 004714 000406      BR       100$
952 004716 032777 000400 174054 3$:      BIT      #SW08,aSWR   ;BRANCH IF TIMES NOT WANTED
953 004724 001402      BEQ      100$
954 004726 004767 176214      JSR      PC,OUTGAP   ;TYPE GAP TIMES
955
956 004732      100$:
(1) 004732 004767 175642      JSR      PC,.RESTORE ;RESTORE REGISTERS FROM THE STACK
957 004736 000207      RTS      PC         ;RETURN TO TEST
958
959      .SBTTL          DELAY SUBROUTINES
960      ;THIS SUBROUTINE CAUSES A DELAY OF 350 MS.
961 004740 004767 177410  DELAY:  JSR      PC,TIMON
962 004744 010246      MOV      R2,-(SP)   ;SAVE R2 ON THE STACK
963 004746 012702 004756      MOV      #2$,R2    ;SET RETURN ADDRESS FOR TIMER
964 004752      1$:
(1) 004752 000163 004444      JMP      TIMER(R3) ;GO TO TIMER & RETURN VIA R2
965 004756 032704 004000 2$:      BIT      #4000,R4
966 004762 001773      BEQ      1$
967 004764 012602      MOV      (SP)+,R2  ;RESTORE R2
968 004766 000207      RTS      PC
969
970      ;THIS SUBROUTINE ALLOWS A CALLER SPECIFIED DELAY (UP TO 65MS.)
971      ;CALL:  MOV      DELAY TIME,DELTIM ;LOAD DELAY TIME (IN US)
972      DELAYV: JSR      PC,DELAYV
973 004770 005767 174120  DELAYV: TST      DELTIM   ;BRANCH IF 0 DELAY
974 004774 001413      BEQ      3$
975 004776 004767 177352      JSR      PC,TIMON  ;TURN TIMER ON
976 005002 010246      MOV      R2,-(SP)  ;SAVE R2 ON THE STACK
977 005004 012702 005014      MOV      #2$,R2   ;SET RETURN ADDRESS FROM TIMER
978 005010      1$:
(1) 005010 000163 004444      JMP      TIMER(R3) ;GO TO TIMER & RETURN VIA R2
979 005014 023704 001114 2$:      CMP      a#DELTIM,R4
980 005020 101373      BHI      1$
981 005022 012602      MOV      (SP)+,R2  ;RESTORE R2
982 005024 000207      3$:      RTS      PC
983
984      .SBTTL          DIVIDE SUBROUTINE
985      ;THIS SUBROUTINE DIVIDES A DOUBLE PRECISION # AND RETURNS THE RESULT
986      ;TO THE CALLER ON THE STACK. BOTH DIVIDEND & DIVISOR MUST BE POSITIVE.
987      ;CALL:  MOV      LEAST SIGNIFICANT HALF DIVIDEND,-(SP)
988      ;      MOV      #MOST SIGNIFICANT HALF DIVIDEND,-(SP)

```

```

989      :      MOV      #DIVISOR, -(SP)
990      :      JSR      PC, DIVIDE
991      : RETURN
992      :      (SP)=REMAINDER ON STACK
993      :      2(SP)=QUOTIENT
994
995      ;NOTE: THIS SUBROUTINE DESTROYS PREVIOUS CONTENTS OF R0,R1,R2 & R3.
996
997      DIVIDE: CLR      -(SP)          ;SAVE LOC FOR SIGNS
998      :      MOV      #17, -(SP)      ;SET ITERATION COUNT
999      :      MOV      12(SP), R1       ;GET LSH DIVIDEND
1000     :      MOV      10(SP), R0      ;GET MSH DIVIDEND
1001     :      MOV      6(SP), R2       ;GET DIVISOR
1002     :      NEG      R2              ;NEGATE DIVISOR
1003     :      CLC
1004     :      BR      2$              ;CLEAR 'C' BIT IN PSW
1005     :      ROL      R0              ;ROTATE MSH DIVIDEND
1006     1$:  MOV      R0, R3           ;SAVE IN R3
1007     :      ADD      R2, R3         ;SUBTRACT DIVISOR FROM MSH DIVIDEND
1008     :      BCC     2$              ;BRANCH IF DIVIDEND > DIVISOR
1009     :      MOV      R3, R0         ;SAVE REMAINDER IN R0
1010     2$:  ROL      R1              ;ROTATE LSH DIVIDEND
1011     :      DEC      (SP)           ;DECREMENT ITERATION COUNT
1012     :      BNE     1$
1013     :      TST      (SP)+          ;POP ITERATION COUNTER
1014     :      TST      (SP)+          ;POP SIGN CORRECTION
1015     :      MOV      R1, 6(SP)      ;PUSH REMAINDER ON STACK
1016     :      MOV      R0, 4(SP)      ;PUSH QUOTIENT ONTO STACK
1017     :      MOV      (SP)+, (SP)

```

```

1019 005114 000207          RTS      PC
1020
1021          SBTTL  DRIVE SUBROUTINES
1022          ;SUBROUTINE TO CHECK IF DRIVE IS AVAILABLE
1023          ;CALL:  MOVB  DRIVE#,DRVNUM
1024          ;        JSR   PC,DRVAVA
1025          ;RETURN:  'C' BIT SET IF NOT AVAILABLE
1026 005116 113765 001004 000010 DRVAVA: MOVB  @#DRVNUM,CS2(R5) ;LOAD DRIVE #
1027 005124 032765 040000 000026          BIT   #TAP,DT(R5) ;CHECK IF TAPE UNIT
1028 005132 001003          BNE   1$
1029 005134 004767 000034          JSR   PC,RHINIT
1030 005140 000262          SEV   ;SET 'V' TO IND NOT AVAIL
1031 005142 000207          1$:   RTS      PC ;RETURN
1032
1033          ;SUBROUTINE TO CHECK IF TU16 SLAVE IS AVAILABLE FOR TEST
1034          ;CALL:  MOVB  DRIVE #,@#DRVNUM ;PASS DRIVE # VIA DRVNUM
1035          ;        MOVB  SLAVE #,@#SLVNUM ;PASS SLAVE # VIA SLVNUM
1036          ;        JSR   PC,SLVAVA ;CALL SUBROUTINE
1037 005144 113765 001004 000010 SLVAVA: MOVB  @#DRVNUM,CS2(R5) ;LOAD DRIVE #
1038 005152 113765 001005 000032          MOVB  @#SLVNUM,TC(R5) ;AND SLAVE #
1039 005160 032765 002000 000026          BIT   #SPR,DT(R5) ;BRANCH IF SLAVE PRESENT
1040 005166 001001          BNE   1$
1041 005170 000262          SEV   ;SET 'V' TO INDICATE NO SLAVE
1042 005172 000207          1$:   RTS      PC
1043
1044          ;SUBROUTINE TO INITIALIZE RH CONTROLLER
1045          ;CALL:  JSR   PC,RHINIT
1046
1047 005174 012765 000040 000010 RHINIT: MOV   #40,CS2(R5)
1048 005202 113765 001004 000010          MOVB  @#DRVNUM,CS2(R5)
1049 005210 005046          CLR   -(SP)
1050 005212 113716 001005          MOVB  @#SLVNUM,(SP)
1051 005216 012665 000032          MOV   (SP)+,TC(R5) ;LOAD SLAVE # INTO TC REG
1052 005222 052765 000300 000032          BIS   #NORM11,TC(R5)
1053 005230 000207          RTS      PC
1054
1055          ;SUBROUTINE TO WAIT FOR DRIVE READY (DRY)
1056 005232 005027 WAITRDY: CLR   (PC)+ ;CLEAR WAIT TIMER
1057 005234 000000 WAITTIM: .WORD 0
1058 005236 105765 000012          1$:   TSTB  DS(R5) ;WAIT FOR READY TO SET
1059 005242 100406          BMI   2$
1060 005244 005267 177764          INC   WAITTIM ;INCREMENT WAIT TIMER
1061 005250 001372          BNE   1$ ;BRANCH IF TIME HAS NOT EXPIRED
1062 005252 000004 014275          TYPE,E.TIMEXP ;TYPE 'TIME EXPIRED WAITING FOR RDY'
1063 005256 000425          BR   99$ ;TAKE ERROR EXIT
1064 005260 032765 002000 000012 2$:   BIT   #EOT,DS(R5) ;CHECK FOR END OF TAPE
1065 005266 001415          BEQ   3$ ;BRANCH IF NO EOT
1066 005270 000004 013330          TYPE,M.NAM ;TYPE 'END OF TAPE'
1067 005274 000004 013657          TYPE,M.EOT ;REWIND SLAVE
1068 005300 004767 000032          JSR   PC,.REWIND ;BRANCH IF ERROR ON REWIND
1069 (1) 005304 102412          BVS   99$ ;WRITE A RECORD
1070 005312 005215          JSR   PC,WRITE ;SET 'GO' BIT
1071 005314 004767 177712          INC   (R5) ;WAIT FOR READY
1072 005320 000404          JSR   PC,WAITRDY ;TAKE ERROR EXIT
1073 005322 032765 040000 000012 3$:   BR   99$ ;CHECK ERROR EXIT
          BIT   #ERR,DS(R5)
    
```

N04

DZTUG-A TMO2/TU16J DRIVE FUNCTION TIMER MACY11 27(1006) 09-DEC-76 10:45 PAGE 3-1  
DZTUGA.P11 08-DEC-76 12:38 DRIVE SUBROUTINES

SEQ 0052

1074	005330	001401		BEQ	100\$
1075	005332	000262	99\$:	SEV	
1076	005334	000207	100\$:	RTS	PC
1077					

805

DZTUG-A TMD2/TU16J DRIVE FUNCTION TIMER MACY11 27(1006) 09-DEC-76 10:45 PAGE 4  
DZTUGA.P11 09-DEC-76 12:39 DRIVE SUBROUTINES

SEQ 0053

1079  
1090

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

```
1084 ;      MOVB  SLAVE #,Q#SLVNUM  
1085 ;      JSR   PC,REWIND  
1086 ;SUBROUTINE RETURNS TO CALLER WITH SELECTED SLAVE AT 'BOT', & 'V' SET IF  
1087 ;AN ERROR OCCURS.  
1088  
1089 005336 004767 177632 .REWIND:JSR  PC,RHINIT          ;INITIALIZE CONTROLLER
```

```

1093 005342 004367 000206      JSR      R3,TMCMD      ;GO TO TM COMMAND SUBROUTINE
1094 005346 000000      .WORD    0             ;BUS ADDRESS (NOT USED)
1095 005350 000000      .WORD    0             ;WORD COUNT (NOT USED)
1096 005352 000000      .WORD    0             ;FRAME COUNT (NOT USED)
1097 005354 000006      .WORD    RWD          ;REWIND COMMAND
1098 005356 005215      INC      (R5)         ;SET 'GO' BIT
1099 005360 032765 000002 000012 1$: BIT      #BOT,DS(R5)   ;BRANCH IF 'BOT' SET
1100 005366 001005      BNE     2$           ;
1101 005370 032765 040000 000012  BIT      #ERR,DS(R5)   ;CHECK ERROR BIT
1102 005376 001006      BNE     99$         ;BRANCH IF ERROR BIT SET
1103 005400 000767      BR      1$           ;
1104
1105 005402 032765 020000 000012 2$: BIT      #PIP,DS(R5)   ;WAIT FOR TAPE MOTION TO STOP
1106 005410 001374      BNE     2$           ;
1107 005412 000401      BR      100$        ;
1108 005414 000262      99$: SEV             ;
1109 005416 000207      100$: RTS          PC
1110
1111      ;SUBROUTINE TO WRITE 256. WORD RECORD
1112      ;CALL: JSR      PC,WRITE
1113
1114 005420 004367 000130  WRITE: JSR      R3,TMCMD      ;GO TO TM COMMAND SUBROUTINE
1115 005424 015702      .WORD    WTBUF        ;BUS ADDRESS
1116 005426 177600      .WORD    WRDCNT       ;WORD COUNT
1117 005430 177400      .WORD    FRMCNT       ;FRAME COUNT
1118 005432 000060      .WORD    WFW         ;WRITE FORWARD COMMAND
1119 005434 000207      RTS      PC
1120
1121      ;SUBROUTINE TO READ A 256. WORD RECORD.
1122      ;CALL: JSR      PC,READ
1123
1124 005436 004337 005554  READ: JSR      R3,#TMCMD
1125 005442 015702      .WORD    RDBUF        ;ADDRESS OF READ BUFFER
1126 005444 177600      .WORD    WRDCNT       ;2'S COMPLEMENT OF WORD COUNT
1127 005446 177400      .WORD    FRMCNT       ;2'S COMPLEMENT OF FRAME COUNT
1128 005450 000070      .WORD    RDFWD        ;READ FORWARD COMMAND
1129 005452 000207      RTS      PC
1130
1131      ;SUBROUTINE TO INITIATE READ REVERSE COMMAND
1132      ;CALL: JSR      PC,REVRD
1133
1134 005454 004367 000074  REVRD: JSR      R3,TMCMD
1135 005460 016302      .WORD    RDBUF+256.   ;ADDRESS OF READ REVERSE BUFFER
1136 005462 177600      .WORD    WRDCNT       ;2'S COMPLEMENT OF WORD COUNT
1137 005464 177400      .WORD    FRMCNT       ;2'S COMPLEMENT OF FRAME COUNT
1138 005466 000076      .WORD    RDREV        ;READ REVERSE COMMAND
1139 005470 000207      RTS      PC
1140
1141      ;SUBROUTINE TO SPACE FORWARD 1 RECORD
1142 005472 012765 177777 000006  FWDSPC: MOV     #-1,FC(R5) ;LOAD RECORD COUNT
1143 005500 012715 000031      MOV     #SPCFWD+1,(R5) ;LOAD COMMAND
1144 005504 004767 177522      JSR      PC,WAITRDY   ;WAIT FOR READY
1145 005510 000207      RTS      PC          ;RETURN
1146
1147      ;SUBROUTINE TO WRITE A RECORD AND BACK SPACE OVER THE RECORD.
1148 005512 004767 177702  WRT.BK: JSR      PC,WRITE ;WRITE THE RECORD
    
```



```

1149 005516 005215          INC      (R5)          ;SET 'GO' BIT
1150 005520 004767 177506   JSR      PC, WAITRDY
1151 005524 102412          BVS      2$
1152 005526 012765 177777 000006   MOV      #-1, FC(R5)   ;LOAD RECORD COUNT
1153 005534 012715 000033   MOV      #SPCREV+1, (R5) ;LOAD COMMAND
1154 005540 004767 177466   JSR      PC, WAITRDY
1155 005544 102402          BVS      2$
1156 005546 004767 177166   1$:     JSR      PC, DELAY ;WAIT FOR TAPE MOTION TO STOP
1157 005552 000207          2$:     RTS      PC
1158
1159          ;SUBROUTINE TO LOAD A COMMAND
1160          ;CALL: JSR      R3, TMCMD
1161          ;      .WORD   BUS ADDRESS
1162          ;      .WORD   WORD COUNT (2'S COMPLEMENT)
1163          ;      .WORD   FRAME COUNT (2'S COMPLEMENT)
1164          ;      .WORD   COMMAND
1165
1166 005554 012365 000004   TMCMD:  MOV      (R3)+, BA(R5) ;LOAD BUS ADDRESS
1167 005560 012365 000002   MOV      (R3)+, WC(R5)   ;LOAD WORD COUNT
1168 005564 012365 000006   MOV      (R3)+, FC(R5)   ;LOAD FRAME COUNT
1169 005570 012315          MOV      (R3)+, (R5)     ;LOAD COMMAND
1170 005572 000203          RTS      R3              ;RETURN
1171
1172          ;SUBROUTINE TO PRINT TU16 SERIAL NUMBER
1173          ;JSR      PC, SNPT
1174
1175 005574 016503 000030   SNPT:   MOV      SN(R5), R3
1176 005600 012701 001144   MOV      #ODIGITS, R1
1177 005604 000303          SWAB     R3
1178 005606 006003          ROR     R3
1179 005610 006003          ROR     R3
1180 005612 006003          ROR     R3
1181 005614 006003          ROR     R3          ;GET FIRST DIGIT
1182 005616 042703 177760   BIC     #177760, R3
1183 005622 052703 000260   BIS     #260, R3
1184 005626 110321          MOVB    R3, (R1)+     ;FILL FIRST DIGIT
1185 005630 016503 000030   MOV     SN(R5), R3
1186 005634 000303          SWAB     R3
1187 005636 042703 177760   BIC     #177760, R3
1188 005642 052703 000260   BIS     #260, R3
1189 005646 110321          MOVB    R3, (R1)+     ;GET SECOND DIGIT
1190 005650 016503 000030   MOV     SN(R5), R3
1191 005654 006003          ROR     R3
1192 005656 006003          ROR     R3
1193 005660 006003          ROR     R3
1194 005662 006003          ROR     R3
1195 005664 042703 177760   BIC     #177760, R3
1196 005670 052703 000260   BIS     #260, R3
1197 005674 110321          MOVB    R3, (R1)+     ;GET THIRD DIGIT
1198 005676 016503 000030   MOV     SN(R5), R3
1199 005702 042703 177760   BIC     #177760, R3
1200 005706 052703 000260   BIS     #260, R3
1201 005712 110321          MOVB    R3, (R1)+     ;GET FOURTH DIGIT
1202 005714 105011          CLRB    (R1)
1203 005716 000004 001144   TYPE, ODIGITS ;TYPE SERIAL NUMBER
1204 005722 000207          RTS     PC              ;RETURN
    
```

```

1205
1206          .SBTTL  PROGRAM INITIALIZATION
1207 005724 012706 000600      INIT:  MOV      #STKPTR,SP      ;SET STACK PTR
1208
1209 005730 013746 000006      SUSWR: MOV      @#6,-(SP)      ;SAVE VECTORS
1210 005734 013746 000004      MOV      @#4,-(SP)
1211 005740 012737 005760 000004      MOV      #61$,@#4      ;SET UP FOR TIMEOUT
1212 005746 022777 177777 173024      CMP      #-1,@SWR      ;REFERENCE HARDWARE SWITCH REGISTER
1213 005754 001402      BEQ      60$
1214 005756 000404      BR      62$
1215 005760 022626      61$:  CMP      (SP)+,(SP)+      ;ADJUST STACK
1216 005762 012767 000176 173010      60$:  MOV      #SWREG,SWR      ;POINT TO SOFTWARE SWITCH REG
1217 005770 012637 000004      62$:  MOV      (SP)+,@#4      ;RESTORE VECTORS
1218 005774 012637 000006      MOV      (SP)+,@#6
1219 006000 022737 000176 001000      CMP      #SWREG,@#SWR
1220 006006 001002      BNE      64$
1221 006010 004767 174014      JSR      PC,CNTLU
1222 006014 105037 001124      64$:  CLRB     @#PRGFLG      ;CLEAR PROGRAM FLAG
1223 006020 105037 001121      CLRB     @#ITCNT      ;CLEAR ITERATION COUNT
1224 006024 105037 001122      CLRB     @#TSTNUM      ;SET TEST # 0
1225 006030 105037 001123      CLRB     @#ERFLG      ;CLEAR ERROR FLAG
1226 006034 105067 173070      CLRB     ASFLG      ;CLEAR ASK FLAG
1227 006040 012737 000006 000004      MOV      #ERRVEC+2,@#ERRVEC
1228 006046 012737 000002 000006      MOV      #RTI,@#ERRVEC+2      ;CHECK IF 'LP' IS AVAILABLE
1229 006054 005037 001264      2$:  CLR      @#INBUF
1230 006060 000004 001374      TYPE,CRLF
1231 006064 000004 013330      TYPE,M.NAM      ;TYPE TITLE
1232 006070 000004 013377      TYPE,I.REG      ;ASK USER TO TYPE CONT BASE ADRS
1233 006074 004767 175374      JSR      PC,INPUT      ;GET USER INPUT
1234 006100 004767 175154      4$:  JSR      PC,CNVTA0      ;CONVERT ASCII TO OCTAL
1235 006104 013737 001116 001010      MOV      @#OCTALO,@#TMBASE      ;SET NEW ADDRESS
1236 006112 013705 001010      5$:  MOV      @#TMBASE,R5
1237
1238          ;ROUTINE TO CHECK IF CONTROLLER (RH11) IS AVAILAABLE
1239 006116 000261      SEC      ;SET 'C' IN PSW
1240 006120 005715      TST      (R5)      ;BRANCH IF CONTROLLER AVAIL
1241 006122 103003      BCC      6$
1242 006124 000004 013716      TYPE,E.NCON
1243 006130 000675      BR      INIT
1244 006132 012737 003650 000004      6$:  MOV      #ERRTRP,@#ERRVEC      ;SET ERROR TRAP VECTOR
    
```

```

1246          ;ROUTINE TO GET TMO2 DRIVES USER DESIRES TO TEST
1247 006140 105037 001123 DRIVES: CLR  @#ERFLG          ;CLEAR ERROR FLAG
1248 006144 012701 001154      MOV  #DRVTBL,R1          ;MARK ALL DRIVES AS NOT TO
1249 006150 012700 000004      MOV  #4,R0             ;BE TESTED. A '0' INDICATES
1250 006154 005021          1$: CLR  (R1)+          ;THAT A DRIVE IS NOT TO BE
1251 006156 005300               DEC  R0             ;TESTED
1252 006160 001375               BNE  1$
1253 006162 000004 013444      TYPE,I,DRVS
1254 006166 004767 175302      JSR  PC,INPUT          ;GET USER INPUT
1255 006172 012700 001264      MOV  #INBUF,R0
1256 006176 122710 000101      CMPB #'A,(R0)         ;AN 'A' SPECIFIES ALL
1257 006202 001013               BNE  3$             ;DRIVES TO BE TESTED
1258 006204 110667 172714      MOVB SP,PRGFLG        ;SET FLAG TO IND ALL DRIVES
1259 006210 012701 001154      MOV  #DRVTBL,R1      ;MARK ALL DRIVES TO BE TESTED
1260 006214 012700 000004      MOV  #4,R0           ;A '-1' INDICATES THAT A DRIVE
1261 006220 012721 177777      2$: MOV  #-1,(R1)+    ;IS TO BE TESTED
1262 006224 005300               DEC  R0
1263 006226 001374               BNE  2$
1264 006230 000417               BR   CHKDRV          ;GO CHECK DRIVE AVAILABILITY
1265
1266          ;GET USER SELECTED DRIVES AND MARK EACH DRIVE SELECTED TO BE TESTED
1267 006232 122710 000015      3$: CMPB #CR,(R0)
1268 006236 001414               BEQ  CHKDRV
1269 006240 121027 000054      CMPB (R0),#',         ;CHECK IF 'COMMA'
1270 006244 001001               BNE  4$
1271 006246 105720               TSTB (R0)+          ;STEP PTR PAST 'COMMA'
1272 006250 112001          4$: MOVB (R0)+,R1
1273 006252 042701 177770      BIC  #177770,R1
1274 006256 112761 177777 001154      MOVB #-1,DRVTBL(R1)
1275 006264 000240               NOP
1276 006266 000761               BR   3$
1277
1278          ;ASCERTAIN THAT DRIVES (TMO2'S) SPECIFIED ARE AVAILABLE
1279 006270 005000      CHKDRV: CLR  R0          ;A 0/-1 INDICATES THAT THE
1280 006272 105760 001154      1$: TSTB DRVTBL(R0)    ;DRIVE IS NOT/IS TO BE TESTED
1281 006276 001005               BNE  3$
1282 006300 005200          2$: INC  R0
1283 006302 122700 000010      CMPB #8.,R0
1284 006306 001371               BNE  1$
1285 006310 000421               BR   4$
1286 006312 110037 001004      3$: MOVB R0,@#DRVNUM
1287 006316 004737 005116      JSR  PC,@#DRVAVA      ;CHECK IF AVAILABLE
1288 006322 102366          BVC  2$               ;'V' BIT SET INDICATES NOT AVAIL
1289 006324 000004 013763      TYPE,E,NDRV
1290 006330 116037 001132 014015      MOVB DIGTAB(R0),@#E.NAVA ;SET DRIVE # IN MESSAGE
1291 006336 000004 014015      TYPE,E,NAVA
1292 006342 110637 001123      MOVB SP,@#ERFLG      ;SET 'ERROR' FLAG
1293 006346 105060 001154      CLRB DRVTBL(R0)     ;MARK DRIVE UNAVAILABLE
1294 006352 000752          BR   2$             ;CHECK NEXT DRIVE
1295 006354 105737 001123      4$: TSTB @#ERFLG      ;GO GET SLAVES IF NO ERROR
1296 006360 001403          BEQ  SLAVES
1297 006362 105737 001124      TSTB @#PRGFLG        ;ASK USER TO RETYPE DRIVES IF
1298 006366 001664          BEQ  DRIVES          ;'ALL' NOT SPECIFIED
1299
1300          ;ROUTINE TO GET SLAVES (TU16'S) USER DESIRES TO TEST
1301 006370 105037 001123      SLAVES: CLRB @#ERFLG ;CLEAR 'ERROR' FLAG

```

```

1302 006374 012701 001164      MOV      #SLVTBL,R1      ;MARK ALL SLAVES (64.) AS NOT
1303 006400 012700 000040      MOV      #32.,R0        ;TO BE TESTED.A 0 INDICATES THAT
1304 006404 005021          1$: CLR      (R1)+         ;A DRIVE'S SLAVE IS NOT TO BE
1305 006406 005300          DEC      R0             ;TESTED
1306 006410 001375          BNE     1$
1307 006412 005000          CLR      R0             ;R0 = DRIVE # FOR SLAVES
1308 006414 012701 001164      MOV      #SLVTBL,R1     ;R1 POINTS TO DRIVE'S SLAVE
1309 006420 105760 001154      2$: TSTB   DRVTBL(R0)   ;IF DRIVE IS TO BE TESTED
1310 006424 001007          BNE     4$             ;GO TO 4$ OTHERWISE
1311 006426 062701 000010      3$: ADD     #8.,R1       ;STEP SLAVE PTR TO NEXT DRIVE'S
1312 006432 005200          INC     R0             ;SLAVES AND INCREMENT DRIVE #
1313 006434 122700 000010      CMPB   #8.,R0          ;CHECK ALL DRIVES
1314 006440 001367          BNE     2$            ;AND WHEN ALL DRIVES CHECKED
1315 006442 000454          BR      CHKSLV        ;GO CHECK SLAVE AVAILABILITY
1316
1317 006444 105737 001124      4$: TSTB   @#PRGFLG     ;BRANCH IF USER SELECTED ALL
1318 006450 001020          BNE     5$            ;DRIVES
1319 006452 110067 172326      MOVB   R0,DRVNUM       ;GET DRIVE #
1320 006456 116037 001132 013525  MOVB   DIGTAB(R0),@#I.DRV ;PREPARE USER ACTION MESSAGE
1321 006464 000004 013506      TYPE,I.SLVS
1322 006470 004767 175000      JSR    PC,INPUT        ;GET USER INPUT
1323 006474 012703 001264      MOV     #INBUF,R3      ;SET PTR TO USER INPUT
1324 006500 122710 000101      CMPB   #'A,(R0)       ;BRANCH IF USER DOES NOT WANT
1325 006504 001015          BNE     7$            ;'ALL' SLAVES
1326 006506 110637 001124      MOVB   SP,@#PRGFLG     ;SET 'ALL' INDICATOR
1327 006512 012701 001164      5$: MOV     #SLVTBL,R1   ;MARK ALL SLAVES FOR ALL
1328 006516 012700 000040      MOV     #32.,R0        ;DRIVES AS TO BE TESTED
1329 006522 012721 177777      6$: MOV     #-1,(R1)+   ;
1330 006526 005300          DEC     R0
1331 006530 001374          BNE     6$
1332 006532 105737 001124      TSTB   @#PRGFLG       ;BRANCH IF ALL WAS SELECTED
1333 006536 001016          BNE     CHKSLV
1334
1335 006540 122713 000015      7$: CMPB   #CR,(R3)     ;GET USER SELECTED SLAVES FOR
1336 006544 001730          BEQ    3$             ;DRIVE
1337 006546 121327 000054      CMPB   (R3),#'.'      ;STEP PTR PAST 'COMMA'
1338 006552 001001          BNE     8$
1339 006554 105723          TSTB   (R3)+
1340 006556 112304          8$: MOVB   (R3)+,R4     ;AND MARK SELECED SLAVE
1341 006560 042704 177770      BIC    #177770,R4     ;AS TO BE TESTED
1342 006564 060104          ADD    R1,R4
1343 006566 112714 177777      MOVB   #-1,(R4)
1344 006572 000762          BR     7$
1345
1346          ;ASCERTAIN THAT SLAVES (TU16'S) SELECTED ARE AVAILABLE
1347 006574 005000      CHKSLV: CLR    R0      ;R0 WILL CONTAIN THE DRIVE #
1348 006576 005001      CLR    R1      ;AND R1 THE SLAVE #
1349 006600 012702 001164      MOV     #SLVTBL,R2   ;SET PTR TO SLAVE TABLE
1350 006604 105760 001154      1$: TSTB   DRVTBL(R0) ;BRANCH IF DRIVE SELECTED
1351 006610 001007          BNE     3$           ;& AVAILABLE FOR TEST
1352 006612 005200          2$: INC     R0       ;INCREMENT DRIVE #
1353 006614 062702 000010      ADD     #8.,R2       ;STEP SLAVE PTR TO NEXT DRIVE'S
1354 006620 022700 000010      CMP     #8.,R0       ;SLAVES. BRANCH TO 1$ IF NOT ALL
1355 006624 001367          BNE     1$           ;DRIVES CHECKED OTHERWISE EXIT
1356 006626 000434          BR     7$
1357

```

```

1358 006630 005001          3$: CLR R1 ;SET SLAVE # C
1359 006632 105712          4$: TSTB (R2) ;BRANCH IF DRIVE'S SLAVE IS SEL-
1360 006634 001006          BNE 6$ ;ECTED FOR TEST
1361 006636 005201          5$: INC R1 ;INCREMENT SLAVE #
1362 006640 005202          INC R2 ;STEP PTR TO NEXT SLAVE
1363 006642 022701 000010  CMP #8.,R1 ;GO TO 4$ IF ALL SLAVES NOT
1364 006646 001371          BNE 4$ ;CHECKED
1365 006650 000760          BR 2$ ;OTHERWISE GO TO 2$ ABOVE
1366
1367 006652 110037 001004          6$: MOVB RD, @#DRVNUM ;PASS DRIVE & SLAVE #
1368 006656 110137 001005          MOVB R1, @#SLVNUM
1369 006662 004737 005144          JSR PC, @#SLVAVA ;AND CHECK IF AVAILABLE
1370 006666 102363          BVC 5$ ;'V' BIT SET ON RETURN IND-
1371 006670 116037 001132 014005  MOVB DIGTAB(RD), @#E.DRV ;ICATES ERROR. PREPARE ERROR
1372 006676 116137 001132 014015  MOVB DIGTAB(R1), @#E.NAVA ;MESSAGE
1373 006704 000004 013777          TYPE, E.NSLV
1374 006710 110637 001123          MOVB SP, @#ERFLG ;SET ERROR INDICATOR
1375 006714 105012          CLRB (R2) ;CLEAR SLAVE TABLE ENTRY
1376 006716 000747          BR 5$ ;GET NEXT SLAVE
1377
1378 006720 105737 001123          7$: TSTB @#ERFLG ;BRANCH IF NO ERROR
1379 006724 001403          BEQ 100$
1380 006726 105737 001124          TSTB @#PRGFLG ;BRANCH IF NOT 'ALL'
1381 006732 001616          BEQ SLAVES ;ASK USER TO RETYPE SLAVES
1382 006734 012737 003650 000004 100$: MOV #ERRTRP, @#ERRVEC
1383
1384 ;SCAN DIVE AND SLAVE TABLE FOR DRIVE/SLAVE COMBINATION TO TEST
1385 006742 105037 001004          CLRB @#DRVNUM ;SET DRIVE AND SLAVE # C
1386 006746 105037 001005          CLRB @#SLVNUM
1387 006752 012737 001164 001006  MOV #SLVTBL, @#SLVPTR ;SET PTR TO SLAVE TABLE
1388 006760 105037 001125          CLRB @#UNTFND ;CLEAR 'UNIT FOUND' INC.
1389
1390 006764 113700 001004          BEGIN: MOVB @#DRVNUM, RD ;GET DRIVE #
1391 006770 113701 001005          MOVB @#SLVNUM, R1 ;AND SLAVE #
    
```

1394	006774	013702	001006		MOV	Q#SLVPTR,R2		:GET SLAVE PTR
1395	007000	105760	001154	1\$:	TSTB	DRVTBL(R0)		:BRANCH IF DRIVE AVAIL TO TEST
1396	007004	001011			BNE	3\$		
1397	007006	005001			CLR	R1		:CLEAR SLAVE #
1398	007010	062702	000010		ADD	#8.,R2		:AND STEP PTR TO NEXT DRIVE'S
1399	007014	005200		2\$:	INC	R0		:SLAVES AND INCREMENT DRIVE #
1400	007016	022700	000010		CMP	#8.,R0		:EXIT TEST IF ALL DRIVES
1401	007022	001366			BNE	1\$		:CHECKED OTHERWISE CONTINUE
1402	007024	000137	012732		JMP	Q#END		:SCAN FOR NEXT 'UNIT'
1403								
1404	007030	105712		3\$:	TSTB	(R2)		:BRANCH IF SLAVE ON DRIVE IS
1405	007032	001007			BNE	4\$		:AVAILABLE THERWISE STEP
1406	007034	005202			INC	R2		:PTR TO NEXT SLAVE
1407	007036	005201			INC	R1		:INCREMENT SLAVE #
1408	007040	122701	000010		CMPB	#8.,R1		:UNTIL ALL SLAVES CHECKED
1409	007044	001371			BNE	3\$		:WHEN ALL SLAVES CHECKED
1410	007046	005001			CLR	R1		:SET SLAVE # 7
1411	007050	000761			BR	2\$		:AND CONTINUE SCAN
1412								
1413	007052	110637	001125	4\$:	MOVB	SP,Q#UNTFND		:INDICATE THAT A 'UNIT' IS FOUND
1414	007056	110037	001004		MOVB	R0,Q#DRVNUM		:SET DRIVE 3

```

1419 007062 110137 001005      MOVB   R1, @#SLVNUM      ;SET SLAVE #
1420 007066 010237 001006      MOV    R2, @#SLVPTR     ;SAVE SLAVE PTR
1421
1422 007072 105737 001130      5$:   TSTB   @#ASFLG
1423 007076 001044              BNE    7$
1424 007100 112767 000001 172022  MOVB   #1, ASFLG
1425
1426 007106 105037 001124      CLRB   @#PRGFLG         ;CLEAR PROGRAM INDICATOR
1427 007112 000004 013565      TYPE, I.SKEW           ;ASK USER IF HE WANTS TO RUN SKEW TESTS
1428 007116 004767 174352      JSR    PC, INPUT       ;GET USER INPUT
1429 007122 012703 001264      MOV    #INBUF, R3      ;GET REPLY
1430 007126 122713 000060      CMPB   #'0', (R3)      ;BRANCH IF 'NO' (0)
1431 007132 001406              BEQ    6$
1432 007134 122713 000061      CMPB   #'1', (R3)      ;CHECK IF 'YES' (1)
1433 007140 001354              BNE    5$              ;NEITHER SO ASK AGAIN
1434 007142 111337 001124      MOVB   (R3), @#PRGFLG  ;SET INDICATOR
1435 007146 000420              BR     7$
1436
1437 007150 105037 001127      6$:   CLRB   @#NRZFLG     ;CLEAR NRZ INDICATOR
1438 007154 000004 013626      TYPE, I.NRZ           ;ASK USER IF DRIVE 'NRZ' ONLY
1439 007160 004767 174310      JSR    PC, INPUT       ;GET USER INPUT
1440 007164 012703 001264      MOV    #INBUF, R3      ;GET REPLY
1441 007170 122713 000060      CMPB   #'0', (R3)      ;BRANCH IF 'NO' (0)
1442 007174 001405              BEQ    7$
1443 007176 122713 000061      CMPB   #'1', (R3)      ;CHECK IF 'YES' (1)
1444 007202 001362              BNE    6$              ;ASK AGAIN IF NEITHER
1445 007204 111337 001127      MOVB   (R3), @#NRZFLG  ;SET INDICATOR
1446 007210
1447
1448 007210 052737 000100 177560  TYPHDR: BIS    #100, @#TKS      ;SET KEYBOARD IE BIT
1449 007216 000004 014346      TYPE, L.HDR1
1450 007222 116037 001132 014526  MOVB   DIGTAB(R0), @#L.DRV  ;SET DRIVE #
1451 007230 116137 001132 014540  MOVB   DIGTAB(R1), @#L.SLV  ;AND SLAVE #
1452 007236 112737 000071 014543  MOVB   #'9, @#L.CHAN       ;GET SLAVES CHANNEL TYPE
1453 007244 032765 010000 000026  BIT    #CH7, DT(R5)
1454 007252 001403              BEQ    1$
1455 007254 112737 000067 014543  MOVB   #'7, @#L.CHAN       ;SET 7 CHANNEL
1456 007262 000004 014461      1$:   TYPE, L.HDR2
1457 007266 004767 176302      JSR    PC, SNPT         ;GO PRINT SERIAL NUMBER
1458 007272 000004 014562      TYPE, L.HDR3
1459 007276 012737 007336 001002  MOV    #TST001, @#SCPADR   ;SET 'SCOPE' ADDRESS FOR FIRST TEST
1460 007304 010500              MOV    R5, R0
1461 007306 062700 000006      ADD    #FC, R0          ;R0 CONTAINS ADDRESS OF FC REG
1462 007312 010501              MOV    R5, R1
1463 007314 062701 000012      ADD    #DS, R1          ;R1 CONTAINS ADDRESS OF DS REG
1464 007320 012703 004444      MOV    #TIMER, R3       ;SET JUMP ADDRESS TO TIMER
1465 007324 105737 001124      TSTB   @#PRGFLG        ;BRANCH IF NOT SKEW TESTS
1466 007330 001402              BEQ    TST001
1467 007332 000137 012764      JMP    @#SKEWTST

```

```

1469 .SBTTL START OF TESTS
1470 ;TEST 001 - WRITE FROM BOT
1471 ;THIS TEST WILL MEASURE ACCELERATION DELAY REQUIRED TO
1472 ;MOVE THE TAPE APPROXIMATELY SEVEN (7) INCHES FORWARD
1473 ;FROM DEAD STOP BEFORE STARTING TO TRANSFER DATA.
1474
1475 ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
1476 007336 112737 000001 001122 †ST001: MOVB #1,‡TSTNUM ;SET TEST #
1477 007344 012702 007370 MOV #1,R2 ;SET RETURN PC FROM TIMER
1478 007350 004767 175762 JSR PC,REWIND ;REWIND SLAVE
1479 (1) 007354 102420 BVS 99$ ;BRANCH IF ERROR ON REWIND
1479 007356 004767 176036 JSR PC,WRITE ;GO SETUP WRITE COMMAND
1480 007362 004767 174766 JSR PC,TIMON ;TURN TIMER ON
1481 007366 005215 INC (R5) ;SET 'GO' BIT
1482
1483 007370 005765 000032 1$: TST TC(R5) ;BRANCH WHEN 'ACCL'=0
1484 007374 100002 BPL 2$
1485 007376 000163 004444 JMP TIMER(R3) ;GO TO TIMER & RETJRN VIA R2
1486
1487 007402 004767 175624 2$: JSR PC,WAITRDY ;WAIT FOR COMMAND TO FINISH
1488 007406 102403 BVS 99$ ;BRANCH IF ERROR
1489 007410 004767 175066 JSR PC,TIMOK ;GO CHECK TIME
1490 007414 000401 BR 100$
1491 007416 104400 99$: HLT
1492 007420 104000 100$: SCOPE
1493
1494 ;TEST 002 - WRITE START
1495 ;THIS TST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
1496 007422 112737 000002 001122 †ST002: MOVB #2,‡TSTNUM ;SET TEST # 2
1497 007430 004767 175764 JSR PC,WRITE ;INITIATE WRITE COMMAND
1498 007434 012702 007446 MOV #1,R2 ;SET RETURN PC FROM TIMER
1499 007440 004767 174710 JSR PC,TIMON
1500 007444 005215 INC (R5) ;SET 'GO' BIT
1501
1502 007446 005765 000032 1$: TST TC(R5) ;BRANCH WHEN 'ACCL'=0
1503 007452 100002 BPL 2$
1504 007454 000163 004444 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
1505
1506 007460 004767 175546 2$: JSR PC,WAITRDY ;WAIT FOR READY
1507 007464 102403 BVS 99$ ;BRANCH IF ERROR
1508 007466 004767 175010 JSR PC,TIMOK ;GO CHECK TIME RECORDED
1509 007472 000401 BR 100$ ;EXIT VIA SCOPE
1510
1511 007474 104400 99$: HLT ;REPORT ERROR
1512 007476 104000 100$: SCOPE
1513
1514 ;TEST 003- WRITE SHUTDOWN
1515 ;THIS TEST MEASURES TIME FROM 'FC REG'=0 TO 'SWDN'=1.
1516 007500 112737 000003 001122 †ST003: MOVB #3,‡TSTNUM ;SET TEST#3
1517 007506 004767 175706 JSR PC,WRITE ;INITIATE WRITE COMMAND
1518 007512 005215 INC (R5) ;SET 'GO' BIT
1519
1520 007514 005710 1$: TST (R0) ;BRANCH WHEN WRITING FINISHED
1521 007516 001404 BEQ 2$
1522 007520 032711 040000 BIT #ERR,(R1) ;MONITOR ERROR BIT
1523 007524 001017 BNE 99$
    
```





```

1577 007722 004767 174426      JSR    PC,TIMON      ;TURN TIMER ON
1578 007726 005215      INC     (R5)         ;SET 'GO' BIT
1579
1580 007730 005765 000032      1$:    TST    TC(R5)      ;BRANCH WHEN 'ACCL' RESETS
1581 007734 100002      BPL    2$           ;
1582 007736 000163 004444      JMP    TIMER(R3)    ;GO TO TIMER & RETURN VIA R2
1583
1584 007742 004767 175264      2$:    JSR    PC,WAITRDY  ;WAIT FOR READY
1585 007746 102403      BVS    99$         ;BRANCH IF ERROR
1586 007750 004767 174526      JSR    PC,TIMOK     ;CHECK RECORDED TIME
1587 007754 000401      BR     100$        ;
1588
1589 007756 104400      99$:   HLT
1590 007760 104000      100$:  SCOPE
1591
1592      ;TEST 006 - READ START
1593      ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
1594 007762 112737 000006 001122  TST006: MOVB    #6,2#TSTNUM  ;SET TEST #6
1595 007770 004767 175516      JSR    PC,WRT.BK    ;WRITE A RECORD & BACK SPACE
1596 007774 102422      BVS    99$
1597 007776 004767 175434      JSR    PC,READ
1598 010002 012702 010014      MOV    #1,R2        ;SET RETURN PC FROM TIMER
1599 010006 004767 174342      JSR    PC,TIMON     ;TURN TIMER ON
1600 010012 005215      INC     (R5)         ;SET 'GO' BIT
1601
1602 010014 005765 000032      1$:    TST    TC(R5)      ;BRANCH WHEN 'ACCL' RESETS
1603 010020 100002      BPL    2$           ;
1604 010022 000163 004444      JMP    TIMER(R3)    ;GO TO TIMER & RETURN VIA R2
1605
1606 010026 004767 175200      2$:    JSR    PC,WAITRDY  ;
1607 010032 102403      BVS    99$
1608 010034 004767 174442      JSR    PC,TIMOK     ;
1609 010040 000401      BR     100$
1610
1611 010042 104400      99$:   HLT
1612 010044 104000      100$:  SCOPE
1613
1614      ;TEST 007 - READ SHUTDOWN
1615      ;THIS TEST MEASURES TIME FROM 'FC REG'=FRAME COUNT TO 'SWDN'=1.
1616 010046 112737 000007 001122  TST007: MOVB    #7,2#TSTNUM  ;SET TEST #7
1617 010054 004767 175432      JSR    PC,WRT.BK    ;WRITE A RECORD & BACK SPACE
1618 010060 102430      BVS    99$         ;BRANCH IF ERROR
1619 010062 004767 175350      JSR    PC,READ
1620 010066 005215      INC     (R5)         ;SET 'GO' BIT
1621
1622 010070 022710 000400      1$:    CMP    #-FRMCNT,(R0)  ;WAIT FOR FRAME COUNT TO
1623 010074 001404      BEQ    2$           ;= # OF FRAMES WRITTEN
1624 010076 032711 040000      BIT    #ERR,(R1)    ;MONITOR ERROR BIT
1625 010102 001017      BNE    99$
1626 010104 000771      BR     1$
1627
1628 010106      2$:
1629 010106 004767 174242      JSR    PC,TIMON     ;TURN TIMER ON
1630 010112 010702      MOV    PC,R2        ;SET RETURN PC FROM TIMER
1631 010114 032711 000020      BIT    #SDWN,(R1)   ;BRANCH WHEN SDWN SETS
1631 010120 001002      BNE    3$
    
```

```

1632 010122 000163 004444          JMP     TIMER(R3)          ;GO TO TIMER & RETURN VIA R2
1633
1634 010126 004767 175100          3$:   JSR     PC, WAITRDY
1635 010132 102403                    BVS     99$
1636 010134 004767 174342          JSR     PC, TIMOK
1637 010140 000401                    BR      100$
1638
1639 010142 104400          99$:   HLT
1640 010144 104000          100$:  SCOPE          ;REPORT ERROR
1641
1642          ;TEST 010 - READ SETTLEDOWN
1643          ;THIS TEST MEASUREC TIME FROM 'SWDN'=1 TO 'SWDN'=0.
1644 010146 112737 000010 001122  TST010: MOVB   #10, 2#TSTNUM          ;SET TEST #10
1645 010154 012702 010232          MOV     #4$, R2          ;SET RETURN PC FROM TIMER
1646 010160 004767 175326          JSR     PC, WRT.BK        ;WRITE A RECORD & BACK SPACE
1647 010164 102436                    BVS     99$
1648 010166 004767 175244          JSR     PC, READ
1649 010172 005215                    INC     (R5)          ;SET 'GO' BIT
1650
1651 010174 105711          1$:   TSTB   (R1)          ;WAIT FOR READY
1652 010176 100404                    BMI     2$          ;BRANCH WHEN SET
1653 010200 032711 040000          BIT     #ERR, (R1)      ;CHECK ERROR BIT
1654 010204 001026                    BNE     99$
1655 010206 000772                    BR      1$
1656
1657 010210 032711 000020          2$:   BIT     #SDWN, (R1)  ;WAIT FOR ASSERTION OF 'SDWN'
1658 010214 001004                    BNE     3$
1659 010216 032711 040000          BIT     #ERR, (R1)      ;MONITOR ERROR BIT
1660 010222 001017                    BNE     99$
1661 010224 000771                    BR      2$
1662
1663 010226          3$:
1664 (1) 010226 004767 174122          JSR     PC, TIMON          ;TURN TIMER ON
1665 010232 032765 000020 000012  4$:   BIT     #SDWN, DS(R5)  ;WAIT FOR NEGATION OF SDWN
1666 010240 001402                    BEQ     5$
1667 010242 000163 004444          JMP     TIMER(R3)        ;GO TO TIMER & RETURN VIA R2
1668 010246 004767 174760          5$:   JSR     PC, WAITRDY
1669 010252 102403                    BVS     99$
1670 010254 004767 174222          JSR     PC, TIMOK
1671 010260 000401                    BR      100$
1672
1673 010262 104400          99$:   HLT
1674 010264 104000          100$:  SCOPE
1675
1676
1677          ;TEST 011-READ REVERSE START
1678          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
1679 010266 112737 000011 001122  TST011: MOVB   #11, 2#TSTNUM          ;SET RETURN PC FROM TIMER
1680 010274 012702 010332          MOV     #1$, R2          ;WRITE A RECORD
1681 010300 004767 175114          JSR     PC, WRITE        ;SET 'GO' BIT
1682 010304 005215                    INC     (R5)
1683 010306 004767 174720          JSR     PC, WAITRDY
1684 010312 102422                    BVS     99$
1685 010314 004767 174420          JSR     PC, DELAY        ;WAIT FOR TAPE MOTION TO STOP
1686 010320 004767 175130          JSR     PC, REVRD
    
```

```

1687 010324 004767 174024      JSR    PC,TIMON      ;TURN TIMER ON
1688 010330 005215              INC    (R5)          ;SET 'GO' BIT
1689
1690 010332 005765 000032      1$:   TST   TC(R5)    ;BRANCH WHEN 'ACCL' = 0
1691 010336 100002              BPL   2$            ;
1692 010340 000163 004444      JMP   TIMER(R3)     ;GO TO TIMER & RETURN VIA R2
1693
1694 010344 004767 174652      2$:   JSR   PC,WAITRDY ;BRANCH IF ERROR
1695 010350 102403              BVS   99$          ;
1696 010352 004767 174124      JSR   PC,TIMOK
1697 010356 000401              BR    100$         ;
1698
1699 010360 104400      99$:   HLT
1700 010362 104000      100$:  SCOPE
1701
1702      ;TEST 012-READ REVERSE SHUTDOWN
1703      ;THIS TEST MEASURES TIME FROM 'FC REG' = FRAME COUNT TO 'SDWN'=1.
1704 010364 112737 000012 001122  TST012: MOVB   #12,2#TSTNUM
1705 010372 012702 010442      MOV   #3$,R2      ;SET RETURN PC FROM TIMER
1706 010376 004767 175016      JSR   PC,WRITE    ;WRITE A RECORD
1707 010402 005215              INC   (R5)        ;SET 'GO' BIT
1708 010404 004767 174622      JSR   PC,WAITRDY
1709 010410 102427              BVS   99$
1710 010412 004767 175036      JSR   PC,REVRD
1711 010416 005215              INC   (R5)        ;SET 'GO' BIT
1712
1713 010420 022710 000400      1$:   CMP   #-FRMCNT,(R0) ;BRANCH WHEN FRAME COUNT
1714 010424 001404              BEQ   2$          ;= # OF RECORD WRITTEN
1715 010426 032711 040000      BIT   #ERR,(R1)   ;MONITOR ERROR BIT IN 'DS' REG
1716 010432 001016              BNE   99$
1717 010434 000771              BR    1$
1718
1719 010436      2$:
1720 010436 004767 173712      JSR   PC,TIMON    ;TURN TIMER ON
1721 010442 032711 000020      3$:   BIT   #SDWN,(R1) ;BRANCH WHEN SDWN SETS
1722 010446 001002              BNE   4$
1723 010450 000163 004444      JMP   TIMER(R3)   ;GO TO TIMER & RETURN VIA R2
1724 010454 004767 174552      4$:   JSR   PC,WAITRDY ;WAIT FOR READY
1725 010460 102403              BVS   99$
1726 010462 004767 174014      JSR   PC,TIMOK
1727 010466 000401              BR    100$
1728
1729 010470 104400      99$:   HLT
1730 010472 104000      100$:  SCOPE
1731
1732      ;TEST 013-READ REVERSE SETTLEDOWN
1733      ;THIS TEST MEASURES TIME FROM 'SDWN'=1 TO 'SDWN'=0.
1734 010474 112737 000013 001122  TST013: MOVB   #13,2#TSTNUM
1735 010502 012702 010566      MOV   #4$,R2      ;SET RETURN PC FROM TIMER
1736 010506 004767 174706      JSR   PC,WRITE    ;WRITE A RECORD
1737 010512 005215              INC   (R5)        ;SET 'GO' BIT
1738 010514 004767 174512      JSR   PC,WAITRDY
1739 010520 102435              BVS   99$
1740 010522 004767 174726      JSR   PC,REVRD
1741 010526 005215              INC   (R5)        ;SET 'GO' BIT

```

```

1742
1743 010530 105711 1$: TSTB (R1) ;BRANCH WHEN
1744 010532 100404 BMI 2$ ;READY SETS
1745 010534 032711 040000 BIT #ERR,(R1)
1746 010540 001025 BNE 99$
1747 010542 000772 BR 1$
1748
1749 010544 032711 000020 2$: BIT #SDWN,(R1)
1750 010550 001004 BNE 3$
1751 010552 032711 040000 BIT #ERR,(R1)
1752 010556 001016 BNE 99$
1753 010560 000771 BR 2$
1754
1755 010562 3$:
(1) 010562 004767 173566 JSR PC,TIMON ;TURN TIMER ON
1756 010566 032711 000020 4$: BIT #SDWN,(R1) ;BRANCH WHEN SWDN = 0
1757 010572 001402 BEQ 5$
1758 010574 000163 004444 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
1759
1760 010600 004767 174426 5$: JSR PC,WAITRDY ;WAIT FOR READY
1761 010604 102403 BVS 99$
1762 010606 004767 173670 JSR PC,TIMOK
1763 010612 000401 BR 100$
1764
1765 010614 104400 99$: HLT
1766 010616 104000 100$: SCOPE
1767
1768 ;REWIND DRIVE
1769 010620 A:
(1) 010620 004767 174512 JSR PC,REWIND ;REWIND SLAVE
(1) 010624 102401 BVS 99$ ;BRANCH IF ERROR ON REWIND
1770 010626 102002 BVC 100$
1771 010630 104400 99$: HLT
1772 010632 000772 BR A
1773 010634 100$:
1774
1775 ;TEST 014-TURN AROUND DELAY (FORWARD-REVERSE)
1776 ;THIS TEST MEASURES TIME FROM 'GO'=1 (READ REVERSE) TO 'ACCL'=0
1777 010634 112737 000014 001122 TST014: MOVB #14,#TSTNUM
1778 010642 012702 010674 MOV #2$ ,R2 ;SET RETURN PC FROM TIMER
1779 010646 004767 174546 JSR PC,WRITE ;WRITE A RECORD
1780 010652 005215 INC (R5) ;SET 'GO' BIT
1781 010654 004767 174352 JSR PC,WAITRDY
1782 010660 102420 BVS 99$
1783
1784 010662 004767 174566 1$: JSR PC,REVRD ;READ THE RECORD (REVERSE)
1785 010666 004767 173462 JSR PC,TIMON ;TURN TIMER ON
1786 010672 005215 INC (R5) ;SET 'GO' BIT
1787
1788 010674 005765 000032 2$: TST TC(R5) ;WAIT FOR 'ACCL' = 0
1789 010700 100002 BPL 3$
1790 010702 000163 004444 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
1791
1792 010706 004767 174320 3$: JSR PC,WAITRDY
1793 010712 102403 BVS 99$
1794 010714 004767 173562 JSR PC,TIMOK
    
```

```

1795 010720 000401          BR      100$
1796
1797 010722 104400          99$:  HLT
1798 010724 104000          100$: SCOPE
1799
1800
1801          ;TEST 015- TURN AROUND DELAY (REVERSE-FORWARD)
1802          ;THIS TEST MEASURES TIME FROM 'GO'=1 (READ) TO 'ACCL'=0.
1802 010726 112737 000015 001122 †ST015: MOVB   #15,2#TSTNUM
1803 010734 012702 011002          MOV     #2$,R2          ;SET RETURN PC FROM TIMER
1804 010740 004767 174454          JSR    PC,WRITE        ;WRITE A RECORD
1805 010744 005215          INC    (R5)           ;SET 'GO' BIT
1806 010746 004767 174260          JSR    PC,WAITRDY      ;WAIT FOR READY
1807 010752 102426          BVS   99$
1808 010754 004767 174474          JSR    PC,REVRD        ;READ A RECORD IN THE
1809 010760 005215          INC    (R5)           ;SET 'GO' BIT
1810
1811 010762 004767 174244          JSR    PC,WAITRDY
1812 010766 102420          BVS   99$
1813
1814 010770 004767 174442          1$:   JSR    PC,READ      ;READ RECORD FORWARD
1815 010774 004767 173354          JSR    PC,TIMON        ;TURN TIMER ON
1816 011000 005215          INC    (R5)           ;SET 'GO' BIT
1817
1818 011002 005765 000032          2$:   TST    TC(R5)      ;WAIT FOR 'ACCL' = 0
1819 011006 100002          BPL   3$
1820 011010 000163 004444          JMP    TIMER(R3)      ;GO TO TIMER & RETURN VIA R2
1821
1822 011014 004767 174212          3$:   JSR    PC,WAITRDY
1823 011020 102403          BVS   99$
1824 011022 004767 173454          JSR    PC,TIMOK
1825 011026 000401          BR      100$
1826
1827 011030 104400          99$:  HLT
1828 011032 104000          100$: SCOPE
1829
1830          ;TEST 016-GAP SIZE (STOP HALF)
1831 011034 112737 000016 001122 †ST016: MOVB   #16,2#TSTNUM
1832 011042 012702 011100          MOV     #1$,R2          ;SET RETURN PC FROM TIMER
1833 011046 004767 174346          JSR    PC,WRITE        ;WRITE A RECORD
1834 011052 005215          INC    (R5)           ;SET 'GO' BIT
1835 011054 004767 174152          JSR    PC,WAITRDY
1836 011060 102421          BVS   99$
1837 011062 004767 173652          JSR    PC,DELAY        ;DELAY 350 MS
1838 011066 004767 174362          JSR    PC,REVRD        ;READ REVERSE RECORD
1839 011072 004767 173256          JSR    PC,TIMON        ;TURN TIMER ON
1840 011076 005215          INC    (R5)           ;SET 'GO' BIT
1841
1842 011100 005710          1$:   TST    (R0)          ;WAIT FOR FRAME COUNT > C
1843 011102 001002          BNE   2$
1844 011104 000163 004444          JMP    TIMER(R3)      ;GO TO TIMER & RETURN VIA R2
1845
1846 011110 004767 174116          2$:   JSR    PC,WAITRDY      ;WAIT FOR READY BIT TO SET
1847 011114 102403          BVS   99$
1848 011116 004767 173360          JSR    PC,TIMOK        ;CHECK TIME
1849 011122 000401          BR      100$
:ESC

```

1851	011124	104400		99\$: HLT	
1852	011126	104000		100\$: SCOPE	
1853					
1854				:TEST 017-GAP SIZE (START HALF)	
1855	011130	112737	000017	001122 †TST017: MOVB	#17, @#TSTNUM
1856	011136	012702	011210		MOV #1\$, R2 ;SET RETURN PC FROM TIMER
1857	011142	004767	174252		JSR PC, WRITE ;WRITE A RECORD
1858	011146	005215			INC (R5) ;SET 'GO' BIT
1859	011150	004767	174056		JSR PC, WAITRDY ;WAIT FOR READY
1860	011154	102427			BVS 99\$
1861	011156	004767	174272		JSR PC, REVRD ;READ REVERSE THE RECORD
1862	011162	005215			INC (R5) ;SET 'GO' BIT
1863	011164	004767	174042		JSR PC, WAITRDY ;WAIT FOR READY
1864	011170	102421			BVS 99\$ ;BRANCH ON ERROR
1865	011172	004767	173542		JSR PC, DELAY ;WAIT FOR TAPE MOTION TO STOP
1866	011176	004767	174234		JSR PC, READ ;READ RECORD
1867	011202	004767	173146		JSR PC, TIMON ;TURN TIMER ON
1868	011206	005215			INC (R5) ;SET 'GO' BIT
1869					
1870	011210	005710		1\$: TST	(R0) ;WAIT FOR FRAME COUNT > 0
1871	011212	001002			BNE 2\$
1872	011214	000163	004444		JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
1873					
1874	011220	004767	174006	2\$: JSR	PC, WAITRDY ;WAIT FOR READY
1875	011224	102403			BVS 99\$
1876	011226	004767	173250		JSR PC, TIMOK ;CHECK TIME
1877	011232	000401			BR 100\$
1878					
1879	011234	104400		99\$: HLT	
1880	011236	104000		100\$: SCOPE	
1881					
1882				:TEST 020- GAP SIZE (INTERRECORD)	
1883				:THIS TEST MEASURES TIME FROM 'GO'=1 TO 'FC REG' >0.	
1884	011240	112737	000020	001122 †TST020: MOVB	#20, @#TSTNUM
1885	011246	012702	011330		MOV #1\$, R2 ;SET RETURN PC FROM TIMER
1886	011252	004767	174142		JSR PC, WRITE ;WRITE A RECORD
1887	011256	005215			INC (R5) ;SET 'GO' BIT
1888	011260	004767	173746		JSR PC, WAITRDY ;WAIT FOR READY
1889	011264	102433			BVS 99\$
1890	011266	004767	174126		JSR PC, WRITE ;WRITE SECOND RECORD
1891	011272	005215			INC (R5) ;SET 'GO' BIT
1892	011274	004767	173732		JSR PC, WAITRDY ;WAIT FOR READY
1893	011300	102425			BVS 99\$
1894	011302	004767	174146		JSR PC, REVRD ;READ REVERSE SECOND RECORD
1895	011306	005215			INC (R5) ;SET 'GO' BIT
1896	011310	004767	173716		JSR PC, WAITRDY ;WAIT FOR READY
1897	011314	102417			BVS 99\$
1898	011316	004767	174132		JSR PC, REVRD ;READ REVERSE FIRST RECORD
1899	011322	004767	173026		JSR PC, TIMON ;TURN TIMER ON
1900	011326	005215			INC (R5) ;SET 'GO' BIT
1901					
1902	011330	005710		1\$: TST	(R0) ;WAIT FOR FRAME COUNT > 0
1903	011332	001002			BNE 2\$
1904	011334	000163	004444		JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
1905					
1906	011340	004767	173666	2\$: JSR	PC, WAITRDY ;WAIT FOR READY

1907	011344	102403			BVS	99\$	
1908	011346	004767	173130		JSR	PC,TIMOK	
1909	011352	000401			BR	100\$	
1910							
1911	011354	104400			99\$:	HLT	
1912	011356	104000			100\$:	SCOPE	
1913							
1914							
1915							
1916							
1917							
1918							
1919							
1920							
1921							
1922							
1923							
1924							
1925							
1926	011360	112737	000021	001122	TST021:	MOVB	#21,R#TSTNUM
1927	011366	012702	011524			MOV	#4\$,R2
1928	011372	004767	173740			JSR	PC,REWIND
(1)	011376	102530				BVS	99\$
1929	011400	005067	167510			CLR	DELTIM
1930	011404	012700	000021			MOV	#17,R0
1931	011410	004767	174004		1\$:	JSR	PC,WRITE
1932	011414	005215				INC	(R5)
1933	011416	004767	173610			JSR	PC,WAITRDY
1934	011422	102516				BVS	99\$
1935	011424	004767	173340			JSR	PC,DELAYV
1936	011430	062767	000022	167456		ADD	#18,DELTIM
1937	011436	005300				DEC	R0
1938	011440	001363				BNE	1\$
1939							
1940	011442	012700	000021			MOV	#17,R0
1941	011446	004767	174002		2\$:	JSR	PC,REVVD
1942	011452	005215				INC	(R5)
1943	011454	004767	173552			JSR	PC,WAITRDY
1944	011460	102477				BVS	99\$
1945	011462	005300				DEC	R0
1946	011464	001370				BNE	2\$
1947							
1948	011466	012700	000020			MOV	#15,R0
1949	011472	012701	001054			MOV	#GAPTBL,R1
1950	011476	004767	173734			JSR	PC,READ
1951	011502	005215				INC	(R5)
1952							
1953	011504	004767	173522		3\$:	JSR	PC,WAITRDY
1954	011510	102463				BVS	99\$
1955	011512	004767	173720			JSR	PC,READ
1956	011516	004767	172632			JSR	PC,TIMON
1957	011522	005215				INC	(R5)
1958							
1959	011524	005765	000006		4\$:	TST	FC(R5)
1960	011530	001002				BNE	5\$
1961	011532	000163	004444			JMP	TIMER(R3)

```

: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
:BEETWEEN 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 BETEWEN GO=1 TO FC > 0 STORED IN 'GAPTBL'
: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.
    
```

```

;SET RETURN PC FROM TIMER
;REWIND SLAVE
;BRANCH IF ERROR ON REWIND
;CLEAR VARIABLE DELAY TIME
;SET # OF RECORDS TO WRITE
;WRITE 17. RECORDS
;SET 'GO' BIT
;WAIT FOR READY
;DELAY BEFORE WRITING NEXT REC.
;SET NEXT DELAY TIME
;DECREMENT RECORDS WRITTEN COUNT
;SET # OF RECS. TO REVERSE READ
;REVERSE READ 17. RECORDS
;SET 'GO' BIT
;WAIT FOR READY
;DECREMENT RECORD COUNT
;SET # OF RECORDS TO READ
;SET PTR TO GAP TABLE FOR TEST
;READ A RECORD
;SET 'GO' BIT
;WAIT FOR READY
;READ NEXT RECORD
;TURN TIMER ON
;SET 'GO' BIT
;WAIT FOR FRAME COUNT > 0
;GO TO TIMER & RETURN VIA R2
    
```





2017	011736	032711	040000		BIT	#ERR, (R1)		: MONITOR ERROR BIT
2018	011742	001022			BNE	99\$		
2019	011744	000771			BR	1\$		
2020								
2021	011746			2\$:				
(1)	011746	004767	172402		JSR	PC, TIMON		: TURN TIMER ON
2022	011752	105711		3\$:	TSTB	(R1)		: WAIT FOR READY TO SET
2023	011754	100402			BMI	4\$		
2024	011756	000163	004444		JMP	TIMER(R3)		: GO TO TIMER & RETURN VIA R2
2025	011762	012700	000003	4\$:	MOV	#3, R0		: SET TO DIVIDE BY 8
2026	011766	006204		5\$:	ASR	R4		: BY SHIFTING RIGHT 3 PLACES
2027	011770	005300			DEC	R0		
2028	011772	001375			BNE	5\$		
2029	011774	004767	173232		JSR	PC, WAITRDY		
2030	012000	102403			BVS	99\$		
2031	012002	004767	172474		JSR	PC, TIMOK		: CHECK TIME
2032	012006	000401			BR	100\$		
2033								
2034	012010	104400		99\$:	HLT			
2035	012012	104000		100\$:	SCOPE			
2036								
2037								
2038	012014	112737	000024	001122	: TEST 024-DATA TIME (556BPI)			
2039	012022	012702	012102		TST024: MOV	#24, #TSTNUM		
2040	012026	004767	173304		MOV	#3, R2		: SET RETURN PC FROM TIMER
(1)	012032	102442			JSR	PC, .REWIND		: REWIND SLAVE
2041	012034	052765	000700	000032	BVS	99\$		: BRANCH IF ERROR ON REWIND
2042	012042	004367	173506		BIS	#BPI556+NORM11, TC(R5)		: LOAD TAPE CONTROL REGISTER
2043	012046	015702			JSR	R3, TMCMD		: WRITE 2224. WORD RECORD
2044	012050	173520			.WORD	WTBUF		
2045	012052	167240			.WORD	-2224.		
2046	012054	000060			.WORD	-4448.		
2047	012056	005215			.WORD	WFWD		
2048					INC	(R5)		: SET 'GO' BIT
2049	012060	022710	167240	1\$:	CMP	#-4448., (R0)		: BRANCH WHEN WRITING BEGINS
2050	012064	001004			BNE	2\$		
2051	012066	032711	040000		BIT	#ERR, (R1)		: MONITOR ERROR BIT
2052	012072	001022			BNE	99\$		
2053	012074	000771			BR	1\$		
2054								
2055	012076			2\$:				
(1)	012076	004767	172252		JSR	PC, TIMON		: TURN TIMER ON
2056	012102	105711		3\$:	TSTB	(R1)		: BRANCH WHEN READY SETS
2057	012104	100402			BMI	4\$		
2058	012106	000163	004444		JMP	TIMER(R3)		: GO TO TIMER & RETURN VIA R2
2059								
2060	012112	012700	000003	4\$:	MOV	#3, R0		: SET SHIFT COUNT
2061	012116	006204		5\$:	ASR	R4		
2062	012120	005300			DEC	R0		
2063	012122	001375			BNE	5\$		
2064	012124	004767	173102		JSR	PC, WAITRDY		
2065	012130	102403			BVS	99\$		
2066	012132	004767	172344		JSR	PC, TIMOK		: CHECK TIME
2067	012136	000401			BR	100\$		
2068								
2069	012140	104400		99\$:	HLT			

```

2070 012142 104000          100$: SCOPE
2071
2072          ;TEST 025-DATA TIME (800BPI)
2073 012144 112737 000025 001122 †TST025: MOVB #025, @#TSTNUM
2074 012152 012702 012232          MOV #3$, R2          ;SET RETURN PC FROM TIMER
2075 012156 004767 173154          JSR PC, .REWIND      ;REWIND SLAVE
(1) 012162 102442          BVS 99$             ;BRANCH IF ERROR ON REWIND
2076 012164 052765 001300 000032  BIS #BPI800+NORM11, TC(R5) ;SET 800 BPI
2077 012172 004367 173356          JSR R3, TMCMD      ;WRITE 3200. WORD RECORD
2078 012176 015702          .WORD WTBUF
2079 012200 171600          .WORD -3200.
2080 012202 163400          .WORD -6400.
2081 012204 000060          .WORD WFWO
2082 012206 005215          INC (R5)          ;SET 'GO' BIT
2083
2084 012210 022710 163400          1$: CMP #-6400., (R0) ;WAIT FOR WRITING TO START
2085 012214 001004          BNE 2$
2086 012216 032711 0-0000          BIT #ERR, (R1)    ;MONITOR ERROR BIT
2087 012222 001022          BNE 99$
2088 012224 000771          BR 1$
2089
2090          2$:
(1) 012226 004767 172122          JSR PC, TIMON      ;TURN TIMER ON
2091 012232 105711          3$: TSTB (R1)       ;BRANCH WHEN READY SETS
2092 012234 100402          BMI 4$
2093 012236 000163 004444          JMP TIMER(R3)     ;GO TO TIMER & RETURN VIA R2
2094
2095 012242 012700 000003          4$: MOV #3, R0      ;SET SHIFT COUNT
2096 012246 006204          5$: ASR R4
2097 012250 005300          DEC R0
2098 012252 001375          BNE 5$
2099 012254 004767 172752          JSR PC, WAITRDY
2100 012260 102403          BVS 99$
2101 012262 004767 172214          JSR PC, TIMOK     ;CHECK TIME
2102 012266 000401          BR 100$
2103
2104 012270 104400          99$: HLT
2105 012272 104000          100$: SCOPE
2106
2107          ;TEST 026-DATA TIME (1600BPI)
2108 012274 112737 000026 001122 †TST026: MOVB #026, @#TSTNUM
2109 012302 105737 001127          TSTB @#NRZFLG     ;BRANCH IF DRIVE 'NRZ ONLY'
2110 012306 001046          BNE TST027
2111 012310 012702 012370          MOV #3$, R2       ;SET RETURN PC FROM TIMER
2112 012314 004767 173016          JSR PC, .REWIND   ;REWIND SLAVE
(1) 012320 102437          BVS 99$           ;BRANCH IF ERROR ON REWIND
2113 012322 052765 002300 000032  BIS #PE1600+NORM11, TC(R5) ;SET 1600 BPI
2114 012330 004367 173220          JSR R3, TMCMD    ;WRITE 3200. WORD RECORD
2115 012334 015702          .WORD WTBUF
2116 012336 171600          .WORD -3200.
2117 012340 163400          .WORD -6400.
2118 012342 000060          .WORD WFWO
2119 012344 005215          INC (R5)         ;SET 'GO' BIT
2120
2121 012346 022710 163400          1$: CMP #-6400., (R0) ;BRANCH WHEN WRITING STARTS
2122 012352 001004          BNE 2$
    
```

2123	012354	032711	040000		BIT	#ERR, (R1)	: MONITOR ERROR BIT
2124	012360	001017			BNE	99\$	
2125	012362	000771			BR	1\$	
2126							
2127	012364			2\$:			
(1)	012364	004767	171764		JSR	PC, TIMON	: TURN TIMER ON
2128	012370	105711		3\$:	TSTB	(R1)	: BRANCH WHEN READY SETS
2129	012372	100402			BMI	4\$	
2130	012374	000163	004444		JMP	TIMER(R3)	: GO TO TIMER & RETURN VIA R2
2131							
2132	012400	006204		4\$:	ASR	R4	: DIVIDE TIME BY 4
2133	012402	006204			ASR	R4	
2134	012404	004767	172622		JSR	PC, WAITRDY	
2135	012410	102403			BVS	99\$	
2136	012412	004767	172064		JSR	PC, TIMOK	: CHECK TIME
2137	012416	000401			BR	100\$	
2138							
2139	012420	104400		99\$:	HLT		
2140	012422	104000		100\$:	SCOPE		
2141							
2142							
2143							
2144	012424	112737	000027	001122	; TEST-027-ERASE		
2145	012432	012702	012460		; THIS TEST MEASURES TIME FROM 'GO'=1 TO 'RDY'=1.		
2146	012436	004337	005554		TST027:	MOVB	#27, @#TSTNUM
2147	012442	000000				MOV	#1\$, R2
2148	012444	000000				JSR	R3, @#TMCMD
2149	012446	000000				.WORD	0
2150	012450	000024				.WORD	0
2151	012452	004767	171676			.WORD	0
2152	012456	005215				.WORD	ERASE
2153						JSR	PC, TIMON
2154	012460	105711		1\$:	TSTB	(R1)	: BRANCH WHEN READY SETS
2155	012462	100402			BMI	2\$	
2156	012464	000163	004444		JMP	TIMER(R3)	: GO TO TIMER & RETURN VIA R2
2157							
2158	012470	004767	172536	2\$:	JSR	PC, WAITRDY	
2159	012474	102403			BVS	99\$	
2160	012476	004767	172000		JSR	PC, TIMOK	
2161	012502	000401			BR	100\$	
2162							
2163	012504	104400		99\$:	HLT		
2164	012506	104000		100\$:	SCOPE		
2165							
2166							
2167							
2168	012510	112737	000030	001122	; TEST-030 TAPE MARK		
2169	012516	012702	012560		; THIS TEST MEASURES TIME FROM 'GO'=1 TO 'RDY'=1.		
2170	012522	004767	172672		TST030:	MOVB	#30, @#TSTNUM
2171	012526	005215				MOV	#1\$, R2
2172	012530	004767	172476			JSR	PC, WRITE
2173	012534	102423				INC	(R5)
2174	012536	004337	005554			JSR	PC, WAITRDY
2175	012542	000000				BVS	99\$
2176	012544	000000				JSR	R3, @#TMCMD
2177	012546	000000				.WORD	0
						.WORD	0
						.WORD	0

2178	012550	000026		.WORD	WFMK	
2179	012552	004767	171576	JSR	PC.TIMON	;TURN TIMER ON
2180	012556	005215		INC	(R5)	;SET 'GO' BIT
2181						
2182	012560	105711		1\$: TSTB	(R1)	;BRANCH WHEN READY SETS
2183	012562	100402		BMI	2\$	
2184	012564	000163	004444	JMP	TIMER(R3)	;GO TO TIMER 3 RETURN VIA R2
2185						
2186	012570	004767	172436	2\$: JSR	PC.WAITRDY	
2187	012574	102403		BVS	99\$	
2188	012576	004767	171700	JSR	PC.TIMOK	
2189	012602	000401		BR	100\$	
2190						
2191	012604	104400		99\$: HLT		
2192	012606			100\$:		
(1)	012606	004767	172524	JSR	PC.REWIND	;REWIND SLAVE
(1)	012612	102774		BVS	99\$	;BRANCH IF ERROR ON REWIND
2193	012614	104000		SCOPE		
2194						

2196	012616	012700	000012		FINISH: MOV	#10.,R0		:SET LINE FEED COUNT
2197	012622	000004	001374		1\$: TYPE,CRLF			
2198	012626	005300			DEC	R0		
2199	012630	001374			BNE	1\$		
2200	012632	032777	000100	166140	BIT	#SW06,@SWR		
2201	012640	001410			BEQ	2\$		
2202	012642	113700	001004		MOVB	@#DRVNUM,R0		
2203	012646	113701	001005		MOVB	@#SLVNUM,R1		
2204	012652	113702	001006		MOVB	@#SLVPTR,R2		
2205	012656	000137	007210		JMP	@#TYPHDR		
2206	012662	105237	001005		2\$: INCB	@#SLVNUM		:SET NEXT SLAVE #
2207	012666	005237	001006		INC	@#SLVPTR		:AND ITS POINTER
2208	012672	122737	000010	001005	CMPB	#8,@#SLVNUM		:BRANCH IF LAST SLAVE (7)
2209	012700	001402			BEQ	3\$		
2210	012702	000137	006764		JMP	@#BEGIN		:BEGIN TEST ON NEXT SLAVE
2211	012706	105037	001005		3\$: CLRB	@#SLVNUM		:SET SLAVE #0
2212	012712	105237	001004		INCB	@#DRVNUM		:AND INCREMENT DRIVE #
2213	012716	122737	000010	001004	CMPB	#8,@#DRVNUM		:AND CHECK IF LAST DRIVE
2214	012724	001402			BEQ	END		
2215	012726	000137	006764		JMP	@#BEGIN		
2216								
2217	012732	105737	001125		END: TSTB	@#UNTFND		:BRANCH IF A UNIT WAS FOUND
2218	012736	001004			BNE	1\$		
2219	012740	000004	014050		TYPE,E.UNIT			
2220	012744	000137	005724		JMP	@#INIT		
2221	012750	000000			1\$: HALT			
2222	012752	004767	167012		JSR	PC,CKSWR		:CHECK FOR CNTL G
2223	012756	000005			RESET			
2224	012760	000137	005724		JMP	@#INIT		:RESTART

```

2226 ;SKEW TAPE TIMING TESTS
2227 ;THE FOLLOWING TESTS REQUIRE A SPECIALLY WRITTEN 800 BPI SKEW TAPE
2228 012764 012737 012772 001002 SKEWTST:MOV #TST031,#SCPADR ;SET SCOPE POINTER
2229
2230 ;TEST 031- SKEW TAPE SPEED TEST-FORWARD
2231 ;THIS TEST READS 32" OF TAPE (26400.-800. = 25600. FRAMES) THEN
2232 ;DIVIDES TIME BY 32. TO GET TIME TO READ 1" (800. FRAMES) OF TAPE.
2233 012772 112737 000031 001122 †TST031: MOV #31,#TSTNUM
2234 013000 012702 013056 MOV #25,R2 ;SET RETURN PC FROM TIMER
2235 013004 004767 172326 JSR PC,REWIND ;REWIND SLAVE
(1) 013010 102441 BVS 99$ ;BRANCH IF ERROR ON REWIND
2236 013012 052765 001300 000032 BIS #BPI800+NORM11,TC(R5) ;SET 800 BPI
2237 013020 052765 000010 000010 BIS #BAI,CS2(R5) ;INHIBIT BUS ADDRESS INCREMENT
2238 013026 004337 005554 JSR R3,#TMCMD ;READ 32" OF TAPE-FORWARD
2239 013032 015702 .WORD RDBUF
2240 013034 177777 .WORD -1.
2241 013036 063440 10$: .WORD 26400. ;FRAME COUNT
2242 013040 000070 .WORD RDFWD
2243 013042 005215 INC (R5) ;SET 'GO' BIT
2244
2245 013044 022710 001440 1$: CMP #800.,(R0) ;WAIT FOR FIRST 800 FRAMES
2246 013050 101375 BHI 1$ ;TO BE READ
2247
2248 013052 004767 171276 JSR PC,TIMON ;TURN TIMER ON
2249 013056 023710 013036 2$: CMP #10$, (R0) ;WAIT FOR READING TO FINISH
2250 013062 103402 BLO 3$
2251 013064 000163 004444 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2252
2253 013070 012700 000005 3$: MOV #5,R0 ;DIVIDE TIME BY 32.
2254 013074 006204 4$: ASR R4
2255 013076 005300 DEC R0
2256 013100 001375 BNE 4$
2257 013102 004767 172066 JSR PC,RHINIT ;INIT DRIVE
2258 013106 004767 171370 JSR PC,TIMOK ;CHECK TIME
2259 013112 000401 BR 100$
2260
2261 013114 104400 99$: HLT
2262 013116 104000 100$: SCOPE
2263
2264 ;TEST 032-SKEW TAPE SPEED TEST-REVERSE
2265 ;THIS TEST READS FORWARD 40" (32000. FRAMES) OF TAPE, THEN READS REVERSE
2266 ;32" (26400.-800. = 25600. FRAMES) OF TAPE. THE TIME IS THEN DIVIDED BY
2267 ;32. TO GET TIME TO READ 1" (800. FRAMES) OF TAPE.
2268 013120 112737 000032 001122 †TST032: MOV #32,#TSTNUM
2269 013126 012702 013254 MOV #3$,R2 ;SET RETURN PC FROM TIMER
2270 013132 004767 172200 JSR PC,REWIND ;REWIND SLAVE
(1) 013136 102465 BVS 99$ ;BRANCH IF ERROR ON REWIND
2271 013140 052765 001300 000032 BIS #BPI800+NORM11,TC(R5)
2272 013146 052765 000010 000010 BIS #BAI,CS2(R5)
2273 013154 004337 005554 JSR R3,#TMCMD ;READ FORWARD 32000. FRAMES
2274 013160 015702 .WORD RDBUF
2275 013162 177777 .WORD -1.
2276 013164 076400 10$: .WORD 32000. ;WORD COUNT
2277 013166 000070 .WORD RDFWD ;FRAME COUNT
2278 013170 005215 INC (R5) ;READ FORWARD
;SET 'GO' BIT
2279
    
```

2280	013172	023710	013164	1\$:	CMP	Q#10\$, (R0)	
2281	013176	101375			BHI	1\$	
2282							
2283	013200	004767	171770		JSR	PC, RHINIT	; INIT DRIVE
2284	013204	004767	171530		JSR	PC, DELAY	; WAIT FOR TAPE MOTION TO STOP
2285	013210	052765	001300	00032	BIS	#BPI800+NORM11, TC(R5)	; SET 800 BPI
2286	013216	052765	000010	000310	BIS	#BAI, CS2(R5)	; INHIBIT BUS ADDRESS INCREMENT
2287	013224	004337	005554		JSR	R3, Q#TMCMD	; READ REVERSE 32" OF TAPE
2288	013230	015702			.WORD	R0BUF	; READ BUFFER
2289	013232	177777			.WORD	-1.	; WORD COUNT
2290	013234	063440		11\$:	.WORD	26400.	; FRAME COUNT
2291	013236	000076			.WORD	RDREV	; READ REVERSE
2292	013240	005215			INC	(R5)	; SET 'GO' BIT
2293							
2294	013242	022710	001440	2\$:	CMP	#800., (R0)	; WAIT FOR FIRST 800 FRAMES
2295	013246	101375			BHI	2\$	; TO BE READ
2296							
2297	013250	004767	171100		JSR	PC, TIMON	; TURN TIMER ON
2298	013254	023710	013234	3\$:	CMP	Q#11\$, (R0)	; WAIT FOR ALL FRAMES TO BE READ
2299	013260	103402			BLO	4\$	
2300	013262	000163	004444		JMP	TIMER(R3)	; GO TO TIMER & RETURN VIA R2
2301							
2302	013266	012700	000005	4\$:	MOV	#5, R0	; DIVIDE TIME BY 32.
2303	013272	006204		5\$:	ASR	R4	
2304	013274	005300			DEC	R0	
2305	013276	001375			BNE	5\$	
2306	013300	004767	171670		JSR	PC, RHINIT	
2307	013304	004767	171172		JSR	PC, TIMOK	
2308	013310	000401			RR	100\$	
2309							
2310	013312	104400		99\$:	HLT		
2311	013314			100\$:			
(1)	013314	004767	172016		JSR	PC, .REWIND	; REWIND SLAVE
(1)	013320	102774			BVS	99\$	; BRANCH IF ERROR ON REWIND
2312	013322	104000			SCOPE		
2313							
2314	013324	000137	012616		JMP	Q#FINISH	
2315							
2316							
2317							



```

2319 .SBTTL PROGRAM MESSAGES
2320 ;OPERATOR INSTRUCTIONS
2321 M.NAM: .ASCIZ <CR><LF>'TU16J DRIVE FUNCTION TIMER (DZTUG-A)'
```

013330	005015	052524	033061		
013336	020112	051104	053111		
013344	020105	052506	041516		
013352	044524	047117	052040		
013360	046511	051105	024040		
013366	055104	052524	026507		
013374	024501	000			
2322	013377	015	052012	050131	I.REG: .ASCIZ <CR><LF>'TYPE FIRST ADDRESS OF CONTROLLER '
	013404	020105	044506	051522	
	013412	020124	042101	051104	
	013420	051505	020123	043117	
	013426	041440	047117	051124	
	013434	046117	042514	020122	
	013442	000040			
2323	013444	054524	042520	052040	I.DRVS: .ASCIZ %TYPE TMO2 DRIVE #'S TO BE TESTED %
	013452	030115	020062	051104	
	013460	053111	020105	023443	
	013466	020123	047524	041040	
	013474	020105	042524	052123	
	013502	042105	000040		
2324	013506	047506	020122	046524	I.SLVS: .ASCII 'FOR TMO2 DRIVE '
	013514	031060	042040	044522	
	013522	042526	040		
2325	013525	060	020055	054524	I.DRV: .ASCIZ %0- TYPE SLAVE #'S TO BE TESTED %
	013532	042520	051440	040514	
	013540	042526	021440	051447	
	013546	052040	020117	042502	
	013554	052040	051505	042524	
	013562	020104	000		
2326	013565	123	042520	042105	I.SKEW: .ASCIZ 'SPEED TESTS ONLY? (YES/NO = 1/0)'
	013572	052040	051505	051524	
	013600	047440	046116	037531	
	013606	024040	042531	027523	
	013614	047516	036440	030440	
	013622	030057	000051		
2327	013626	051116	020132	047117	I.NRZ: .ASCIZ 'NRZ ONLY? (YES/NO = 1/0)'
	013634	054514	020077	054450	
	013642	051505	047057	020117	
	013650	020075	027461	024460	
	013656	000			
2328	013657	015	042412	042116	M.EOT: .ASCIZ <CR><LF>'END OF TAPE'<CR><LF>
	013664	047440	020106	040524	
	013672	042520	005015	000	
2329					
2330					;ERROR MESSAGES
2331	013677	015	052012	040522	E.TRP4: .ASCIZ <CR><LF>'TRAPPED TO 4'
	013704	050120	042105	052040	
	013712	020117	000064		
2332	013716	047516	041440	047117	E.NCON: .ASCIZ 'NO CONTROLLER AT ADDRESS SPECIFIED'<CR><LF>
	013724	051124	046117	042514	
	013732	020122	052101	040440	
	013740	042104	042522	051523	
	013746	051440	042520	044503	
	013754	044506	042105	005015	

```

2333 013762 000
      013763 124 030115 020062 E.NDRV: .ASCIZ 'TMO2 DRIVE '
      013770 051104 053111 020105
      013776 000
2334 013777 104 044522 042526 E.NSLV: .ASCII 'DRIVE '
      014004 040
2335 014005 060 051440 040514 E.DRV: .ASCII 'D SLAVE '
      014012 042526 040
2336 014015 060 047040 052117 E.NAVA: .ASCIZ 'D NOT AVAILABLE FOR TEST'<CR><LF>
      014022 040440 040526 046111
      014030 041101 042514 043040
      014036 051117 052040 051505
      014044 006524 000012
2337 014050 047516 052040 030115 E.UNIT: .ASCIZ 'NO TMO2/TU16J UNIT FOUND TO TEST'<CR><LF>
      014056 027462 052524 033061
      014064 020112 047125 052111
      014072 043040 052517 042116
      014100 052040 020117 042524
      014106 052123 005015 000
2338 014113 123 043117 020124 E.SFT: .ASCIZ 'SOFT ERROR (DATA)'<CR><LF>
      014120 051105 047522 020122
      014126 042050 052101 024501
      014134 005015 000
2339 014137 124 051505 020124 E.HDR: .ASCIZ 'TEST # '
      014144 020043 000
2340 014147 040 042504 044526 E.HDR1: .ASCII ' DEVICE ERROR'<CR><LF>
      014154 042503 042440 051122
      014162 051117 005015
2341 014166 051503 004461 041527 .ASCIZ 'CS1'<HT>'WC'<HT>'BA'<HT>'FC'<HT>'CS2'<HT>'DS'<HT>'ER'<HT>'TC'<CR><LF>
      014174 041011 004501 041506
      014202 041411 031123 042011
      014210 004523 051105 052011
      014216 006503 000012
2342 014222 047440 052125 047440 E.HDR2: .ASCIZ ' OUT OF RANGE ERROR'<CR><LF>
      014230 020106 040522 043516
      014236 020105 051105 047522
      014244 006522 000012
2343 014250 005015 044524 042515 E.TIMOV: .ASCIZ <CR><LF>'TIMER OVERFLOWED'<CR><LF>
      014256 020122 053117 051105
      014264 046106 053517 042105
      014272 005015 000
2344 014275 015 052012 046511 E.TIMEX: .ASCIZ <CR><LF>'TIME EXPIRED WAITING FOR RDY'<CR><LF>
      014302 020105 054105 044520
      014310 042522 020104 040527
      014316 052111 047111 020107
      014324 047506 020122 042122
      014332 006531 000012
2345 014336 043440 050101 021440 E.GAP: .ASCIZ ' GAP # '
      014344 000040
2346
2347 ;TIME DOCUMENT LINES
2348 014346 025052 025052 025052 [.HDR1: .ASCIZ '*****
      014354 025052 025052 025052
      014362 025052 025052 025052
      014370 025052 025052 025052
      014376 025052 025052 025052

```

```

014404 025052 025052 025052
014412 025052 025052 025052
014420 025052 025052 025052
014426 025052 025052 025052
014434 025052 025052 025052
014442 025052 025052 025052
014450 025052 025052 025052
014456 005015 000
2349 014461 052 052040 030115 L.HDR2: .ASCII '* TMO2 DRIVE FUNCTION TIMES- DRIVE #'
014466 020062 051104 053111
014474 020105 052506 041516
014502 044524 047117 052040
014510 046511 051505 020055
014516 051104 053111 020105
014524 020043
2350 014526 020060 046123 053101 L.DRV: .ASCII '0 SLAVE #'
014534 020105 020043
2351 014540 020060 040 L.SLV: .ASCII '0 '
2352 014543 071 041440 040510 L.CHAN: .ASCIZ '9 CHAN. SER #'
014550 027116 051440 051105
014556 021440 000040
2353 014562 006440 025012 005015 L.HDR3: .ASCII '* <CR><LF>' * <CR><LF>
2354 014570 020052 052506 041516 .ASCIZ '* FUNCTION' <HT><HT>' TIME(SPECIFICATION)' <HT>' TIME(ACTUAL)' <CR><LF>
014576 044524 047117 004411
014604 044524 042515 051450
014612 042520 044503 044506
014620 040503 044524 047117
014626 004451 044524 042515
014634 040450 052103 040525
014642 024514 005015 000
2355
2356 014647 122 047101 042507 L.RNG: .ASCIZ 'RANGE='
014654 036075 000
2357 014657 101 052103 040525 L.ACT: .ASCIZ 'ACTUAL='
014664 036514 000
2358
2359 :TEST DESCRIPTOR HEADERS
2360 014667 052 053440 044522 A.T001: .ASCIZ '* WRITE FROM BOT' <HT>
014674 042524 043040 047522
014702 020115 047502 004524
014710 000
2361 014711 052 053440 044522 A.T002: .ASCIZ '* WRITE START' <HT><HT>
014716 042524 051440 040524
014724 052122 004411 000
2362 014731 052 053440 044522 A.T003: .ASCIZ '* WRITE SHUTDOWN' <HT>
014736 042524 051440 052510
014744 042124 053517 004516
014752 000
2363 014753 052 053440 044522 A.T004: .ASCIZ '* WRITE SETTLEDOWN' <HT>
014760 042524 051440 052105
014766 046124 042105 053517
014774 004516 000
2364 014777 052 051040 040505 A.T005: .ASCIZ '* READ FROM BOT' <HT><HT>
015004 020104 051106 046517
015012 041040 052117 004411
015020 000

```

2365	015021	052	051040	040505	A.TO06: .ASCIZ	'* READ START' <HT><HT>
	015026	020104	052123	051101		
	015034	004524	000011			
2366	015040	020052	042522	042101	A.TO07: .ASCIZ	'* READ SHUTDOWN' <HT><HT>
	015046	051440	052510	042124		
	015054	053517	004516	000011		
2367	015062	020052	042522	042101	A.TO10: .ASCIZ	'* READ SETTLEDOWN' <HT>
	015070	051440	052105	046124		
	015076	042105	053517	004516		
	015104	000				
2368	015105	052	051040	040505	A.TO11: .ASCIZ	'* READ REV START' <HT>
	015112	020104	042522	020126		
	015120	052123	051101	004524		
	015126	000				
2369	015127	052	051040	040505	A.TO12: .ASCIZ	'* READ REV SHUTDOWN' <HT>
	015134	020104	042522	020126		
	015142	044123	052125	047504		
	015150	047127	000011			
2370	015154	020052	042522	042101	A.TO13: .ASCIZ	'* READ REV SETTLEDOWN' <HT>
	015162	051040	053105	051440		
	015170	052105	046124	042105		
	015176	053517	004516	000		
2371	015203	052	052040	051125	A.TO14: .ASCIZ	'* TURN AROUND DELAY F-R' <HT>
	015210	020116	051101	052517		
	015216	042116	042040	046105		
	015224	054501	043040	051055		
	015232	000011				
2372	015234	020052	052524	047122	A.TO15: .ASCIZ	'* TURN AROUND DELAY R-F' <HT>
	015242	040440	047522	047125		
	015250	020104	042504	040514		
	015256	020131	026522	004506		
	015264	000				
2373	015265	052	043440	050101	A.TO16: .ASCIZ	'* GAP SIZE-STOP HALF' <HT>
	015272	051440	055111	026505		
	015300	052123	050117	044040		
	015306	046101	004506	000		
2374	015313	052	043440	050101	A.TO17: .ASCIZ	'* GAP SIZE-START HALF' <HT>
	015320	051440	055111	026505		
	015326	052123	051101	020124		
	015334	040510	043114	000011		
2375	015342	020052	040507	020120	A.TO20: .ASCIZ	'* GAP SIZE-INTERRECORD' <HT>
	015350	044523	042532	044455		
	015356	052116	051105	042522		
	015364	047503	042122	000011		
2376	015372	020052	040507	020120	A.TO21: .ASCIZ	'* GAP CONSISTANCY' <HT>
	015400	047503	051516	051511		
	015406	040524	041516	004531		
	015414	000				
2377	015415	052	042040	052101	A.TO23: .ASCIZ	'* DATA TIME-200BPI' <HT>
	015422	020101	044524	042515		
	015430	031055	030060	050102		
	015436	004511	000			
2378	015441	052	042040	052101	A.TO24: .ASCIZ	'* DATA TIME-556BPI' <HT>
	015446	020101	044524	042515		
	015454	032455	033065	050102		
	015462	004511	000			

G07

SEQ 0084

```

2379 015465 052 042040 052101 A.T025: .ASCIZ '* DATA TIME-800BPI'<HT>
      015472 020101 044524 042515
      015500 034055 030060 050102
      015506 004511 000
2380 015511 052 042040 052101 A.T026: .ASCIZ '* DATA TIME-1600BPI'<HT>
      015516 020101 044524 042515
      015524 030455 030066 041060
      015532 044520 000011
2381 015536 020052 051105 051501 A.T027: .ASCIZ '* ERASE GAP TIME'<HT>
      015544 020105 040507 020120
      015552 044524 042515 000011
2382 015550 020052 051127 052111 A.T030: .ASCIZ '* WRITE FILE MARK'<HT>
      015566 020105 044506 042514
      015574 046440 051101 004513
      015602 000
2383 015603 052 052040 050101 A.T031: .ASCIZ '* TAPE SPEED-FWD'<HT>
      015610 020105 050123 042505
      015616 026504 053506 004504
      015624 000
2384 015625 052 052040 050101 A.T032: .ASCIZ '* TAPE SPEED-REV'<HT>
      015632 020105 050123 042505
      015640 026504 042522 004526
      015646 000
2385
2386 015647 015 057012 000107 L.CNTG: .ASCIZ <CR><LF>'↑G'
2387 015654 005015 053523 036522 L.SWR: .ASCIZ <CR><LF>'SWR='
      015662 000
2388 015663 040 047040 053505 L.NEW: .ASCIZ ' NEW='
      015670 020075 000
2389 015673 015 037412 005015 L.QUEST: .ASCIZ <CR><LF>'?'<CR><LF>
      015700 000
2390 015702 .EVEN
2391 015702 RDBUF=.
2392 015702 WTBUF=.
2393 015702 .BLKW 128.
2394 000001 .END

```

A	010620	CNTRLO=	000017	E.HDR	014137	L.HDR3	014562	RESVEC=	000010
ACCL	= 100000	CNTRLU=	000025	E.HDR1	014147	L.NEW	015663	REVRD	005454
ANGTAB	001412	CNVDEC	002730	E.HDR2	014222	L.QUES	015673	RHINIT	005174
AS	= 000016	CNVOC	002622	E.NAVA	014015	L.RNG	014647	RMR	= 000004
ASFLG	001130	CNVTA0	003260	E.NCON	013716	L.SLV	014540	RWD	= 000006
ATA	= 100000	CNVTD	002742	E.NDRV	013763	L.SWR	015654	RWDOFF=	000002
ATIME	001012	CNVTO	002634	E.NSLV	013777	MCPE	= 020000	R10	=%000000
ATIMTB	001014	COUNT	001764	E.SFT	014113	MDPE	= 000400	R11	=%000001
A.T001	014667	CR	= 000015	E.TIME	014275	MMVEC	= 000250	R12	=%000002
A.T002	014711	CRLF	001374	E.TIMO	014250	MOL	= 010000	R13	=%000003
A.T003	014731	CSITM	= 002000	E.TRP4	013677	MR	= 000024	R14	=%000004
A.T004	014753	CS1	= 000000	E.UNIT	014050	MXF	= 001000	R15	=%000005
A.T005	014777	CS2	= 000010	FC	= 000006	M.EOT	013657	SC	= 100000
A.T006	015021	DASH	001405	FCE	= 001000	M.NAM	013330	SCOPE	= 104000
A.T007	015040	DB	= 000022	FINISH	012616	NAMPTR	001672	SCPADR	001002
A.T010	015062	DCONST	003036	FMT	= 000020	NEC	= 010000	SDWN	= 000020
A.T011	015105	DELAY	004740	FPEVEC=	000244	NEF	= 004000	SKEWTS	012764
A.T012	015127	DELAYV	004770	FRMCNT=	177400	NEM	= 004000	SLA	= 000001
A.T013	015154	DELTIM	001114	FWDSPC	005472	NOP	= 000000	SLAVES	006370
A.T014	015203	DIGTAB	001132	GAP	001120	NORM11=	000300	SLR	= 177774
A.T015	015234	DIVIDE	005026	GAPOK	004612	NRZFLG	001127	SLVAVA	005144
A.T016	015265	CLT	= 100000	GAPTBL	001054	NSG	= 000400	SLVNUM	001005
A.T017	015313	DPR	= 000400	GO	= 000001	OCTALO	001116	SLVPTR	001006
A.T020	015342	DRIVES	006140	GTIMTB	001572	ODIGIT	001144	SLVTBL	001164
A.T021	015372	DRVAVA	005116	HLT	= 104400	CPI	= 020000	SN	= 000030
A.T023	015415	DRVNUM	001004	HT	= 000011	OR	= 000200	SNPT	005574
A.T024	015441	DRVVTBL	001154	IDB	= 000010	OSC	= 000100	SPACE	001410
A.T025	015465	DRY	= 000200	IE	= 000100	OJT	002240	SPACE2	001407
A.T026	015511	DRYCLR=	000010	I_F	= 000001	OUTBUF=	005724	SPCFWD=	000030
A.T027	015536	DS	= 000012	ILR	= 000002	OUTGAP	003146	SPCREV=	000032
A.T030	015560	DT	= 000026	INBUF	001264	OUT*SPC	003052	SPR	= 002000
A.T031	015603	DTE	= 010000	INCVAE=	000100	PARVEC=	000114	SSC	= 000100
A.T032	015625	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	013525	PFVEC	= 000024	SWREG	000176
BEGIN	006764	DV4	= 000004	I.DRVS	013444	PGE	= 002000	SW06	= 000100
BELL	001403	DV5	= 000005	I.NRZ	013626	PIP	= 020000	SW07	= 000200
BKSLSH	001377	DV6	= 000006	I.REG	013377	PIRQ	= 177772	SW08	= 000400
BOT	= 000002	DV7	= 000007	I.SKEW	013565	PIRVEC=	000240	SW09	= 001000
BPI200=	000000	ECHO	001401	I.SLVS	013506	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	014657	RDBUF	= 015702	TBITVE=	000014
CH7	= 010000	ERR	= 040000	L.CHAN	014543	RDFWD	= 000070	TC	= 000032
CKSWR	001770	ERRTRP	003650	L.CNTG	015647	RDRV	= 000076	T.CRLF	002360
CLR	= 000040	ERRVEC=	000004	L.DRV	014526	RDSW	001766	TEMPST	001762
CNTLU	002030	E.DRV	014005	L.HDR1	014346	RDY	= 000200	TIB	001760
CNTRLC=	000003	E.GAP	014336	L.HDR2	014461	READ	005436	TIMER	004444

TIMERR	004454	TSTNUM	001122	TST023	011672	TYPOCT	002630	\$CNTRL	002325
TIMERO	004470	TST001	007336	TST024	012014	UBREAK=	177770	\$CRLF	002326
TIMER1	004420	TST002	007422	TST025	012144	UNS =	040000	\$FILL	002315
T.MOK	004502	TST003	007500	TST026	012274	UNTFND	001125	\$HI =	000011
TIMON	004354	TST004	007570	TST027	012424	UPE =	020000	\$NULL	002314
TFB =	177562	TST005	007676	TST030	012510	WAITRD	005232	\$TKFLG	002317
TKISR	003606	TST006	007762	TST031	012772	WAITTI	005234	\$TPB	002322
TKS =	177560	TST007	010046	TST032	013120	WC =	000002	\$TPFLG	002316
TKVEC =	000060	TST010	010146	TTIN	002242	WCE =	040000	\$TPS	002320
TMBASE	001010	TST011	010266	TTIN1	002262	WCHKF =	000050	. =	016302
TMCMD	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	WFWD =	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 =	015702		
TRTVEC=	000014	TST022	011664	TYPHDR	007210	\$CHARC	002324		

. ABS. 016302 000

ERRORS DETECTED: 0  
 DEFAULT GLOBALS GENERATED: 0

DZTUGA,DZTUGA+DZTUGA.P11  
 RUN-TIME: 11 17 1 SECONDS  
 RUN-TIME RATIO: 64/30=2.1  
 CORE USED: 7K (13 PAGES)

J07