

# TM02/TU45

DRIVE FUNCTION TIMER  
CZTULA0

AH-E479A-MC  
COPYRIGHT © 75-78  
FICHE 1 OF 1

JUL 1978  
**digital**  
MADE IN USA

The table contains 12 columns and 12 rows of small, illegible technical diagrams and text blocks. Each cell appears to contain a small chart or a set of data points, but the text is too small to read. The diagrams likely represent various drive function parameters or timing characteristics for the TM02/TU45 device.



.NLIST SEQ,LOC,BIN  
.REM\_

#### IDENTIFICATION

PRODUCT CODE: AC-E478A-MC  
PRODUCT NAME: CZTULAO TMO2/TU45 DRIVE FUNCTION TIMER  
DATE CREATED: 25 MAY 1978  
MAINTAINER: COMPUTER SPECIAL SYSTEMS  
AUTHOR: JOHN ADAMS/R. BARNES/R. J. COLLINS

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

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

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

TMO2 DRIVE FUNCTION TIMER  
TABLE OF CONTENTS

PAGE 2

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

TMO2 DRIVE FUNCTION TIMER  
ABSTRACT

PAGE 3

ABSTRACT

PROGRAM CZTULA MEASURES THE TIME REQUIRED AND GAP SIZES PRODUCED BY THE TMO2/TU45 MAGTAPE DRIVE/SLAVE.

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

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

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

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

TMO2 DRIVE FUNCTION TIMER  
REQUIREMENTS

PAGE 4

CHAPTER 1  
REQUIREMENTS

PDP-11 FAMILY CENTRAL PROCESSOR WITH 4K MEMORY WITH UP TO 64 TM11/TMO2  
CONTROLLER/MAGTAPE STATIONS.

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)

CZTUKA CONTROL LOGIC TEST  
CZTUJA BASIC FUNCTION TEST

TMO2 DRIVE FUNCTION TIMER  
LOADING AND STARTING PROCEDURE

PAGE 5

## CHAPTER 2

### LOADING AND STARTING PROCEDURE

THE PROCEDURE IS AS FOLLOWS:

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

PROGRAM WILL REQUEST DRIVE (TMO2) AND SLAVE (TU45) 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.

TMO2 DRIVE FUNCTION TIMER  
SWITCH SETTINGS

PAGE 6

CHAPTER 3

SWITCH SETTINGS

IF A CONSOLE SWITCH REGISTER IS NOT PRESENT, THE FOLLOWING  
PROCEDURE MUST BE IMPLEMENTED:

- A) LOAD ADDRESS 1000(8) LABELLED "SWR"
- B) DEPOSIT THE VALUE 176(8)
- C) LOAD ADDRESS 176(8)
- D) DEPOSIT THE DESIRED SWITCH VALUE.

SW15	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	LOOP SUBTEST	THIS SWITCH WHEN SET LOOPS THE CURRENT SUBTEST REGARDLESS OF ERROR CONDITION.
SW13	INHIBIT ERROR TYPEOUT	THIS SWITCH WHEN SET INHIBITS ERROR TYPEOUT.
SW11	INHIBIT SUB- TEST ITERATION	THIS SWITCH WHEN SET CAUSES EACH SUBTEST TO BE EXECUTED ONLY ONCE. (INITIAL STARTUP ONLY).
SW10	INHIBIT FUNCTION TIME PUBLICATION	THIS SWITCH WHEN SET WILL INHIBIT THE PRINTING OF THE FUNCTION TIMES. (SEE CHAPTER 8.)
SW09	RING BELL ON ERROR	THIS SWITCH WHEN SET WILL RING THE BELL ON THE TTY WHEN AN ERROR IS DETECTED.
SW07	HALT AFTER SELECTED TEST	THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO HALT AFTER THE TEST SELECTED IN SW05-SW00 IS EXECUTED.
SW06	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.



TMO2 DRIVE FUNCTION TIMER  
ERRORS

PAGE 7

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-III III = 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 = CCCCCCC

TMO2 DRIVE FUNCTION TIMER  
SUBROUTINE ABSTRACTS

PAGE 8

## 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.

### 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.

TMO2 DRIVE FUNCTION TIMER  
SUBROUTINE ABSTRACTS

PAGE 9

5.3 .HLT

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

TMO2 DRIVE FUNCTION TIMER  
MISCELLANEOUS

PAGE 10

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 3.5 MIN.

WHEN SW11=0 (ITERATE SUBTESTS) THE TIME REQUIRED IS 4.5 MIN.

TMO2 DRIVE FUNCTION TIMER  
PROGRAM DESCRIPTION

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=<080000-068000>	ACTUAL=074000
* WRITE START	RANGE=<005500-004500>	ACTUAL=005000
* WRITE SHUTDOWN	RANGE=<004500-003600>	ACTUAL=004050
* WRITE SETTLEDOWN	RANGE=<006000-004600>	ACTUAL=005300
* READ FROM BOT	RANGE=<009500-007900>	ACTUAL=008700
* READ START	RANGE=<001800-001500>	ACTUAL=001650
* READ SHUTDOWN	RANGE=<001700-001300>	ACTUAL=001500
* READ SETTLEDOWN	RANGE=<006000-004600>	ACTUAL=005300
* READ REV START	RANGE=<001800-001500>	ACTUAL=001650
* READ REV SHUTDOWN	RANGE=<002300-001900>	ACTUAL=002100
* READ REV SETTLEDOWN	RANGE=<006000-004600>	ACTUAL=005300
* TURN AROUND DELAY F-R	RANGE=<008000-006000>	ACTUAL=007000
* TURN AROUND DELAY R-F	RANGE=<008000-006000>	ACTUAL=007000
* GAP SIZE-STOP HALF	RANGE=<008000-006000>	ACTUAL=007000
* GAP SIZE-START HALF	RANGE=<008000-006000>	ACTUAL=007000
* GAP SIZE-INTERRECORD	RANGE=<008000-006600>	ACTUAL=007300
* GAP CONSISANCY	RANGE=<008000-006800>	ACTUAL=007400
* DATA TIME-200 BPI	RANGE=<014560-013710>	ACTUAL=014135
* DATA TIME-556 BPI	RANGE=<014560-013710>	ACTUAL=014135
* DATA TIME-800BPI	RANGE=<014560-013710>	ACTUAL=014135
* DATA TIME-1600BPI	RANGE=<015000-013000>	ACTUAL=014000
* ERASE GAP TIME	RANGE=<061000-055000>	ACTUAL=058000
* WRITE FILE MARK	RANGE=<064000-058000>	ACTUAL=061000

TMO2 DRIVE FUNCTION TIMER

PAGE 12

7.1.1 SAMPLE TIME DOCUMENT FOR TAPE SPEED TESTS

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

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

*FUNCTION	TIME(SPECIFICATION)	TIME(ACTUAL)
*TAPE SPEED FWD	RANGE=<013800-013000>	ACTUAL=013330
*TAPE SPEED REV	RANGE=<013800-013000>	ACTUAL=013330

TMO2 DRIVE FUNCTION TIMER

7.2 TEST SEQUENCE WITH RELATED ADJUSTMENTS AND ASSOCIATED HARDWARE

TEST NO./NAME	RELATED ADJUSTMENTS	ASSOCIATED HARDWARE
1. WRITE FROM BOT	*NONE	*M8921 ROM*M8903 ACCL CNTR
2. WRITE START	* "	* " * "
3. WRITE SHUTDOWN	* "	* " * "
4. WRITE SETTLEDOWN	* "	*M8921 SETTLEDOWN ONE SHOT
5. READ FROM BOT	* "	*M8921 ROM*M8903 ACCL CNTR
6. READ START	* "	* " * "
7. READ SHUTDOWN	* "	* " * "
10. READ SETTLEDOWN	* "	*M8921 SETTLEDOWN ONE SHOT
11. READ REVERSE START	* "	*M8921 ROM*M8903 ACCL CNTR
12. READ REVERSE SHUTDOWN	* "	* " * "
13. READ REVERSE SETTLEDOWN	* "	*M8921 SETTLEDOWN ONE SHOT
14. TURN AROUND F-R	* "	*M8921 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	*FWD/REV SPEED	* " " "

TMO2 DRIVE FUNCTION TIMER

PAGE 14

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	* "	*M8921 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\*\*\*\*\*

TMO2 DRIVE FUNCTION TIMER

PAGE 15

7.3 TEST DESCRIPTIONS:

THE FIRST THIRTEEN (13) TESTS (T1 - T15) ARE CHECKS OF THE ROM CIRCUITS IN THE TU45 (M8921), THE ACCL COUNTER IN THE TMO2 (M8903), AND THE SETTLEDOWN ONE SHOT (M8921).

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:

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

TMO2 DRIVE FUNCTION TIMER

PAGE 16

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

TMO2 DRIVE FUNCTION TIMER

PAGE 17

T11. READ REVERSE START:

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

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

T12. READ REVERSE SHUTDOWN

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

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

T13. READ REVERSE SETTLEDOWN:

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

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

T14. TURN AROUND DELAY-FORWARD TO REVERSE

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

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

TMO2 DRIVE FUNCTION TIMER

PAGE 18

T15. TURN AROUND DELAY-REVERSE TO FORWARD

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

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

TMO2 DRIVE FUNCTION TIMER

PAGE 19

GAP MEASUREMENTS:

THE PREVIOUS THIRTEEN (13) TESTS WERE MEASUREMENTS OF LOGIC DELAYS PERFORMED BY THE TMO2 OR TU45 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.

CZTULAO TMO2 DRIVE FUNCTION TIMER  
CZTULA.P11 07-JUN-78 17:07

MACY11 30(1046) 13-JUN-78<sup>J 2</sup> 13:54 PAGE 22

SEQ 0022

7. STOP

TMO2 DRIVE FUNCTION TIMER

PAGE 20

T21. GAP CONSISTENCY:

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

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

\*\* (SEE GTIMTBL IN CZTULA 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

PAGE 21

T27. ERASE:

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

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

T30. TAPE MARK:

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

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

TMO2 DRIVE FUNCTION TIMER

PAGE 22

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.

```

.NLIST MC
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004

          .LIST ME,SEQ,LOC,BIN
          .ENABL ABS
          .TITLE CZTULAO TMO2 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 'FC' 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

          .SBTTL REGISTER ASSIGNMENTS
;;DEFINITIONS AND REGISTER ASSIGNMENTS
;;GENERAL REGISTER ASSIGNMENTS
          R0=X0
          R1=X1
          R2=X2
          R3=X3
          R4=X4
          R5=X5
          SP=X6
          PC=X7
          R10=X0
          R11=X1
          R12=X2
          R13=X3
          R14=X4
          R15=X5

;;REGISTER ADDRESSES
          PSW= 177776 ;;PROCESSER STATUS WORD
          SLR= 177774 ;;STACK LIMIT REGISTER (11/40,11/45)
  
```

1005	177772	PIRQ= 177772	::PROGRAM INTERRUPT REQ. (11/45)
1006	177770	UBREAK= 177770	::MICRO-BREAK REGISTER (11/45)
1007	177570	HSWR= 177570	::SWITCH REGISTER
1008	177570	DISPLAY=177570	::DISPLAY REGISTER (11/45)
1009	177560	TKS= 177560	::KEYBOARD CSR
1010	177562	TKB= 177562	::KEYBOARD DATA BUFFER REGISTER
1011	177564	TPS= 177564	::TELEPRINTER CSR
1012	177566	TPB= 177566	::TELEPRINTER DATA BUFFER REGISTER
1013			
1014		:::VECTOR ADDRESSES	
1015	000004	ERRVEC=4	:::ADDRESS OF ERROR VECTOR
1016	000010	RESVEC=10	:::ADDRESS OF RESERVED INST. TRAP VECTOR
1017	000014	TBITVEC=14	:::ADDRESS OF 'T' BIT TRAP VECTOR
1018	000014	TRTVEC=14	:::ADDRESS OF 'TRACE' TRAP VECTOR
1019	000014	BPTVEC=14	:::ADDRESS OF 'BREAKPOINT' TRAP VECTOR
1020	000020	IOTVEC=20	:::ADDRESS OF IOT TRAP VECTOR
1021	000024	PFVEC=24	:::ADDRESS OF POWER FAIL TRAP VECTOR
1022	000030	EMTVEC=30	:::ADDRESS OF EMT VECTOR
1023	000034	TRAPVEC=34	:::ADDRESS OF TRAP VECTOR
1024	000060	TKVEC= 60	:::ADDRESS OF TTY KEYBOARD INT. VECTOR
1025	000064	TPVEC=64	:::ADDRESS OF TTY PRINTER INTERRUPT VECTOR
1026	000114	PARVEC= 114	:::ADDRESS OF MA/MF PARITY ERROR VECTOR
1027	000240	PIRVEC=240	:::ADDRESS OF PIRQ VECTOR
1028	000244	FPEVEC=244	:::ADDRESS OF FLOATING POINT INT. VECTOR
1029	000250	MMVEC=250	:::ADDRESS OF MEM MGMT ERROR TRAP VECTOR
1030			
1031		:::CLOCK ADDRESS AND VECTORS	
1032	172540	PLKCSR= 172540	:::KW11-P
1033	000104	PLKVEC= 104	
1034	177546	KS= 177546	:::KW11-L
1035	000100	LKVEC= 100	
1036	177514	LPS= 177514	:::LP11
1037	177516	LPB= 177516	
1038			
1039		:::RH11, TM02/TU45 REGISTERS	
1040	172440	TMCS1= 172440	
1041			
1042		:::TM02/TU45 INDEX VALUES	
1043	000000	CS1= 00	:::CONTROL STATUS #1
1044	000002	WC= 02	
1045	000004	BA= 04	:::BUS ADDRESS REGISTER
1046	000006	FC= 06	:::FRAME COUNT
1047	000010	CS2= 10	:::CONTROL STATUS #2
1048	000012	DS= 12	:::DRIVE STATUS
1049	000014	ER= 14	:::ERROR REG #1
1050	000016	AS= 16	:::ATTENTION SUMMARY
1051	000022	DB= 22	:::DATA BUFFER REG
1052	000024	MR= 24	:::MAINTENANCE REG
1053	000026	DT= 26	:::DRIVE TYPE REG
1054	000030	SN= 30	:::SERIAL NUMBER REGISTER
1055	000032	TC= 32	:::TAPE CONTROL REG
1056			
1057		.SBTTL TM02/TU45 REGISTER BITS	
1058		:::RHCS1-CS1(R5)	
1059	000001	GO= 1	
1060	000000	NOP= 0	

1061	000002	RWDOFF=	2
1062	000006	RWD=	6
1063	000010	DRYCLR=	10
1064	000026	WFMK=	26
1065	000024	ERASE=	24
1066	000030	SPCFWD=	30
1067	000032	SPCREV=	32
1068	000050	WCHKF=	50
1069	000056	WCHKR=	56
1070	000060	WFWD=	60
1071	000070	RDFWD=	70
1072	000076	RDREV=	76
1073	000100	IE=	100
1074	000200	RDY=	200
1075	000400	A16=	400
1076	001000	A17=	1000
1077	002000	PSEL=	2000
1078	004000	DVA=	4000
1079	020000	MCPE=	20000
1080	040000	TRE=	40000
1081	100000	SC=	100000
1082		:RHCS2-CS2(R5)	
1083	000000	DV0=	0
1084	000001	DV1=	1
1085	000002	DV2=	2
1086	000003	DV3=	3
1087	000004	DV4=	4
1088	000005	DV5=	5
1089	000006	DV6=	6
1090	000007	DV7=	7
1091	000010	BAI=	10
1092	000020	PAT=	20
1093	000040	CLR=	40
1094	000100	IR=	100
1095	000200	OR=	200
1096	000400	MDPE=	400
1097	001000	MXF=	1000
1098	002000	PGE=	2000
1099	004000	NEM=	4000
1100	010000	NED=	10000
1101	020000	UPE=	20000
1102	040000	WCE=	40000
1103	100000	DLT=	100000
1104		:RHDS-DS(R5)	
1105	000001	SLA=	1
1106	000002	BOT=	2
1107	000004	TMK=	4
1108	000010	IDB=	10
1109	000020	SDWN=	20
1110	000040	PES=	40
1111	000100	SSC=	100
1112	000200	DRY=	200
1113	000400	DPR=	400
1114	002000	EOT=	2000
1115	004000	WRL=	4000
1116	010000	MOL=	10000

1117 020000  
1118 040000  
1119 100000  
1120  
1121 000001  
1122 000002  
1123 000004  
1124  
1125 000020  
1126 000100  
1127 000200  
1128 000400  
1129 001000  
1130 002000  
1131 004000  
1132 010000  
1133 020000  
1134 040000  
1135  
1136  
1137 000100  
1138  
1139  
1140 002000  
1141 010000  
1142 040000  
1143  
1144  
1145 000300  
1146 000320  
1147 000000  
1148 000400  
1149 001000  
1150 002000  
1151 100000  
1152  
1153  
1154 104400  
1155 104000  
1156 000004  
1157  
1158  
1159 005340  
1160 177400  
1161 177600  
1162  
1163 000003  
1164 000011  
1165 000012  
1166 000015  
1167 000017  
1168 000025  
1169 000000  
1170 000000 000000  
1171 000002 000000  
1172 000004 000006

PIP= 20000  
ERR= 40000  
ATA= 100000  
:RHER-ER(R5)  
ILF= 1  
ILR= 2  
RMR= 4  
  
FMT= 20  
INCVAE= 100  
PEFLRC= 200  
NSG= 400  
FCE= 1000  
CSITM= 2000  
NEF= 4000  
DTE= 10000  
OPI= 20000  
UNS= 40000  
  
:RHMR-MR(R5)  
OSC= 100  
  
:RHDT-DT(R5)  
SPR= 2000  
CH7= 10000  
TAP= 40000  
  
:RHTC-TC(R5)  
NORM11= 300  
CDM11= 320  
BPI200= 0  
BPI556= 000400  
BPI800= 001000  
PE1600= 002000  
ACCL= 100000  
  
:INSTRUCTION EQUATES  
HLT= TRAP  
SCOPE= EMT  
TYPE= IOT  
  
:MISCELLANEOUS EQUATES  
OUTBUF=INIT  
FRMCNT= -256.  
WRDCNT= -128.  
:ASCII EQUATES  
CNTRLC= 3  
HT= 11  
LF= 12  
CR= 15  
CNTRLO= 17  
CNTRLU= 25  
.=0  
0  
0  
.+2

:OUTPUT BUFFER START AT BEG OF PROGRAM  
:FRAME COUNT  
:WORD COUNT  
  
:ASCII CODE FOR CONTROL C ( C )  
:ASCII CODE FOR HORIZONTAL TAB  
:ASCII CODE FOR LINE FEED  
:ASCII CODE FOR CARRIAGE RETURN  
:ASCII CODE FOR CONTROL O ( O )  
:ASCII CODE FOR CONTROL U ( U )

1173	000006	000000	HALT
1174	000010	000012	.+2
1175	000012	000000	HALT
1176	000014	000016	.+2
1177	000016	000000	HALT
1178	000020	000022	.+2
1179	000022	000000	HALT
1180	000024	000026	.+2
1181	000026	000000	HALT
1182	000030	000032	.+2
1183	000032	000000	HALT
1184	000034	000036	.+2
1185	000036	000000	HALT
1186	000040	000042	.+2
1187	000042	000000	HALT
1188	000044	000046	.+2
1189	000046	000000	HALT
1190	000050	000052	.+2
1191	000052	000000	HALT
1192	000054	000056	.+2
1193	000056	000000	HALT
1194	000060	000062	.+2
1195	000062	000000	HALT
1196	000064	000066	.+2
1197	000066	000000	HALT
1198	000070	000072	.+2
1199	000072	000000	HALT
1200	000074	000076	.+2
1201	000076	000000	HALT
1202	000100	000102	.+2
1203	000102	000000	HALT
1204	000104	000106	.+2
1205	000106	000000	HALT
1206	000110	000112	.+2
1207	000112	000000	HALT
1208	000114	000116	.+2
1209	000116	000000	HALT
1210	000120	000122	.+2
1211	000122	000000	HALT
1212	000124	000126	.+2
1213	000126	000000	HALT
1214	000130	000132	.+2
1215	000132	000000	HALT
1216	000134	000136	.+2
1217	000136	000000	HALT
1218	000140	000142	.+2
1219	000142	000000	HALT
1220	000144	000146	.+2
1221	000146	000000	HALT
1222	000150	000152	.+2
1223	000152	000000	HALT
1224	000154	000156	.+2
1225	000156	000000	HALT
1226	000160	000162	.+2
1227	000162	000000	HALT
1228	000164	000166	.+2

1229	000166	000000	HALT
1230	000170	000172	.+2
1231	000172	000000	HALT
1232	000174	000176	.+2
1233	000176	000000	HALT
1234	000200	000202	.+2
1235	000202	000000	HALT
1236	000204	000206	.+2
1237	000206	000000	HALT
1238	000210	000212	.+2
1239	000212	000000	HALT
1240	000214	000216	.+2
1241	000216	000000	HALT
1242	000220	000222	.+2
1243	000222	000000	HALT
1244	000224	000226	.+2
1245	000226	000000	HALT
1246	000230	000232	.+2
1247	000232	000000	HALT
1248	000234	000236	.+2
1249	000236	000000	HALT
1250	000240	000242	.+2
1251	000242	000000	HALT
1252	000244	000246	.+2
1253	000246	000000	HALT
1254	000250	000252	.+2
1255	000252	000000	HALT
1256	000254	000256	.+2
1257	000256	000000	HALT
1258	000260	000262	.+2
1259	000262	000000	HALT
1260	000264	000266	.+2
1261	000266	000000	HALT
1262	000270	000272	.+2
1263	000272	000000	HALT
1264	000274	000276	.+2
1265	000276	000000	HALT
1266	000300	000302	.+2
1267	000302	000000	HALT
1268	000304	000306	.+2
1269	000306	000000	HALT
1270	000310	000312	.+2
1271	000312	000000	HALT
1272	000314	000316	.+2
1273	000316	000000	HALT
1274	000320	000322	.+2
1275	000322	000000	HALT
1276	000324	000326	.+2
1277	000326	000000	HALT
1278	000330	000332	.+2
1279	000332	000000	HALT
1280	000334	000336	.+2
1281	000336	000000	HALT
1282	000340	000342	.+2
1283	000342	000000	HALT
1284	000344	000346	.+2

1285	000346	000000	HALT
1286	000350	000352	.+2
1287	000352	000000	HALT
1288	000354	000356	.+2
1289	000356	000000	HALT
1290	000360	000362	.+2
1291	000362	000000	HALT
1292	000364	000366	.+2
1293	000366	000000	HALT
1294	000370	000372	.+2
1295	000372	000000	HALT
1296	000374	000376	.+2
1297	000376	000000	HALT
1298	000400	000402	.+2
1299	000402	000000	HALT
1300	000404	000406	.+2
1301	000406	000000	HALT
1302	000410	000412	.+2
1303	000412	000000	HALT
1304	000414	000416	.+2
1305	000416	000000	HALT
1306	000420	000422	.+2
1307	000422	000000	HALT
1308	000424	000426	.+2
1309	000426	000000	HALT
1310	000430	000432	.+2
1311	000432	000000	HALT
1312	000434	000436	.+2
1313	000436	000000	HALT
1314	000440	000442	.+2
1315	000442	000000	HALT
1316	000444	000446	.+2
1317	000446	000000	HALT
1318	000450	000452	.+2
1319	000452	000000	HALT
1320	000454	000456	.+2
1321	000456	000000	HALT
1322	000460	000462	.+2
1323	000462	000000	HALT
1324	000464	000466	.+2
1325	000466	000000	HALT
1326	000470	000472	.+2
1327	000472	000000	HALT
1328	000474	000476	.+2
1329	000476	000000	HALT
1330	000500	000502	.+2
1331	000502	000000	HALT
1332	000504	000506	.+2
1333	000506	000000	HALT
1334	000510	000512	.+2
1335	000512	000000	HALT
1336	000514	000516	.+2
1337	000516	000000	HALT
1338	000520	000522	.+2
1339	000522	000000	HALT
1340	000524	000526	.+2

1341	000526	000000	HALT
1342	000530	000532	.+2
1343	000532	000000	HALT
1344	000534	000536	.+2
1345	000536	000000	HALT
1346	000540	000542	.+2
1347	000542	000000	HALT
1348	000544	000546	.+2
1349	000546	000000	HALT
1350	000550	000552	.+2
1351	000552	000000	HALT
1352	000554	000556	.+2
1353	000556	000000	HALT
1354	000560	000562	.+2
1355	000562	000000	HALT
1356	000564	000566	.+2
1357	000566	000000	HALT
1358	000570	000572	.+2
1359	000572	000000	HALT
1360	000574	000576	.+2
1361	000576	000000	HALT
1362	000600	000602	.+2
1363	000602	000000	HALT
1364	000604	000606	.+2
1365	000606	000000	HALT
1366	000610	000612	.+2
1367	000612	000000	HALT
1368	000614	000616	.+2
1369	000616	000000	HALT
1370	000620	000622	.+2
1371	000622	000000	HALT
1372	000624	000626	.+2
1373	000626	000000	HALT
1374	000630	000632	.+2
1375	000632	000000	HALT
1376	000634	000636	.+2
1377	000636	000000	HALT
1378	000640	000642	.+2
1379	000642	000000	HALT
1380	000644	000646	.+2
1381	000646	000000	HALT
1382	000650	000652	.+2
1383	000652	000000	HALT
1384	000654	000656	.+2
1385	000656	000000	HALT
1386	000660	000662	.+2
1387	000662	000000	HALT
1388	000664	000666	.+2
1389	000666	000000	HALT
1390	000670	000672	.+2
1391	000672	000000	HALT
1392	000674	000676	.+2
1393	000676	000000	HALT
1394	000700	000702	.+2
1395	000702	000000	HALT
1396	000704	000706	.+2

```

1397 000706 000000 HALT
1398 000710 000712 .+2
1399 000712 000000 HALT
1400 000714 000716 .+2
1401 000716 000000 HALT
1402 000720 000722 .+2
1403 000722 000000 HALT
1404 000724 000726 .+2
1405 000726 000000 HALT
1406 000730 000732 .+2
1407 000732 000000 HALT
1408 000734 000736 .+2
1409 000736 000000 HALT
1410 000740 000742 .+2
1411 000742 000000 HALT
1412 000744 000746 .+2
1413 000746 000000 HALT
1414 000750 000752 .+2
1415 000752 000000 HALT
1416 000754 000756 .+2
1417 000756 000000 HALT
1418 000760 000762 .+2
1419 000762 000000 HALT
1420
1421 ;SETUP TRAP VECTORS
1422 000014 000014 .=TBITVEC
1423 000016 000000 .WORD .+2 ;SET 'T' TRAP TO TIMER ROUTINE
1424 000020 001776 .WORD HALT ;PRIORITY LEVEL 7
1425 000022 000000 .WORD .TYPE ;SET IOT TRAP TO .TYPE ROUTINE
1426 000024 000026 .WORD 0 ;PRIORITY LEVEL 0
1427 000026 000000 .WORD PFVEC+2 ;POWER FAIL TRAP TO HALT
1428 000030 003552 .WORD HALT ;AT PFVEC+2
1429 000032 000340 .WORD .SCOPE ;SET EMT TRAP TO .SCOPE ROUTINE
1430 000034 003304 .WORD 340 ;PRIORITY LEVEL 7
1431 000036 000340 .WORD .HLT ;SET TRAP TRAP TO .HLT ROUTINE
1432 000060 000060 .=TKVEC ;PRIORITY LEVEL 7
1433 000062 003240 .WORD TKISR
1434 000064 000340 .WORD 340
1435 000200 000200 .=200
1436 000200 000137 005340 JMP @#INIT ;GO TO START OF PROGRAM
1437
1438 .=500
1439 STKPTR= 600 ;STACK
1440
1441 .=1000
1442
1443 ;PROGRAM TAGS
1444
1445 001000 177570 SWR: .WORD HSWR
1446 001002 000000 SCPADR: .WORD 0
1447 001004 000 DRVNUM: .BYTE 0 ;TM02 DRIVE UNDER TEST
1448 001005 000 SLVNUM: .BYTE 0 ;TU45 SLAVE UNDER TEST
1449 001006 000000 SLVPTR: .WORD 0 ;POINTER TO SLAVE TABLE (SLVTBL) BELOW
1450 001010 172440 TMBASE: .WORD TMCS1 ;BASE ADDRESS OF TM02/TU45 REGISTERS
1451 001012 000000 ATIME: .WORD 0 ;CONTAINS 'TICK' COUNT
1452 001014 000020 ATIMTBL: .BLKW 16. ;EACH ENTRY CONTAINS TIME FOR FUNCTION

```

1453			
1454	001054	000020	
1455	001114	000000	
1456	001116	000000	
1457	001120	000	
1458	001121	000	
1459	001122	000	
1460	001123	000	
1461	001124	000	
1462	001125	000	
1463	001126	000	
1464	001127	000	
1465	001130	000	
1466		001132	
1467	001132	030460	
1468	001134	031462	
1469	001136	032464	
1470	001140	033466	
1471	001142	034470	
1472	001144	000006	
1473	001152	000	
1474		001154	
1475	001154	000010	
1476	001164	000100	
1477	001264	000110	
1478	001374	005015	000
1479	001377	134	000
1480	001401	060	000
1481	001403	007	000
1482	001405	055	000
1483	001407	040	
1484	001410	000040	
1485	001412	004476	000
1486		001416	

GAPTBL:	.BLKW	16.
DELTIM:	.WORD	0
OCTALO:	.WORD	0
GAP:	.BYTE	0
ITCNT:	.BYTE	0
TSTNUM:	.BYTE	0
ERFLG:	.BYTE	0
PRGFLG:	.BYTE	0
UNTFND:	.BYTE	0
TYPFLG:	.BYTE	0
NRZFLG:	.BYTE	0
ASFLG:	.BYTE	0
	.EVEN	
DIGTAB:	'01	
	'23	
	'45	
	'67	
	'89	
ODIGITS:	.BLKB	6
	.BYTE	0
	.EVEN	
DRVTBL:	.BLKB	8.
SLVTBL:	.BLKB	64.
INBUF:	.BLKB	72.
CRLF:	.ASCIZ	<CR><LF>
BKSLSH:	.ASCIZ	' '
ECHO:	.ASCIZ	'0'
BELL:	.ASCIZ	<7>
DASH:	.ASCIZ	'-'
SPACE2:	.ASCII	' '
SPACE:	.ASCIZ	' '
ANGTAB:	.ASCIZ	'>'<HT>
	.EVEN	

```

; ENTRIES ARE MADE BY 'SCOPE' ROUTINE
; TIMES RECORDED BY 'GAP CONSISTANCY' TEST
; VARIABLE DELAY

; CONTAINS GAP # (USED FOR TST 021)
; ITERATION COUNT
; TEST #
; ERROR FLAG
; PROGRAM FLAG
; UNIT FOUND INDICATOR

; INDICATES IF DRIVE IS NRZ ONLY.
; 1/0 = YES/NO.

; RESERVE SPACE FOR CONVERTED DIGITS
; TERMINATOR

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

1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494 001416 000000 000000  
1495 001422 017500 015220  
1496 001426 001046 000702  
1497 001432 000702 000550  
1498 001436 001130 000714  
1499 001442 001666 001426  
1500 001446 000264 000226  
1501 001452 000252 000202  
1502 001456 001130 000714  
1503 001462 000264 000226  
1504 001466 000346 000276  
1505 001472 001130 000714  
1506 001476 001440 001130  
1507 001502 001440 001130  
1508 001506 001440 001130  
1509 001512 001440 001130  
1510 001516 001440 001130  
1511 001522 001440 001250  
1512 001526 000000 000000  
1513 001532 002660 002533  
1514 001536 002660 002533  
1515 001542 002660 002533  
1516 001546 002734 002424  
1517 001552 013724 012574  
1518 001556 014400 013250  
1519 001562 002544 002424  
1520 001566 002544 002424  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530 001572 001452 001274  
1531 001576 001564 001406  
1532 001602 001556 001376  
1533 001606 001452 001274  
1534 001612 001426 001250  
1535 001616 001426 001250  
1536 001622 001426 001250  
1537 001626 001426 001250  
1538 001632 001426 001250  
1539 001636 001426 001250  
1540 001642 001426 001250  
1541 001646 001426 001250  
1542 001652 001426 001250

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

: .WORD	MAX,MIN	: TIME IN MS	FUNCTION	TEST #
STIMTBL: .WORD	0,0	: SPARE		
.WORD	8000.,6800.	: 80.0-68.0	WRITE FROM BOT	TST001
.WORD	00550.,00450.	: 5.50-4.50	WRITE START	TST002
.WORD	00450.,00360.	: 4.50-3.90	WRITE SHUTDOWN	TST003
.WORD	00600.,00460.	: 6.0-4.60	WRITE STLDOWN	TST004
.WORD	0950.,0790.	: 9.5-7.9	READ FROM BOT	TST005
.WORD	00180.,00150.	: 1.80-1.50	READ START	TST006
.WORD	00170.,00130.	: 1.70-1.30	READ SHUTDOWN	TST007
.WORD	00600.,00460.	: 6.00-4.60	READ SETTLEDOWN	TST010
.WORD	00180.,00150.	: 1.80-1.50	RD REV START	TST011
.WORD	00230.,00190.	: 2.30-1.90	RD REV SHTDWN	TST012
.WORD	00600.,00460.	: 6.00-4.60	RD REV STLDWN	TST013
.WORD	00800.,00600.	: 8.00-6.00	TRN RND DLY F-R	TST014
.WORD	00800.,00600.	: 8.00-6.00	TRN RND DLY R-F	TST015
.WORD	00800.,00600.	: 8.00-6.00	GAP SIZE STOP	TST016
.WORD	00800.,00600.	: 8.00-6.00	GAP SIZE STRT	TST017
.WORD	00800.,00600.	: 8.00-6.00	GAP SIZE INTER	TST020
.WORD	00800.,00680.	: 8.00-6.80	GAP CONSISANCY	TST021
.WORD	0,0	: 0.0-0.0	DUMMY	TST022
.WORD	01456.,01371.	: 14.56-13.71	DAT TIME 200BPI	TST023
.WORD	01456.,01371.	: 14.56-13.71	DAT TIME 556BPI	TST024
.WORD	01456.,01371.	: 14.56-13.71	DAT TIME 800BPI	TST025
.WORD	01500.,01300.	: 15.00-13.00	DAT TIME 1600PE	TST026
.WORD	06100.,05500.	: 61.00-55.00	ERASE	TST027
.WORD	06400.,05800.	: 64.00-58.00	WRT FILE MARK	TST030
.WORD	01380.,01300.	: 13.80-13.00	READ 1" TAPE	TST031
.WORD	01380.,01300.	: 13.80-13.00	RD REV 1" TAPE	TST032

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

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

: .WORD	MAX,MIN(10)	: TIME IN MS(10)	GAP #	DELAY IN MS(10)
GTIMTBL: .WORD	00810.,00700.	: 8.10-7.00	GAP-0	0 MS
.WORD	00884.,00774.	: 8.84-7.74	GAP-1	1.0 MS
.WORD	00878.,00766.	: 8.78-7.66	GAP-2	2.0 MS
.WORD	00810.,00700.	: 8.10-7.00	GAP-3	3.0 MS
.WORD	00790.,00680.	: 7.90-6.80	GAP-4	4.0 MS
.WORD	00790.,00680.	: 7.90-6.80	GAP-5	5.0 MS
.WORD	00790.,00680.	: 7.90-6.80	GAP-6	6.0 MS
.WORD	00790.,00680.	: 7.90-6.80	GAP-7	7.0 MS
.WORD	00790.,00680.	: 7.90-6.80	GAP-10	8.0 MS
.WORD	00790.,00680.	: 7.90-6.80	GAP-11	9.0 MS
.WORD	00790.,00680.	: 7.90-6.80	GAP-12	10.0 MS
.WORD	00790.,00680.	: 7.90-6.80	GAP-13	11.0 MS
.WORD	00790.,00680.	: 7.90-6.80	GAP-14	12.0 MS

CZTULAO TMO2 DRIVE FUNCTION TIMER  
CZTULA.P11 07-JUN-78 17:07

L 3  
MACY11 30(1046) 13-JUN-78 13:54 PAGE 37  
GAP TIME SPECIFICATION TABLE

SEQ 0037

1543	001656	001426	001250	.WORD	00790.,00680.	:7.90-6.80	GAP-15	13.1 MS
1544	001662	001426	001250	.WORD	00790.,00680.	:7.90-6.80	GAP-16	14.1 MS
1545	001666	001426	001250	.WORD	00790.,00680.	:7.90-6.80	GAP-17	15.1 MS

1546  
 1547  
 1548 001672 000000  
 1549 001674 014215  
 1550 001676 014237  
 1551 001700 014257  
 1552 00 702 014301  
 1553 001704 014325  
 1554 00 706 014347  
 1555 001710 014366  
 1556 001712 014410  
 1557 001714 014433  
 1558 001716 014455  
 1559 001720 014502  
 1560 001722 014531  
 1561 001724 014562  
 1562 001726 014613  
 1563 001730 014641  
 1564 001732 014670  
 1565 001734 014720  
 1566 001736 000000  
 1567 001740 014743  
 1568 001742 014767  
 1569 001744 015013  
 1570 001746 015037  
 1571 001750 015064  
 1572 001752 015106  
 1573 001754 015131  
 1574 001756 015153  
 1575  
 1576  
 1577  
 1578  
 1579  
 1580  
 1581  
 1582  
 1583  
 1584 000011  
 1585 001760 000  
 1586 001761 002  
 1587 001762 000  
 1588  
 1589 001763 000  
 1590 001764 177564  
 1591 001766 177566  
 1592 001770 000  
 1593 001771 000  
 1594 001772 005015 000  
 1595 001776  
 1596  
 1597 001776 010046  
 1598 002000 017600 000002  
 1599 002004 062766 000002 000002  
 1600 002012 105067 177753  
 1601

```

.SBTTL TEST HEADER POINTERS
;THE BELOW TABLE CONTAINS POINTERS TO EACH TEST'S DESCRIPTOR
NAMPTR: .WORD 0
        .WORD A.T001
        .WORD A.T002
        .WORD A.T003
        .WORD A.T004
        .WORD A.T005
        .WORD A.T006
        .WORD A.T007
        .WORD A.T010
        .WORD A.T011
        .WORD A.T012
        .WORD A.T013
        .WORD A.T014
        .WORD A.T015
        .WORD A.T016
        .WORD A.T017
        .WORD A.T020
        .WORD A.T021
        .WORD 0 ;DUMMY TEST
        .WORD A.T023
        .WORD A.T024
        .WORD A.T025
        .WORD A.T026
        .WORD A.T027
        .WORD A.T030
        .WORD A.T031
        .WORD A.T032

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

;;TAGS USED BY THE TYPE ROUTINE BELOW
$HT=11 ;HORIZONTAL TAB
$NULL: .BYTE 0 ;CONTAINS NULL CHARACTER
$FILL: .BYTE 2 ;CONTAINS # OF FILLER CHARACTERS
$TPFLG: .BYTE 0 ;CONTAINS TELEPRINTER AVAILABLE FLAG
;0/377 = AVAIL/NOT AVAIL
$TKFLG: .BYTE 0 ;CONTAINS KEYBOARD AVAILABLE FLAG
$TPS: .WORD 177564 ;ADDRESS OF TELEPRINTER STATUS REGISTER
$TPB: .WORD 177566 ;ADDRESS OF TELEPRINTER DATA BUFFER
$CHARCNT: .BYTE 0 ;CONTAINS # OF CHARS TYPED
$CNTRLO: .BYTE 0 ;CONTAINS CONTROL 0 CHAR (IF TYPED)
$CRLF: .ASCII <15><12>
        .EVEN

.TYPE: MOV RC, -(SP) ;SAVE R0
        MOV @2(SP), R0 ;GET MESSAGE ADDRESS
        ADD #2, 2(SP) ;ADJUST RETURN PC
        CLRB $CNTRLO
  
```

```

1602 002016 105767 177747 1$: TSTB $CNTRLO ;;BRANCH IF CONTROL O( O) WASN'T TYPED
1603 002022 001403 BEQ 12$
1604 002024 000004 001772 TYPE,$CRLF ;;TYPE <CR><LF>
1605 002030 000403 BR 11$
1606 002032 112046 12$: MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1607 002034 001003 BNE 2$ ;;BRANCH IF NOT THE TERMINATOR
1608 002036 005726 TST (SP)+ ;;POP TERMINATOR CHAR OFF THE STACK
1609 002040 012600 11$: MOV (SP)+,RO ;;RESTORE RO
1610 002042 000002 RTI ;;RETURN TO CALLER
1611
1612 002044 122716 000011 2$: CMPB #$HT,(SP) ;;BRANCH IF HORIZONTAL TAB <HT>
1613 002050 001445 BEQ 9$
1614 002052 004767 000026 JSR PC,5$ ;;TYPE CHARACTER
1615 002056 122726 000012 3$: CMPB #12,(SP)+ ;;CHECK IF CHARACTER WAS A LINE FEED
1616 002062 001355 BNE 1$ ;;BRANCH IF NOT LINE FEED
1617 002064 016746 177670 MOV $NULL,-(SP) ;;GET # OF FILLERS REQUIRED AND FILLER
1618 ;;CHARACTER.
1619
1620 002070 105366 000001 4$: DECB 1(SP) ;;DECREMENT FILLERS REQ. COUNT
1621 002074 002770 BLT 3$ ;;BRANCH IF NO MORE FILLERS ARE REQUIRED
1622 002076 004767 000002 JSR PC,5$ ;;TYPE FILLER CHARACTER
1623 002102 000772 BR 4$
1624
1625 002104 105777 177654 5$: TSTB @STPS ;;WAIT FOR OUTPUT DEVICE
1626 002110 100375 BPL -4
1627 002112 122737 000017 001771 CMPB #17,@$CNTRLO ;;CHECK IF CONTROL O WAS TYPED
1628 002120 001403 BEQ 6$ ;;STOP TYPING MESSAGE IF O WAS TYPED
1629 002122 116677 000002 177636 MOV 2(SP),@STPB ;;OUTPUT CHARACTER
1630 002130 122766 000015 000002 6$: CMPB #15,2(SP) ;;BRANCH IF NOT <CR>
1631 002136 001003 BNE 7$
1632 002140 105067 177624 CLRB $CHARCNT ;;CLEAR CHARACTERS TYPED COUNT
1633 002144 000406 BR 8$
1634 002146 122766 000012 000002 7$: CMPB #12,2(SP) ;;BRANCH IF <LF> OR 'NULL'
1635 002154 002002 BGE 8$
1636 002156 105267 177606 INCB $CHARCNT ;;INCREMENT CHARACTER TYPED COUNT
1637 002162 000207 8$: RTS PC
1638
1639 ;;HORIZONTAL TAB <HT> PROCESSER
1640 002164 112716 000040 9$: MOVB #40,(SP) ;;LOAD 'SPACE'
1641 002170 004767 177710 10$: JSR PC,5$ ;;TYPE 'SPACE'
1642 002174 132767 000007 177566 BITB #7,$CHARCNT ;;TYPE SPACES UNTIL A MULTIPLE
1643 002202 001372 BNE 10$ ;;OF 8 CHARACTERS HAVE BEEN TYPED
1644 002204 105726 TSTB (SP)+ ;;POP SPACE
1645 002206 000703 BR 1$ ;;GET NEXT CHARACTER
1646
1647 ;;SUBROUTINE TO SAVE GENERAL REGISTERS ON THE STACK
1648 :CALL: SAVE
1649 002210 010546 .SAVE: MOV R5,-(SP) ;;SAVE REGISTERS ON THE STACK
1650 002212 010446 MOV R4,-(SP)
1651 002214 010346 MOV R3,-(SP)
1652 002216 010246 MOV R2,-(SP)
1653 002220 010146 MOV R1,-(SP)
1654 002222 010046 MOV R0,-(SP)
1655 002224 016646 000014 MOV 14(SP),-(SP) ;;GET RETURN PC
1656 002230 000207 RTS PC ;;RETURN
1657

```

```

1658 ;SUBROUTINE TO RESTORE GENERAL REGISTERS FROM THE STACK
1659 ;CALL: RESTORE
1660 002232 012666 000014 .RESTORE:MOV (SP)+,14(SP) ;MOVE RETURN PC
1661 002236 012600 MOV (SP)+,R0 ;RESTORE REGISTERS
1662 002240 012601 MOV (SP)+,R1
1663 002242 012602 MOV (SP)+,R2
1664 002244 012603 MOV (SP)+,R3
1665 002246 012604 MOV (SP)+,R4
1666 002250 012605 MOV (SP)+,R5
1667 002252 000207 RTS PC ;RETURN
1668
1669 ;SUBROUTINE TO CONVERT OCTAL DATA TO ASCII
1670 ;CALL: MOV NUMBER,R2 ;MOVE NUMBER TO R2
1671 ; JSR PC,CNVOCT
1672
1673 002254 110667 176646 CNVOCT:MOVB SP,TYPFLG ;SET DO NOT TYPE FLAG
1674 002260 000402 BR CNVTO
1675
1676 .SBTTL OCTAL TO ASCII & TYPE ROUTINE
1677 ;SUBROUTINE TO CONVERT OCTAL NUMBER TO ASCII AND TYPE IT OUT
1678 ;CALL: MOV NUMBER,R2 ;PUT # IN R2
1679 ; JSR PC,TYPOCT ;CALL ROUTINE
1680
1681 002262 105037 001126 TYPOCT:CLRB @#TYPFLG ;SET TYPE FLAG
1682 002266 CNVTO:
1683 002266 004767 177716 JSR PC,.SAVE ;SAVE REGISTERS ON THE STACK
1684 002272 012704 001144 MOV #ODIGITS,R4 ;SET PTR TO OUTPUT
1685 002276 005003 CLR R3 ;R3 WILL CONTAIN OCTAL DIGIT
1686 002300 010201 MOV R2,R1 ;GET # TO BE TYPED
1687 002302 006302 1$:ASL R2 ;SHIFT #
1688 002304 006103 ROL R3
1689 002306 012700 000006 MOV #6,R0 ;SET DIGIT COUNTER
1690 002312 000404 BR 3$
1691
1692 002314 006302 2$:ASL R2 ;SHIFT # 3 PLACES LEFT
1693 002316 006103 ROL R3
1694 002320 005301 DEC R1
1695 002322 001374 BNE 2$
1696 002324 012701 000003 3$:MOV #3,R1 ;SET SHIFT COUNTER
1697 002330 116324 001132 MOVB DIGTAB(R3),(R4)+ ;MOVE ASCII EQUIV TO OUTPUT
1698 002334 005003 CLR R3
1699 002336 005300 DEC R0 ;DECREMENT DIGIT COUNT
1700 002340 001365 BNE 2$ ;GET NEXT DIGIT
1701 002342 105737 001126 TSTB @#TYPFLG ;BRANCH IF ASCII IS
1702 002346 001002 BNE 4$ ;NOT TO BE TYPED
1703 002350 000004 001144 TYPE,ODIGITS
1704 002354 4$:
1705 002354 004767 177652 JSR PC,.RESTORE ;RESTORE REGISTERS FROM THE STACK
1706 002360 000207 RTS PC
1707
1708
1709 ;SUBROUTINE TO CONVERT OCTAL DATA TO DECIMAL ASCII
1710 ;CALL: MOV NUMBER,R2 ;MOVE NUMBER TO R2
1711 ; JSR PC,CNVDEC
1712
1713 002362 110637 001126 CNVDEC:MOVB SP,@#TYPFLG ;SET DO NOT TYPE FLAG

```

```

1714 002366 000402          BR      CNVTD
1715                      .SBTTL      OCTAL TO DECIMAL & TYPE ROUTINE
1716                      ;THIS ROUTINE CONVERTS AN OCTAL # TO DECIMAL ASCII AND TYPES IT OUT
1717                      ;CALL: MOV      NUMBER,R2          ;PUT # IN R2
1718                      ;      JSR      PC,TYPDEC          ;CALL ROUTINE
1719
1720 002370 105037 001126  TYPDEC: CLRB      @#TYPFLG          ;SET TYPE FLAG
1721 002374                      CNVTD:
1722 002374 004767 177610          JSR      PC,.SAVE          ;SAVE REGISTERS ON THE STACK
1723 002400 005000          CLR      R0          ;R0 IS INDEX TO DECIMAL CONSTANT
1724 002402 012704 001144          MOV      #ODIGITS,R4      ;SET OUTPUT PTR
1725 002406 005003 1$: CLR      R3          ;R3 CONTAINS DECIMAL DIGIT
1726 002410 166002 002470 2$: SUB      DCONST(R0),R2      ;SUBTRACT DECIMAL CONSTANT UNTIL
1727 002414 103402          BLO      3$          ;INPUT # GOES NEGATIVE
1728 002416 005203          INC      R3          ;KEEPING TRACK OF SUBTRACTIONS
1729 002420 000773          BR      2$
1730 002422 066002 002470 3$: ADD      DCONST(R0),R2      ;ADD BACK CONSTANT WHEN NEGATIVE
1731 002426 116324 001132          MOVB     DIGTAB(R3),(R4)+  ;MOVE ASCII EQUIVALENT
1732 002432 062700 000002          ADD      #2,R0          ;NEXT CONSTANT
1733 002436 005760 002470          TST      DCONST(R0)      ;UNTIL ALL CONSTANTS DONE
1734 002442 001361          BNE      1$
1735 002444 112724 000060          MOVB     #'0,(R4)+      ;LAST DIGIT IS 0
1736 002450 105737 001126          TSTB     @#TYPFLG        ;BRANCH IF ASCII IS
1737 002454 001002          BNE      4$          ;NOT TO BE TYPED
1738 002456 000004 001144          TYPE,ODIGITS
1739 002462 4$:
1740 002462 004767 177544          JSR      PC,.RESTORE      ;RESTORE REGISTERS FROM THE STACK
1741 002466 000207          RTS      PC
1742
1743 002470 023420  DCONST: .WORD 10000.
1744 002472 001750          .WORD 1000.
1745 002474 000144          .WORD 100.
1746 002476 000012          .WORD 10.
1747 002500 000001          .WORD 1.
1748 002502 000000          .WORD 0          ;TERMINATOR
1749
1750                      .SBTTL      TYPE SPECIFIED TIMES ROUTINE
1751                      ;THIS SUBROUTINE OUTPUTS THE TIME SPECIFICATIONS FOR THE TEST
1752                      ;AND ALSO THE ACTUAL TIME RECORDED (ATIME)
1753                      ;FORMAT OF LINE TYPED
1754                      ;RANGE=<AAAAAA-BBBBBB>          ACTUAL=CCCCC
1755                      ;WHERE:          AAAAAA IS MAXIMUM TIME FOR TEST (STIMTBL(TSTNUMX4)).
1756                      ;          BBBBBB IS MINIMUM TIME FOR TEST (STIMTBL(TSTNUMX4+2)).
1757                      ;          CCCCC IS ACTUAL TIME RECORDED BY TEST (ATIME).
1758                      ;CALL: MOVB     TEST NUMBER,R2      ;LOAD TEST NUMBER
1759                      ;      MOV      #TIME,@#ATIME        ;MOVE TIME TO ATIME
1760                      ;      JSR      PC,OUTSPC
1761 002504 010246  OUTSPC: MOV      R2,-(SP)          ;SAVE R2 & R3 ON THE STACK
1762 002506 010346          MOV      R3,-(SP)
1763 002510 006302          ASL      R2          ;MULTIPLY TEST # TIMES 4
1764 002512 006302          ASL      R2          ;TO FORM INDEX INTO STIMTBL
1765 002514 010203          MOV      R2,R3          ;R3 CONTAINS INDEX INTO TABLE
1766 002516 000004 014175          TYPE,L.RNG
1767 002522 016302 001416          MOV      STIMTBL(R3),R2      ;GET MAXIMUM SPEC TIME
1768 002526 004767 177636          JSR      PC,TYPDEC        ;CONVERT TO DECIMAL & TYPE
1769 002532 000004 001405          TYPE,DASH

```

```

1770 002536 016302 001420
1771 002542 004767 177622
1772 002546 000004 001412
1773 002552 000004 014205
1774 002556 013702 001012
1775 002562 004767 177602
1776 002566 000004 001374
1777 002572 012603
1778 002574 012602
1779 002576 000207
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789 002600 010246
1790 002602 010346
1791 002604 116703 176310
1792 002610 006303
1793 002612 006303
1794 002614 000004 014175
1795 002620 016302 001572
1796 002624 004767 177540
1797 002630 000004 001405
1798 002634 016302 001574
1799 002640 004767 177524
1800 002644 000004 001412
1801 002650 000004 014205
1802 002654 013702 001012
1803 002660 004767 177504
1804 002664 000004 013664
1805 002670 113702 001120
1806 002674 004767 177362
1807 002700 000004 001374
1808 002704 012603
1809 002706 012602
1810 002710 000207
1811
1812
1813
1814
1815 002712
1816 002712 004767 177272
1817 002716 012700 001264
1818 002722 012701 001116
1819 002726 005011
1820 002730 005061 000002
1821 002734 122710 000015
1822 002740 001414
1823 002742 112002
1824 002744 042702 177770
1825 002750 012703 000003

```

```

MOV STIMTBL+2(R3),R2 ;GET MINIMUM TIME
JSR PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
TYPE,ANGTAB
TYPE,L.ACT
MOV @#ATIME,R2 ;GET ACTUAL TIME
JSR PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
TYPE,CRLF
MOV (SP)+,R3
MOV (SP)+,R2
RTS PC ;RETURN

.SBTTL TYPE GAP TIMES SUBROUTINE
;THIS SUBROUTINE IS USED TO TYPE THE SPECIFIED GAP SIZES (RECORDED IN
;TST021). IT IS CALLED BY THE GAPOK ROUTINE IF THE GAP SIZE IS OUT OF
;RANGE VIA THE HLT ROUTINE (HLT+2).
;CALL: MOV #GAP,GAP ;LOAD GAP # INTO GAP
; MOV #TIME,ATIME ;LOAD ACTUAL TIME INTO ATIME
; JSR PC,OUTGAP

OUTGAP: MOV R2,-(SP) ;SAVE R2 AND R3
MOV R3,-(SP)
MOVB GAP,R3 ;GET GAP #
ASL R3
ASL R3
TYPE,L.RNG
MOV GTIMTBL(R3),R2 ;GET MAX TIME
JSR PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
TYPE,DASH
MOV GTIMTBL+2(R3),R2 ;GET MIN TIME
JSR PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
TYPE,ANGTAB ;TYPE <
TYPE,L.ACT
MOV @#ATIME,R2 ;GET ACTUAL TIME
JSR PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
TYPE,E.GAP
MOVB @#GAP,R2 ;GET GAP #
JSR PC,TYPDEC ;TYPE GAP #
TYPE,CRLF
MOV (SP)+,R3 ;RESTORE R3 AND R2
MOV (SP)+,R2
RTS PC

.SBTTL ASCII TO OCTAL CONVERT SUBROUTINE
;SUBROUTINE TO CONVERT ASCII DATA TO OCTAL. CONVERTED OCTAL DATA
;IS LEFT IN OCTALO <15-00>.
CNVTAO: JSR PC,.SAVE ;SAVE REGISTERS ON THE STACK
MOV #INBUF,R0 ;SET PTR TO ASCII DATA
MOV #OCTALO,R1 ;GET ADDRESS OF OCTAL DATA
CLR (R1) ;CLEAR OUT OLD OCTAL DATA
CLR 2(R1)
1$: CMPB #CR,(R0) ;<CR> TERMINATES INPUT
BEQ 3$
MOVB (R0)+,R2 ;GET 'OCTAL' DATA
BIC #177770,R2 ;STRIP UNUSED BITS
MOV #3,R3 ;SET SHIFT COUNT

```

1826 002754 006311  
1827 002756 006161 000002  
1828 002762 005303  
1829 002764 001373  
1830 002766 050211  
1831 002770 000761  
1832 002772  
1833 002772 004767 177234  
1834 002776 000207  
1835  
1836

```
2$: ASL (R1) ;SHIFT LAST
    ROL 2(R1) ;OCTAL DIGIT
    DEC R3
    BNE 2$
    BIS R2,(R1) ;AND INSERT THIS DIGIT
    BR 1$ ;GO GET NEXT DIGIT

3$: JSR PC,.RESTORE ;RESTORE REGISTERS FROM THE STACK
    RTS PC ;RETURN
```

1837  
1838  
1839  
1840  
1841

```
.SBTTL PUBLISH SUBROUTINE
;THE PUBLISH SUBROUTINE AVERAGES THE RECORDED TIMES FOR EACH TEST IT-
;ERATION (IF 16. ITERATIONS) AND PLACES THE AVERAGE RESULT IN 'ATIME'.
;IT TYPES THE NAME OF THE FUNCTION THAT WAS TIMED,THE TIME SPEC-
;IFICATION AND THE ACTUAL TIME .
```

1842 003000  
1843 003000 004767 177204  
1844 003004 012700 001014  
1845 003010 113701 001121  
1846 003014 122701 000001  
1847 003020 001423  
1848 003022 005002  
1849 003024 005003  
1850 003026 122701 000004  
1851 003032 001402  
1852 003034 000000  
1853 003036 000777  
1854  
1855

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

1856 003040 062002  
1857 003042 005503  
1858 003044 005301  
1859 003046 001374  
1860

```
1$: ADD (R0)+,R2 ;SUM INDIVIDUAL TIMES
    ADC R3
    DEC R1
    BNE 1$
```

1861 003050 012700 000002  
1862 003054 006203  
1863 003056 006002  
1864 003060 005300  
1865 003062 001374  
1866 003064 010237 001012  
1867

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

1868 003070 113700 001122  
1869 003074 006300  
1870 003076 016067 001672 000002  
1871 003104 000004  
1872 003106 000000

```
4$: MOVB @#TSTNUM,R0 ;GET TEST #
    ASL R0
    MOV NAMPTR(R0),5$ ;GET TEST NAME STRING ADDRESS
    TYPE
```

1873 003110 113702 001122  
1874 003114 004767 177364  
1875 003120 004767 177106  
1876 003124 000207  
1877

```
5: .WORD 0
    MOVB @#TSTNUM,R2 ;GET TEST #
    JSR PC,OUTSPC ;OUTPUT TIMES
    JSR PC,.RESTORE ;RESTORE REGISTERS FROM THE STACK
    RTS PC
```

1878  
1879  
1880  
1881

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

1882  
1883 003126 010046  
1884 003130 012700 001264  
1885 003134 105737 177560  
1886 003140 100375  
1887  
1888 003142 113746 177562  
1889 003146 042716 000200  
1890 003152 122716 000177  
1891 003156 001004  
1892 003160 124026  
1893 003162 000004 001377  
1894 003166 000762  
1895 003170 122716 000025  
1896 003174 001004  
1897 003176 005726  
1898 003200 000004 001374  
1899 003204 000751  
1900 003206 111637 001401  
1901 003212 111620  
1902 003214 122726 000015  
1903 003220 001403  
1904 003222 000004 001401  
1905 003226 000742  
1906 003230 000004 001374  
1907 003234 012600  
1908 003236 000207  
1909  
1910  
1911 003240 113746 177562  
1912 003244 042716 000200  
1913 003250 022716 000017  
1914 003254 001003  
1915 003256 112667 176507  
1916 003262 000002  
1917  
1918 003264 122726 000003  
1919 003270 001003  
1920 003272 000005  
1921 003274 000137 005340  
1922 003300 000002  
1923  
1924  
1925 003302 000000  
1926  
1927  
1928  
1929  
1930  
1931  
1932 003304 000240  
1933 003306 004767 176676  
1934 003312 110637 001123  
1935 003316 032777 020000 175454  
1936 003324 001075  
1937 003326 000004 013465

```

.INPUT: MOV      RO,-(SP)          ;SAVE RO ON THE STACK
1$:      MOV      #INBUF,RO
2$:      TSTB    @#TKS
          BPL     2$
          MOVB   @#TKB,-(SP)      ;GET CHARACTER
          BIC    #200,(SP)
          CMPB   #177,(SP)       ;CHECK RUBOUT
          BNE    3$
          CMPB   -(RO),(SP)+     ;REMOVE CHARACTER FROM INPUT
          TYPE,BKSLSH
          BR     2$              ;WAIT FOR NEXT CHARACTER
3$:      CMPB   #CNTRLU,(SP)     ;CHECK CONTROL U ( U)
          BNE    4$
          TST    (SP)+
          TYPE,CRLF
          BR     1$
4$:      MOVB   (SP),@#ECHO
          MOVB   (SP),(RO)+
          CMPB   #CR,(SP)+
          BEQ    5$
          TYPE,ECHO
          BR     2$
5$:      TYPE,CRLF
          MOV    (SP)+,RO
          RTS    PC

;KEYBOARD INTERRUPT SERVICE ROUTINE
TKISR:  MOVB   @#TKB,-(SP)      ;GET TYPED CHARACTER
          BIC    #200,(SP)      ;STRIP PARITY BIT
          CMP    #CNTRLO,(SP)   ;BRANCH IF NOT CONTROL O ( O)
          BNE    1$
          MOVB   (SP)+,$CNTRLO  ;SET CONTROL O INDICATOR IN TYPE ROUTINE
          RTI
1$:      CMPB   #3,(SP)+        ;BRANCH IF NOT CONTROL C ( C)
          BNE    2$
          RESET
          JMP    @#INIT         ;RESTART PROGRAM
2$:      RTI                    ;EXIT

.SBTTL      ERROR SERVICE ROUTINES
;ROUTINE TO PROCESS ERROR TRAPS (TRAPS TO 4)
ERRTRP: HALT

;ERROR SERVICE ROUTINE
;THIS ROUTINE PROCESSES TWO TYPES OF ERRORS (OUT OF RANGE AND HARDWARE)
;THE CALLS FOR AN OUT OF RANGE ERROR ARE <HLT+1>,<HLT+2> AND, FOR A
;HARDWARE ERROR THE CALL IS <HLT>.
.HLT:    NOP
1$:      JSR    PC,SAVE         ;SAVE REGISTERS ON THE STACK
          MOVB   SP,@#ERFLG    ;SET ERROR FLAG
          BIT    #SW13,@SWR     ;BRANCH IF NO TYPOT
          BNE    4$
          TYPE,E.HDR
    
```

```

1938 003332 113702 001122      MOVB    @#TSTNUM,R2      ;GET TEST #
1939 003336 004767 176720      JSR     PC,TYPEOCT      ;AND TYPE IT
1940 003342 016600 000016      MOV     16(SP),R0       ;GET RETURN PC
1941 003346 162700 000002      SUB     #2,R0           ;NOW PC OF HLT CALL
1942 003352 111000                MOVB    (R0),R0         ;NOW HLT CALL ITSELF
1943 003354 001417                BEQ     2$              ;BRANCH IF HLT
1944 003356 000004 013550      TYPE,E.HDR2            ;BRANCH IF NOT HLT+2
1945 003362 122700 000002      CMPB   #2,R0           ;BRANCH IF NOT HLT+2
1946 003366 001005                BNE    10$             ;TYPE GAP SPECIFIED TIMES
1947 003370 004767 177204      JSR     PC,OUTGAP       ;TYPE GAP SPECIFIED TIMES
1948 003374 000004 001374      TYPE,CRLF
1949 003400 000447                BR     4$              ;TYPE SPECIFIED TIMES
1950 003402 004767 177076      JSR     PC,OUTSPC
1951 003406 000004 001374      TYPE,CRLF
1952 003412 000442                BR     4$
1953 003414 016500 000014      MOV     ER(R5),R0
1954 003420 032765 002000 000032  BIT     #PE1600,TC(R5)
1955 003426 001403                BEQ    20$
1956 003430 042700 102100      BIC     #102100,R0
1957 003434 000402                BR     21$
1958 003436 042700 102300      BIC     #102300,R0
1959 003442 005700                TST    R0
1960 003444 001003                BNE    22$
1961 003446 000004 013441      TYPE,E.SFT             ;TYPE SOFT ERROR MESSAGE
1962 003452 000434                BR     6$
1963
1964 003454 000004 013475      22$:   TYPE,E.HDR1
1965 003460 010500                MOV    R5,R0           ;GET FIRST ADDRESS OF REGS.
1966 003462 012701 000007      MOV    #7,R1           ;TYPE FIRST 7 REGS.
1967 003466 012002                MOV    (R0)+,R2        ;GET REG CONTENTS
1968 003470 004767 176566      JSR    PC,TYPEOCT      ;AND TYPE IT
1969 003474 000004 001407      TYPE,SPACE2
1970 003500 005301                DEC    R1
1971 003502 001371                BNE    3$
1972 003504 016502 000032      MOV    TC(R5),R2       ;GET CONTENTS OF TC REGISTER
1973 003510 004767 176546      JSR    PC,TYPEOCT
1974 003514 000004 001374      TYPE,CRLF
1975
1976 003520 032777 001000 175252  4$:   BIT     #SW09,@SWR     ;BRANCH IF NO RING THE BELL
1977 003526 001402                BEQ    5$
1978 003530 000004 001403      TYPE,BELL
1979 003534 005777 175240      5$:   TST    @SWR        ;HALT ON ERROR?
1980 003540 100001                BPL    6$
1981 003542 000000                HALT
1982 003544                6$:
1983 003544 004767 176462      JSR    PC,.RESTORE     ;RESTORE REGISTERS FROM THE STACK
1984 003550 000002                RTI                    ;RETURN
1985
1986
1987                .SBTTL          SCOPE SUBROUTINE
1988                ;SCOPE ROUTINE
1989                ;THIS ROUTINE IS ENTERED UPON COMPLETION OF EACH SUBTEST
1990                ;THE SCOPE ROUTINE:
1991                ;   OUTPUTS TIME SPEC ON EACH ITERATION IF SW08 IS SET
1992                ;   REPEATS TEST IF SW14 IS SET
1993                ;   STORES ACTUAL TIME FOR FUNCTION IN TIME TABLE (ATIMTBL)

```

```

1994      : PUBLISHES TIME IF SW10=0
1995      : UPDATES ITERATION COUNT AND IF ITERATIONS COMPLETE CONTINUES
1996      : TO NEXT TEST, OTHERWISE REPEATS TEST.
1997      : DELAYS BEFORE CONTINUING OR REPEATING TEST.
1998      : INITIALIZES DRIVE
1999      : RETURNS: R5=BASE ADDRESS OF TMO2 REGISTERS (ADDRESS OF CS1)
2000      : R1='DS' REG ADDRESS
2001      : R0='FC' REG ADDRESS
2002
2003 003552 013705 001010 .SCOPE: MOV @#TMBASE,R5 ;SET R5 TO FIRST TM REG
2004 003556 032777 000400 175214 BIT #SW08,@SWR ;BRANCH IF SPECIFICATION LINE
2005 003564 001404 BEQ 10$ ;NOT DESIRED ON EACH ITERATION
2006 003566 113702 001122 MOVB @#TSTNUM,R2 ;GET TEST NUMBER
2007 003572 004767 176706 JSR PC,OUTSPC ;OUTPUT TIME RECORDED
2008 003576 032777 040000 175174 10$: BIT #SW14,@SWR ;BRANCH IF CONTINUOUS LOOP
2009 003604 001417 BEQ 2$ ;NOT DESIRED
2010 003606 004767 000542 1$: JSR PC,DELAY ;DELAY 350 MS
2011 003612 004767 000772 JSR PC,RHINIT ;INIT
2012 003616 105037 001123 CLRB @#ERFLG ;CLEAR ERROR FLAG
2013 003622 016716 175154 MOV SCPADR,(SP)
2014 003626 010501 MOV R5,R1
2015 003630 062701 000012 ADD #DS,R1 ;ADDRESS OF 'DS' REG IS IN R1
2016 003634 010500 MOV R5,R0
2017 003636 062700 000006 ADD #FC,R0 ;ADDRESS OF 'FC' REG IS IN R0
2018 003642 000002 RTI
2019
2020 003644 105737 001123 2$: TSTB @#ERFLG ;BRANCH IF ERROR FLAG IS SET
2021 003650 001006 BNE 3$
2022 003652 113700 001121 MOVB @#ITCNT,R0 ;GET ITERATION COUNT
2023 003656 006300 ASL R0 ;STORE TIME IN TABLE
2024 003660 013760 001012 001014 MOV @#ATIME,ATIMTBL(R0)
2025 003666 105237 001121 3$: INCB @#ITCNT ;INCREMENT ITERATION COUNT
2026 003672 032777 004000 175100 BIT #SW11,@SWR ;BRANCH IF SINGLE ITERATION DESIRED
2027 003700 001004 BNE 4$
2028 003702 122737 000004 001121 CMPB #4,@#ITCNT ;BRANCH IF ITERATIONS INCOMPLETE
2029 003710 001336 BNE 1$
2030 003712 011637 001002 4$: MOV (SP),@#SCPADR ;SET SCOPE ADDRESS TO NEXT TEST
2031 003716 032777 002000 175054 BIT #SW10,@SWR ;BRANCH IF NO PUBLICATION DESIRED
2032 003724 001002 BNE 5$
2033 003726 004767 177046 JSR PC,PUBLISH ;GO PUBLISH TEST DATA
2034 003732 105037 001121 5$: CLRB @#ITCNT ;RESET ITERATION COUNT
2035 003736 105777 175036 TSTB @SWR ;BRANCH IF USER DOES NOT WANT TO
2036 003742 001721 BEQ 1$ ;HALT ON A SELECTED TEST
2037 003744 017746 175030 MOV @SWR,-(SP) ;GET SWITCHES
2038 003750 042716 177740 BIC #177740,(SP) ;CLEAR ALL BUT TEST #
2039 003754 005316 DEC (SP) ;FORM TEST # -1
2040 003756 123726 001122 CMPB @#TSTNUM,(SP)+ ;BRANCH IF NOT AT TEST
2041 003762 001311 BNE 1$
2042 003764 000000 HALT
2043 003766 000707 BR 1$
2044
2045 .SBTTL TIMER SUBROUTINES
2046
2047 ;SUBROUTINE TO SYNCHRONIZE THE TIMER AND TURN IT ON.
2048 ;REGISTER 4 IS CLEARED, AND THE OSCILLATOR POLARITY IS MONITORED
2049 ;THE ROUTINE IS EXITED WHEN THE OSCILLATOR POLARITY CHANGES WITH R3

```

```

2050 ;SET TO INDICATE THE POLARITY OF THE OSCILLATOR.
2051 ;CALL: JSR PC,TIMON
2052 ;RETURNS: R3 SET TO INDICATE LAST POLARITY (+24/-24=0/1)
2053 ; R4 = 0
2054
2055 003770 005004 TIMON: CLR R4 ;CLEAR TIME COUNT
2056 003772 012703 000024 MOV #24,R3 ;SET POLARITY TO '0' STATE
2057 003776 032765 000100 000024 BIT #OSC,MR(R5) ;BRANCH IF POLARITY IS '0'
2058 004004 001405 BEQ 2$
2059 004006 032765 000100 000024 1$: BIT #OSC,MR(R5) ;WAIT FOR OSCILLATOR TO RETURN
2060 004014 001374 BNE 1$
2061 004016 000405 BR 4$
2062
2063 004020 005403 2$: NEG R3 ;NEGATE PREV POLARITY INDICATOR
2064 004022 032765 000100 000024 3$: BIT #OSC,MR(R5) ;WAIT FOR OSCILLATOR TO RETURN
2065 004030 001774 BEQ 3$ ;TO '1' STATE
2066 004032 000207 4$: RTS PC
2067
2068 ;SUBROUTINE TO COUNT TIME
2069 ;EACH TIME THE OSCILLATOR TOGGLES (BIT <06> IN MR REG) REGISTER
2070 ;R4 IS INCREMENTED, AND THE REGISTER R3 IS NEGATED TO INDICATE
2071 ;THE LAST STATE OF THE OSCILLATOR.
2072 ;CALL JMP TIMER(R3) ;R3 IS SET BY TIMON ROUTINE
2073 ; R2=RETURN ADDRESS TO CALLER
2074 ;NOTE: THE TIME TO EXECUTE THIS ROUTINE IS VERY CRITICAL. IT MUST BE
2075 ;LESS THAN 40 US.
2076
2077 ;ENTER HERE VIA JMP TIMER(R3) WHEN R3=-24 (PREV STATE=1)
2078 004034 032765 000100 000024 TIMER1: BIT #OSC,MR(R5) ;BRANCH IF CURRENT STATE IS '0'
2079 004042 001406 BEQ TIMER ;GO INCREMENT TIME
2080 004044 000112 JMP (R2) ;RETURN TO TEST
2081
2082 .=TIMER1+24
2083 004060 005403 TIMER: NEG R3 ;NEGATE PREV STATE INDICATOR
2084 004062 005204 INC R4 ;INCREMENT 'TICK' COUNT
2085 004064 100401 BMI TIMERR ;BRANCH ON OVERFLOW
2086 004066 000112 JMP (R2) ;RETURN TO TEST
2087 004070 000004 013576 TIMERR: TYPE,E.TIMOV ;TYPE 'TIMER OVERFLOWED'
2088 004074 104400 HLT ;REPORT HARDWARE ERROR
2089 004076 000177 174700 JMP @SCPADR ;RETURN TO BEGINNING OF TEST
2090
2091 .=TIMER+24
2092 ;ENTER HERE VIA JMP TIMER(R3) WHEN R3=+24 (PREV STATE=0)
2093 004104 032765 000100 000024 TIMERO: BIT #OSC,MR(R5) ;BRANCH IF CURRENT STATE = '1'
2094 004112 001362 BNE TIMER
2095 004114 000112 JMP (R2)
2096
2097 ;SUBROUTINE TO CHECK TIME RECORDED BY SUBTEST.
2098 ;THIS SUBROUTINE COMPUTES THE ACTUAL TIME (IN MICROSECONDS) AND CHECKS
2099 ;THAT THE TIME RECORDED BY THE SUBTEST IS CORRECT BY COMPARING THE TIME
2100 ;WITH THE HIGH LIMIT (STIMTBL(R0)) AND THE LOW LIMIT (STIMTBL+2(R0)).
2101 ;IF THE TIME IS OUT OF RANGE AN OUT OF RANGE ERROR TYPEOUT RESULTS.
2102 ;THE SUBROUTINE IS ENTERED WITH:
2103 ; R4=TICK COUNT
2104
2105 004116 TIMOK:

```

```

2106 004116 004767 176066      JSR    PC,,SAVE          ;SAVE REGISTERS ON THE STACK
2107 004122 012700 000104      MOV    #68.,R0          ;GET TIME PER TICK
2108 004126 010401              MOV    R4,R1            ;GET TICKS COUNT
2109 004130 005002              CLR    R2                ;CLEAR SUMMING REGISTERS
2110 004132 005003              CLR    R3
2111 004134 060002      1$:  ADD    R0,R2            ;MULTIPLY TIME PER TICK
2112 004136 005503              ADC    R3                ;BY TICK COUNT
2113 004140 005301              DEC    R1
2114 004142 001374              BNE    1$
2115 004144 010246              MOV    R2,-(SP)         ;DIVIDE COUNT BY 10.
2116
2117 004146 010346              MOV    R3,-(SP)
2118 004150 012746 000012      MOV    #10.,-(SP)
2119 004154 004767 000262      JSR    PC,DIVIDE
2120 004160 005726              TST    (SP)+            ;DISCARD REMAINDER
2121 004162 012637 001012      MOV    (SP)+,@#ATIME    ;STORE QUOTIENT
2122 004166 113700 001122      MOV    @#TSTNUM,R0      ;GET TEST #
2123 004172 006300              ASL    R0
2124 004174 006300              ASL    R0
2125 004176 023760 001012 001416  CMP    @#ATIME,STIMTBL(R0) ;CHECK THAT TIME IS WITHIN
2126 004204 101004              BHI    2$                ;LIMITS SPECIFIED
2127 004206 023760 001012 001420  CMP    @#ATIME,STIMTBL+2(R0)
2128 004214 101001              BHI    3$
2129 004216 104401      2$:  HLT+1                ;CALL ERROR ROUTINE
2130 004220      3$:
2131 004220 004767 176006      JSR    PC,,RESTORE      ;RESTORE REGISTERS FROM THE STACK
2132 004224 000207      RTS    PC                ;RETURN
2133
2134      ;SUBROUTINE TO CHECK INDIVIDUAL GAP TIMES (PRODUCED BY TST021)
2135      ;SUBROUTINE COMPUTES THE ACTUAL TIME (IN MICROSECONDS) AND CHECKS
2136      ;THAT THE GAP TIME RECORDED BY THE SUBTEST (TST021) BY COMPARING THE
2137      ;TIME WITH THE MAX LIMIT (GTIMTBL-GAPTBL(R1)) AND THE MIN LIMIT
2138      ;(GTIMTBL+2-GAPTBL(R1)).
2139      ;CALL:  MOV    #TICK COUNT,R4          ;R4 CONTAINS TICK COUNT
2140      ;      MOV    #GAP,@#GAP              ;LOCATION GAP CONTAINS GAP #
2141      ;      JSR    PC,GAPOK
2142
2143      GAPOK:
2144 004226 004767 175756      JSR    PC,,SAVE          ;SAVE REGISTERS ON THE STACK
2145 004232 012700 000104      MOV    #68.,R0          ;GET TIME PER TICK
2146 004236 010401              MOV    R4,R1            ;GET TICK COUNT
2147 004240 005002              CLR    R2                ;CLEAR SUMMING REGISTERS
2148 004242 005003              CLR    R3
2149 004244 060002      1$:  ADD    R0,R2            ;MULTIPLY TICK COUNT
2150 004246 005503              ADC    R3                ;BY TIME PER TICK
2151 004250 005301              DEC    R1
2152 004252 001374              BNE    1$
2153
2154 004254 010246              MOV    R2,-(SP)         ;DIVIDE TIME BY 10.
2155 004256 010346              MOV    R3,-(SP)
2156 004260 012746 000012      MOV    #10.,-(SP)
2157 004264 004767 000152      JSR    PC,DIVIDE
2158 004270 005726              TST    (SP)+            ;DISCARD REMAINDER
2159 004272 012637 001012      MOV    (SP)+,@#ATIME    ;STORE QUOTIENT
2160 004276 113703 001120      MOV    @#GAP,R3        ;GET GAP #
2161 004302 006303              ASL    R3                ;MULTIPLY BY 4

```

```

2162 004304 006303 ASL R3 ;TO GET AT TABLE ENTRY
2163 004306 023763 001012 001572 CMP @#ATIME,GTIMTBL(R3) ;CHECK TIME (MAX)
2164 004314 101004 BHI 2$
2165 004316 023763 001012 001574 CMP @#ATIME,GTIMTBL+2(R3) ;CHECK TIME (MIN)
2166 004324 101002 BHI 3$
2167 004326 104402 2$: HLT+2 ;REPORT OUT OF RANGE ERROR
2168 004330 000406 BR 100$
2169 004332 032777 000400 174440 3$: BIT #SW08,@SWR ;BRANCH IF TIMES NOT WANTED
2170 004340 001402 BEQ 100$
2171 004342 004767 176232 JSR PC,OUTGAP ;TYPE GAP TIMES
2172
2173 004346 100$:
2174 004346 004767 175660 JSR PC,.RESTORE ;RESTORE REGISTERS FROM THE STACK
2175 004352 000207 RTS PC ;RETURN TO TEST
2176
2177 .SBTTL DELAY SUBROUTINES
2178 ;THIS SUBROUTINE CAUSES A DELAY OF 350 MS.
2179 004354 004767 177410 DELAY: JSR PC,TIMON
2180 004360 010246 MOV R2,-(SP) ;SAVE R2 ON THE STACK
2181 004362 012702 004372 MOV #2$,R2 ;SET RETURN ADDRESS FOR TIMER
2182 004366 1$:
2183 004366 000163 004060 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2184 004372 032704 004000 2$: BIT #4000,R4
2185 004376 001773 BEQ 1$
2186 004400 012602 MOV (SP)+,R2 ;RESTORE R2
2187 004402 000207 RTS PC
2188
2189 ;THIS SUBROUTINE ALLOWS A CALLER SPECIFIED DELAY (UP TO 65MS.)
2190 ;CALL: MOV DELAY TIME,DELTIM ;LOAD DELAY TIME (IN US)
2191 ; JSR PC,DELAYV
2192 004404 005767 174504 DELAYV: TST DELTIM ;BRANCH IF 0 DELAY
2193 004410 001413 BEQ 3$
2194 004412 004767 177352 JSR PC,TIMON ;TURN TIMER ON
2195 004416 010246 MOV R2,-(SP) ;SAVE R2 ON THE STACK
2196 004420 012702 004430 MOV #2$,R2 ;SET RETURN ADDRESS FROM TIMER
2197 004424 1$:
2198 004424 000163 004060 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2199 004430 023704 001114 2$: CMP @#DELTIM,R4
2200 004434 101373 BHI 1$
2201 004436 012602 MOV (SP)+,R2 ;RESTORE R2
2202 004440 000207 3$: RTS PC
2203
2204 .SBTTL DIVIDE SUBROUTINE
2205 ;THIS SUBROUTINE DIVIDES A DOUBLE PRECISION # AND RETURNS THE RESULT
2206 ;TO THE CALLER ON THE STACK. BOTH DIVIDEND & DIVISOR MUST BE POSITIVE.
2207 ;CALL: MOV LEAST SIGNIFICANT HALF DIVIDEND,-(SP)
2208 ; MOV #MOST SIGNIFICANT HALF DIVIDEND,-(SP)
2209 ; MOV #DIVISOR,-(SP)
2210 ; JSR PC,DIVIDE
2211 ;RETURN
2212 ; (SP)=REMAINDER ON STACK
2213 ; 2(SP)=QUOTIENT
2214
2215 ;NOTE: THIS SUBROUTINE DESTROYS PREVIOUS CONTENTS OF R0,R1,R2 & R3.
2216
2217 004442 005046 DIVIDE: CLR -(SP) ;SAVE LOC FOR SIGNS

```

```

2218 004444 012746 000021      MOV      #17,-(SP)      ;SET ITERATION COUNT
2219 004450 016601 000012      MOV      12(SP),R1     ;GET LSH DIVIDEND
2220 004454 016600 000010      MOV      10(SP),R0     ;GET MSH DIVIDEND
2221 004460 016602 000006      MOV      6(SP),R2      ;GET DIVISOR
2222 004464 005402              NEG      R2             ;NEGATE DIVISOR
2223 004466 000241              CLC                    ;CLEAR 'C' BIT IN PSW
2224 004470 000405              BR       2$            ;
2225 004472 006100      1$:    ROL      R0             ;ROTATE MSH DIVIDEND
2226 004474 010003      MOV      R0,R3         ;SAVE IN R3
2227 004476 060203      ADD      R2,R3         ;SUBTRACT DIVISOR FROM MSH DIVIDEND
2228 004500 103001      BCC      2$            ;BRANCH IF DIVIDEND > DIVISOR
2229 004502 010300      MOV      R3,R0         ;SAVE REMAINDER IN R0
2230 004504 006101      2$:    ROL      R1             ;ROTATE LSH DIVIDEND
2231 004506 005316      DEC      (SP)          ;DECREMENT ITERATION COUNT
2232 004510 001370      BNE      1$            ;
2233 004512 005726      TST      (SP)+         ;POP ITERATION COUNTER
2234 004514 005726      TST      (SP)+         ;POP SIGN CORRECTION
2235 004516 010166 000006      MOV      R1,6(SP)      ;PUSH REMAINDER ON STACK
2236 004522 010066 000004      MOV      R0,4(SP)      ;PUSH QUOTIENT ONTO STACK
2237 004526 012616      MOV      (SP)+,(SP)
2238 004530 000207      RTS      PC
2239
2240              .SBTTL  DRIVE SUBROUTINES
2241      ;SUBROUTINE TO CHECK IF DRIVE IS AVAILABLE
2242      ;CALL:  MOVB   DRIVE#,DRVNUM
2243      ;      JSR    PC,DRVAVA
2244      ;      ;RETURN:  'C' BIT SET IF NOT AVAILABLE
2245 004532 113765 001004 000010  DRVAVA:  MOVB   @#DRVNUM,CS2(R5)  ;LOAD DRIVE #
2246 004540 032765 040000 000026      BIT    #TAP,DT(R5)      ;CHECK IF TAPE UNIT
2247 004546 001003              BNE    1$
2248 004550 004767 000034      JSR    PC,RHINIT
2249 004554 000262              SEV                    ;SET 'V' TO IND NOT AVAIL
2250 004556 000207      1$:    RTS      PC      ;RETURN
2251
2252      ;SUBROUTINE TO CHECK IF TU45 SLAVE IS AVAILABLE FOR TEST
2253      ;CALL:  MOVB   DRIVE #,@#DRVNUM      ;PASS DRIVE # VIA DRVNUM
2254      ;      MOVB   SLAVE #,@#SLVNUM      ;PASS SLAVE # VIA SLVNUM
2255      ;      JSR    PC,SLVAVA      ;CALL SUBROUTINE
2256 004560 113765 001004 000010  SLVAVA:  MOVB   @#DRVNUM,CS2(R5)  ;LOAD DRIVE #
2257 004566 113765 001005 000032      MOVB   @#SLVNUM,TC(R5)      ;AND SLAVE #
2258 004574 032765 002000 000026      BIT    #SPR,DT(R5)      ;BRANCH IF SLAVE PRESENT
2259 004602 001001              BNE    1$
2260 004604 000262              SEV                    ;SET 'V' TO INDICATE NO SLAVE
2261 004606 000207      1$:    RTS      PC
2262
2263      ;SUBROUTINE TO INITIALIZE RH CONTROLLER
2264      ;CALL:  JSR    PC,RHINIT
2265
2266 004610 012765 000040 000010  RHINIT:  MOV    #40,CS2(R5)
2267 004616 113765 001004 000010      MOVB   @#DRVNUM,CS2(R5)
2268 004624 005046              CLR    -(SP)
2269 004626 113716 001005              MOVB   @#SLVNUM,(SP)
2270 004632 012665 000032              MOV    (SP)+,TC(R5)      ;LOAD SLAVE # INTO TC REG
2271 004636 052765 000300 000032      BIS    #NORM11,TC(R5)
2272 004644 000207      RTS    PC
2273

```

```
2274 ;SUBROUTINE TO WAIT FOR DRIVE READY (DRY)
2275 004646 005027 WAITRDY:CLR (PC)+ ;CLEAR WAIT TIMER
2276 004650 000000 WAITTIM:.WORD 0
2277 004652 105765 000012 1$: TSTB DS(R5) ;WAIT FOR READY TO SET
2278 004656 100406 BMI 2$
2279 004660 005267 177764 INC WAITTIM ;INCREMENT WAIT TIMER
2280 004664 001372 BNE 1$ ;BRANCH IF TIME HAS NOT EXPIRED
2281 004666 000004 013623 TYPE,E.TIMEXP ;TYPE 'TIME EXPIRED WAITING FOR RDY'
2282 004672 000425 BR 99$ ;TAKE ERROR EXIT
2283 004674 032765 002000 000012 2$: BIT #EOT,DS(R5) ;CHECK FOR END OF TAPE
2284 004702 001415 BEQ 3$ ;BRANCH IF NO EOT
2285 004704 000004 012660 TYPE,M.NAM ;TYPE 'END OF TAPE'
2286 004710 000004 013206 TYPE,M.EOT ;REWIND SLAVE
2287 004714 004767 000032 JSR PC,.REWIND ;BRANCH IF ERROR ON REWIND
2288 004720 102412 BVS 99$ ;WRITE A RECORD
2289 004722 004767 000106 JSR PC,WRITE ;SET 'GO' BIT
2290 004726 005215 INC (R5) ;WAIT FOR READY
2291 004730 004767 177712 JSR PC,WAITRDY ;TAKE ERROR EXIT
2292 004734 000404 BR 99$ ;CHECK ERROR EXIT
2293 004736 032765 040000 000012 3$: BIT #ERR,DS(R5)
2294 004744 001401 BEQ 100$
2295 004746 000262 99$: SEV
2296 004750 000207 100$: RTS PC
2297
2298 ;SUBROUTINE TO REWIND A UNIT (DRIVE/SLAVE COMBINATION)
2299 ;CALL MOVB DRIVE #,2#DRVNUM
```

CZTULAO TMO2 DRIVE FUNCTION TIMER  
CZTULA.P11 07-JUN-78 17:07

MACY11 30(1046) 13-JUN-78 13:54 PAGE 52  
DRIVE SUBROUTINES

SEQ 0052

2300  
2301  
2302  
2303  
2304  
2305 004752 004767 177632

:       MOVB    SLAVE #,@#SLVNUM  
:       JSR     PC,.REWIND  
:SUBROUTINE RETURNS TO CALLER WITH SELECTED SLAVE AT 'BOT', & 'V' SET IF  
:AN ERROR OCCURS.  
  
.REWIND:JSR     PC,RHINIT                    :INITIALIZE CONTROLLER

```

2306 004756 004367 000206      JSR      R3,TMCMD      ;GO TO TM COMMAND SUBROUTINE
2307 004762 000000              .WORD    0             ;BUS ADDRESS (NOT USED)
2308 004764 000000              .WORD    0             ;WORD COUNT (NOT USED)
2309 004766 000000              .WORD    0             ;FRAME COUNT (NOT USED)
2310 004770 000006              .WORD    RWD           ;REWIND COMMAND
2311 004772 005215              INC      (R5)          ;SET 'GO' BIT
2312 004774 032765 000002 000012 1$:  BIT      #BOT,DS(R5)   ;BRANCH IF 'BOT' SET
2313 005002 001005              BNE     2$             ;
2314 005004 032765 040000 000012  BIT      #ERR,DS(R5)   ;CHECK ERROR BIT
2315 005012 001006              BNE     99$           ;BRANCH IF ERROR BIT SET
2316 005014 000767              BR      1$            ;
2317
2318 005016 032765 020000 000012 2$:  BIT      #PIP,DS(R5)   ;WAIT FOR TAPE MOTION TO STOP
2319 005024 001374              BNE     2$             ;
2320 005026 000401              BR      100$          ;
2321 005030 000262              99$:  SEV              ;
2322 005032 000207              100$: RTS      PC      ;
2323
2324      ;SUBROUTINE TO WRITE 256. WORD RECORD
2325      ;CALL: JSR      PC,WRITE
2326
2327 005034 004367 000130  WRITE: JSR      R3,TMCMD      ;GO TO TM COMMAND SUBROUTINE
2328 005040 015176              .WORD    WTBUF         ;BUS ADDRESS
2329 005042 177600              .WORD    WRDCNT        ;WORD COUNT
2330 005044 177400              .WORD    FRMCNT        ;FRAME COUNT
2331 005046 000060              .WORD    WFWD          ;WRITE FORWARD COMMAND
2332 005050 000207              RTS      PC            ;
2333
2334      ;SUBROUTINE TO READ A 256. WORD RECORD.
2335      ;CALL: JSR      PC,READ
2336
2337 005052 004337 005170  READ:  JSR      R3,@#TMCMD
2338 005056 015176              .WORD    RDBUF         ;ADDRESS OF READ BUFFER
2339 005060 177600              .WORD    WRDCNT        ;2'S COMPLEMENT OF WORD COUNT
2340 005062 177400              .WORD    FRMCNT        ;2'S COMPLEMENT OF FRAME COUNT
2341 005064 000070              .WORD    RDFWD         ;READ FORWARD COMMAND
2342 005066 000207              RTS      PC            ;
2343
2344      ;SUBROUTINE TO INITIATE READ REVERSE COMMAND
2345      ;CALL: JSR      PC,REVRD
2346
2347 005070 004367 000074  REVRD: JSR      R3,TMCMD
2348 005074 015576              .WORD    RDBUF+256.   ;ADDRESS OF READ REVERSE BUFFER
2349 005076 177600              .WORD    WRDCNT        ;2'S COMPLEMENT OF WORD COUNT
2350 005100 177400              .WORD    FRMCNT        ;2'S COMPLEMENT OF FRAME COUNT
2351 005102 000076              .WORD    RDREV         ;READ REVERSE COMMAND
2352 005104 000207              RTS      PC            ;
2353
2354      ;SUBROUTINE TO SPACE FORWARD 1 RECORD
2355 005106 012765 177777 000006  FWDSPC: MOV     #-1,FC(R5)  ;LOAD RECORD COUNT
2356 005114 012715 000031      MOV     #SPCFWD+1,(R5) ;LOAD COMMAND
2357 005120 004767 177522      JSR      PC,WAITRDY   ;WAIT FOR READY
2358 005124 000207      RTS      PC            ;RETURN
2359
2360      ;SUBROUTINE TO WRITE A RECORD AND BACK SPACE OVER THE RECORD.
2361 005126 004767 177702  WRT.BK: JSR      PC,WRITE ;WRITE THE RECORD

```

```

2362 005132 005215          INC      (R5)          ;SET 'GO' BIT
2363 005134 004767 177506   JSR      PC,WAITRDY
2364 005140 102412          BVS      2$
2365 005142 012765 177777 000006   MOV      #-1,FC(R5)    ;LOAD RECORD COUNT
2366 005150 012715 000033     MOV      #SPCREV+1,(R5) ;LOAD COMMAND
2367 005154 004767 177466   JSR      PC,WAITRDY
2368 005160 102402          BVS      2$
2369 005162 004767 177166   1$:     JSR      PC,DELAY    ;WAIT FOR TAPE MOTION TO STOP
2370 005166 000207          2$:     RTS      PC
2371
2372          ;SUBROUTINE TO LOAD A COMMAND
2373          ;CALL: JSR      R3,TMCM
2374          :         .WORD   BUS ADDRESS
2375          :         .WORD   WORD COUNT (2'S COMPLEMENT)
2376          :         .WORD   FRAME COUNT (2'S COMPLEMENT)
2377          :         .WORD   COMMAND
2378
2379 005170 012365 000004   TMCM:   MOV      (R3)+,BA(R5)    ;LOAD BUS ADDRESS
2380 005174 012365 000002     MOV      (R3)+,WC(R5)    ;LOAD WORD COUNT
2381 005200 012365 000006     MOV      (R3)+,FC(R5)    ;LOAD FRAME COUNT
2382 005204 012315          MOV      (R3)+,(R5)      ;LOAD COMMAND
2383 005206 000203          RTS      R3              ;RETURN
2384
2385          ;SUBROUTINE TO PRINT TU45 SERIAL NUMBER
2386          ;JSR      PC,SNPT
2387
2388 005210 016503 000030   SNPT:   MOV      SN(R5),R3
2389 005214 012701 001144     MOV      #ODIGITS,R1
2390 005220 000303          SWAB     R3
2391 005222 006003          ROR      R3
2392 005224 006003          ROR      R3
2393 005226 006003          ROR      R3
2394 005230 006003          ROR      R3              ;GET FIRST DIGIT
2395 005232 042703 177760   BIC      #177760,R3
2396 005236 052703 000260   BIS      #260,R3
2397 005242 110321          MOV      R3,(R1)+        ;FILL FIRST DIGIT
2398 005244 016503 000030   MOV      SN(R5),R3
2399 005250 000303          SWAB     R3
2400 005252 042703 177760   BIC      #177760,R3
2401 005256 052703 000260   BIS      #260,R3
2402 005262 110321          MOV      R3,(R1)+        ;GET SECOND DIGIT
2403 005264 016503 000030   MOV      SN(R5),R3
2404 005270 006003          ROR      R3
2405 005272 006003          ROR      R3
2406 005274 006003          ROR      R3
2407 005276 006003          ROR      R3
2408 005300 042703 177760   BIC      #177760,R3
2409 005304 052703 000260   BIS      #260,R3
2410 005310 110321          MOV      R3,(R1)+        ;GET THIRD DIGIT
2411 005312 016503 000030   MOV      SN(R5),R3
2412 005316 042703 177760   BIC      #177760,R3
2413 005322 052703 000260   BIS      #260,R3
2414 005326 110321          MOV      R3,(R1)+        ;GET FOURTH DIGIT
2415 005330 105011          CLRB     (R1)
2416 005332 000004 001144   TYPE,ODIGITS           ;TYPE SERIAL NUMBER
2417 005336 000207          RTS      PC              ;RETURN

```

```

2418
2419
2420
2421 005340 012706 000600          INIT:  .SBTTL  PROGRAM INITIALIZATION
2422 005344 105037 001124          MOV      #STKPTR,SP          ;SET STACK PTR
2423 005350 105037 001121          CLRB    @#PRGFLG          ;CLEAR PROGRAM FLAG
2424 005354 105037 001122          CLRB    @#ITCNT          ;CLEAR ITERATION COUNT
2425 005360 105037 001123          CLRB    @#TSTNUM          ;SET TEST # 0
2426 005364 105067 173540          CLRB    @#ERFLG          ;CLEAR ERROR FLAG
2427 005370 012737 000006 000004  CLRB    ASFLG            ;CLEAR ASK FLAG
2428 005376 012737 000002 000006  MOV      #ERRVEC+2,@#ERRVEC
2429 005404 005037 001264          MOV      #RTI,@#ERRVEC+2   ;CHECK IF 'LP' IS AVAILABLE
2430 005410 000004 001374          2$:    CLR      @#INBUF
2431 005414 000004 012660          TYPE,CRLF
2432 005420 000004 012726          TYPE,M.NAM                ;TYPE TITLE
2433 005424 004767 175476          TYPE,I.REG                ;ASK USER TO TYPE CONT BASE ADRS
2434 005430 004767 175256          JSR     PC,..INPUT        ;GET USER INPUT
2435 005434 013737 001116 001010  JSR     PC,CNVTAO          ;CONVERT ASCII TO OCTAL
2436 005442 013705 001010          4$:    MOV      @#OCTALO,@#TMBASE ;SET NEW ADDRESS
2437
2438
2439 005446 000261          5$:    MOV      @#TMBASE,R5
2440 005450 005715          ;ROUTINE TO CHECK IF CONTROLLER (RH11) IS AVAILAABLE
2441 005452 103003          SEC
2442 005454 000004 013245          TST     (R5)              ;SET 'C' IN PSW
2443 005460 000727          BCC     6$                ;BRANCH IF CONTROLLER AVAIL
2444 005462 012737 003302 000004  BCC     6$
2445
2446 005470 105037 001123          TYPE,E.NCOM
2447 005474 012701 001154          BR      INIT
2448 005500 012700 000004          6$:    MOV      #ERRTRP,@#ERRVEC ;SET ERROR TRAP VECTOR
2449 005504 005021          ;ROUTINE TO GET TMO2 DRIVES USER DESIRES TO TEST
2450 005506 005300          DRIVES: CLRB    @#ERFLG    ;CLEAR ERROR FLAG
2451 005510 001375          MOV      #DRVTBL,R1       ;MARK ALL DRIVES AS NOT TO
2452 005512 000004 012773          MOV      #4,R0            ;BE TESTED. A '0' INDICATES
2453 005516 004767 175404          1$:    CLR      (R1)+       ;THAT A DRIVE IS NOT TO BE
2454 005522 012700 001264          DEC     R0                ;TESTED
2455 005526 122710 000101          BNE     1$
2456 005532 001013          TYPE,I.DRVS
2457 005534 110667 173364          JSR     PC,..INPUT        ;GET USER INPUT
2458 005540 012701 001154          MOV      #INBUF,R0
2459 005544 012700 000004          CMPB    #'A',(R0)         ;AN 'A' SPECIFIES ALL
2460 005550 012721 177777          BNE     3$                ;DRIVES TO BE TESTED
2461 005554 005300          MOVB    SP,PRGFLG         ;SET FLAG TO IND ALL DRIVES
2462 005556 001374          MOV      #DRVTBL,R1       ;MARK ALL DRIVES TO BE TESTED
2463 005560 000417          MOV      #4,R0            ;A '-1' INDICATES THAT A DRIVE
2464
2465          2$:    MOV      #-1,(R1)+    ;IS TO BE TESTED
2466          DEC     R0
2467          BNE     2$
2468          BR      CHKDRV        ;GO CHECK DRIVE AVAILABILITY
2469
2470          ;GET USER SELECTED DRIVES AND MARK EACH DRIVE SELECTED TO BE TESTED
2471          3$:    CMPB    #CR,(R0)
2472          BEQ     CHKDRV
2473          CMPB    (R0),#',    ;CHECK IF 'COMMA'
2474          BNE     4$
2475          TSTB    (R0)+       ;STEP PTR PAST 'COMMA'
2476          4$:    MOVB    (R0)+,R1
2477          BIC    #177770,R1
2478          MOVB    #-1,DRVTBL(R1)

```

```

2474 005614 000240          NOP
2475 005616 000761          BR      3$
2476
2477          ;ASCERTAIN THAT DRIVES (TMO2'S) SPECIFIED ARE AVAILABLE
2478 005620 005000          CHKDRV: CLR      RO          ;A 0/-1 INDICATES THAT THE
2479 005622 105760 001154  1$:  TSTB     DRVTBL(RO)    ;DRIVE IS NOT/IS TO BE TESTED
2480 005626 001005          BNE      3$
2481 005630 005200          2$:  INC      RO
2482 005632 122700 000010  CMPB     #8.,RO
2483 005636 001371          BNE      1$
2484 005640 000421          BR       4$
2485 005642 110037 001004  3$:  MOVB     RO,@#DRVNUM
2486 005646 004737 004532  JSR      PC,@#DRVAVA      ;CHECK IF AVAILABLE
2487 005652 102366          BVC      2$                ;'V' BIT SET INDICATES NOT AVAIL
2488 005654 000004 013312  TYPE,E.NDRV
2489 005660 116037 001132 013344 MOVB     DIGTAB(RO),@#E.NAVA ;SET DRIVE # IN MESSAGE
2490 005666 000004 013344  TYPE,E.NAVA
2491 005672 110637 001123  MOVB     SP,@#ERFLG        ;SET 'ERROR' FLAG
2492 005676 105060 001154  CLRB     DRVTBL(RO)        ;MARK DRIVE UNAVAILABLE
2493 005702 000752          BR       2$                ;CHECK NEXT DRIVE
2494 005704 105737 001123  4$:  TSTB     @#ERFLG        ;GO GET SLAVES IF NO ERROR
2495 005710 001403          BEQ      SLAVES
2496 005712 105737 001124  TSTB     @#PRGFLG          ;ASK USER TO RETYPE DRIVES IF
2497 005716 001664          BEQ      DRIVES           ;'ALL' NOT SPECIFIED
2498
2499          ;ROUTINE TO GET SLAVES (TU45'S) USER DESIRES TO TEST
2500 005720 105037 001123  SLAVES: CLRB    @#ERFLG     ;CLEAR 'ERROR' FLAG
2501 005724 012701 001164  MOV      #SLVTBL,R1        ;MARK ALL SLAVES (64.) AS NOT
2502 005730 012700 000040  MOV      #32.,RO          ;TO BE TESTED.A 0 INDICATES THAT
2503 005734 005021          1$:  CLR      (R1)+          ;A DRIVE'S SLAVE IS NOT TO BE
2504 005736 005300          DEC      RO                ;TESTED
2505 005740 001375          BNE      1$
2506 005742 005000          CLR      RO                ;RO = DRIVE # FOR SLAVES
2507 005744 012701 001164  MOV      #SLVTBL,R1        ;R1 POINTS TO DRIVE'S SLAVE
2508 005750 105760 001154  2$:  TSTB     DRVTBL(RO)    ;IF DRIVE IS TO BE TESTED
2509 005754 001007          BNE      4$                ;GO TO 4$ OTHERWISE
2510 005756 062701 000010  3$:  ADD      #8.,R1          ;STEP SLAVE PTR TO NEXT DRIVE'S
2511 005762 005200          INC      RO                ;SLAVES AND INCREMENT DRIVE #
2512 005764 122700 000010  CMPB     #8.,RO          ;CHECK ALL DRIVES
2513 005770 001367          BNE      2$                ;AND WHEN ALL DRIVES CHECKED
2514 005772 000454          BR       CHKSLV           ;GO CHECK SLAVE AVAILABILITY
2515
2516 005774 105737 001124  4$:  TSTB     @#PRGFLG        ;BRANCH IF USER SELECTED ALL
2517 006000 001020          BNE      5$                ;DRIVES
2518 006002 110067 172776  MOVB     RO,DRVNUM        ;GET DRIVE #
2519 006006 116037 001132 013054 MOVB     DIGTAB(RO),@#I.DRV ;PREPARE USER ACTION MESSAGE
2520 006014 000004 013035  TYPE,I.SLVS
2521 006020 004767 175102  JSR      PC,,INPUT        ;GET USER INPUT
2522 006024 012703 001264  MOV      #INBUF,R3        ;SET PTR TO USER INPUT
2523 006030 122710 000101  CMPB     #'A,(RO)        ;BRANCH IF USER DOES NOT WANT
2524 006034 001015          BNE      7$                ;'ALL' SLAVES
2525 006036 110637 001124  MOVB     SP,@#PRGFLG      ;SET 'ALL' INDICATOR
2526 006042 012701 001164  5$:  MOV      #SLVTBL,R1        ;MARK ALL SLAVES FOR ALL
2527 006046 012700 000040  MOV      #32.,RO          ;DRIVES AS TO BE TESTED
2528 006052 012721 177777  6$:  MOV      #-1,(R1)+
2529 006056 005300          DEC      RO

```

```
2530 006060 001374          BNE      6$
2531 006062 105737 001124    TSTB    @#PRGFLG          ;BRANCH IF ALL WAS SELECTED
2532 006066 001016          BNE      CHKSLV
2533
2534 006070 122713 000015    7$:     CMPB    #CR,(R3)    ;GET USER SELECTED SLAVES FOR
2535 006074 001730          BEQ      3$                ;DRIVE
2536 006076 121327 000054    CMPB    (R3),#',          ;STEP PTR PAST 'COMMA'
2537 006102 001001          BNE      8$
2538 006104 105723          TSTB    (R3)+
2539 006106 112304          8$:     MOVB    (R3)+,R4      ;AND MARK SELECED SLAVE
2540 006110 042704 177770    BIC     #177770,R4        ;AS TO BE TESTED
2541 006114 060104          ADD     R1,R4
2542 006116 112714 177777    MOVB    #-1,(R4)
2543 006122 000762          BR      7$
2544
2545          ;ASCERTAIN THAT SLAVES (TU45'S) SELECTED ARE AVAILABLE
2546 006124 005000    CHKSLV: CLR     R0          ;R0 WILL CONTAIN THE DRIVE #
2547 006126 005001          CLR     R1                ;AND R1 THE SLAVE #
2548 006130 012702 001164    MOV     #SLVTBL,R2        ;SET PTR TO SLAVE TABLE
2549 006134 105760 001154    1$:     TSTB    DRVTBL(R0)  ;BRANCH IF DRIVE SELECTED
2550 006140 001007          BNE     3$                ;& AVAILABLE FOR TEST
2551 006142 005200          2$:     INC     R0          ;INCREMENT DRIVE #
2552 006144 062702 000010    ADD     #8.,R2            ;STEP SLAVE PTR TO NEXT DRIVE'S
2553 006150 022700 000010    CMP     #8.,R0            ;SLAVES. BRANCH TO 1$ IF NOT ALL
2554 006154 001367          BNE     1$                ;DRIVES CHECKED OTHERWISE EXIT
2555 006156 000434          BR      7$
2556
2557 006160 005001          3$:     CLR     R1          ;SET SLAVE # 0
2558 006162 105712          4$:     TSTB    (R2)        ;BRANCH IF DRIVE'S SLAVE IS SEL-
2559 006164 001006          BNE     6$                ;ECTED FOR TEST
2560 006166 005201          5$:     INC     R1          ;INCREMENT SLAVE #
2561 006170 005202          INC     R2                ;STEP PTR TO NEXT SLAVE
2562 006172 022701 000010    CMP     #8.,R1            ;GO TO 4$ IF ALL SLAVES NOT
2563 006176 001371          BNE     4$                ;CHECKED
2564 006200 000760          BR      2$                ;OTHERWISE GO TO 2$ ABOVE
2565
2566 006202 110037 001004    6$:     MOVB    R0,@#DRVNUM  ;PASS DRIVE & SLAVE #
2567 006206 110137 001005    MOVB    R1,@#SLVNUM
2568 006212 004737 004560    JSR     PC,@#SLVAVA
2569 006216 102363          BVC     5$                ;AND CHECK IF AVAILABLE
2570 006220 116037 001132 013334    MOVB    DIGTAB(R0),@#E.DRV ;'V' BIT SET ON RETURN IND-
2571 006226 116137 001132 013344    MOVB    DIGTAB(R1),@#E.NAVA ;ICATES ERROR. PREPARE ERROR
2572 006234 000004 013326    TYPE ,E.NSLV            ;MESSAGE
2573 006240 110637 001123    MOVB    SP,@#ERFLG
2574 006244 105012          CLRB   (R2)                ;SET ERROR INDICATOR
2575 006246 000747          BR      5$                ;CLEAR SLAVE TABLE ENTRY
2576
2577 006250 105737 001123    7$:     TSTB    @#ERFLG
2578 006254 001403          BEQ     100$              ;BRANCH IF NO ERROR
2579 006256 105737 001124    TSTB    @#PRGFLG
2580 006262 001616          BEQ     SLAVES            ;BRANCH IF NOT 'ALL'
2581 006264 012737 003302 000004 100$:   MOV     #ERRTRP,@#ERRVEC  ;ASK USER TO RETYPE SLAVES
2582
2583          ;SCAN DIVE AND SLAVE TABLE FOR DRIVE/SLAVE COMBINATION TO TEST
2584 006272 105037 001004    CLRB   @#DRVNUM          ;SET DRIVE AND SLAVE # 0
2585 006276 105037 001005    CLRB   @#SLVNUM
```

CZTULAO TMO2 DRIVE FUNCTION TIMER  
CZTULA.P11 07-JUN-78 17:07

MACY11 30(1046) 13-JUN-78 13:54 PAGE 58  
PROGRAM INITIALIZATION

SEQ 0058

2586	006302	012737	001164	001006		MOV	#SLVTBL,@#SLVPTR	;SET PTR TO SLAVE TABLE
2587	006310	105037	001125			CLRB	@#UNTFND	;CLEAR 'UNIT FOUND' IND.
2588								
2589	006314	113700	001004		BEGIN:	MOVB	@#DRVNUM,R0	;GET DRIVE #
2590	006320	113701	001005			MOVB	@#SLVNUM,R1	;AND SLAVE #

```
2591 006324 013702 001006      MOV      @#SLVPTR,R2      ;GET SLAVE PTR
2592 006330 105760 001154      1$:     TSTB     DRVTBL(R0) ;BRANCH IF DRIVE AVAIL TO TEST
2593 006334 001011                BNE      3$
2594 006336 005001                CLR      R1              ;CLEAR SLAVE #
2595 006340 062702 000010      ADD      #8.,R2          ;AND STEP PTR TO NEXT DRIVE'S
2596 006344 005200                2$:     INC      R0          ;SLAVES AND INCREMENT DRIVE #
2597 006346 022700 000010      CMP      #8.,R0         ;EXIT TEST IF ALL DRIVES
2598 006352 001366                BNE      1$              ;CHECKED OTHERWISE CONTINUE
2599 006354 000137 012266      JMP      @#END           ;SCAN FOR NEXT 'UNIT'
2600
2601 006360 105712                3$:     TSTB     (R2)        ;BRANCH IF SLAVE ON DRIVE IS
2602 006362 001007                BNE      4$              ;AVAILABLE THERWISE STEP
2603 006364 005202                INC      R2              ;PTR TO NEXT SLAVE
2604 006366 005201                INC      R1              ;INCREMENT SLAVE #
2605 006370 122701 000010      CMPB     #8.,R1         ;UNTIL ALL SLAVES CHECKED
2606 006374 001371                BNE      3$              ;WHEN ALL SLAVES CHECKED
2607 006376 005001                CLR      R1              ;SET SLAVE # 0
2608 006400 000761                BR       2$              ;AND CONTINUE SCAN
2609
2610 006402 110637 001125      4$:     MOVB     SP,@#UNTFND ;INDICATE THAT A 'UNIT' IS FOUND
2611 006406 110037 001004      MOVB     RO,@#DRVNUM    ;SET DRIVE 3
```

```

2612 006412 110137 001005      MOV  R1,@#SLVNUM      ;SET SLAVE #
2613 006416 010237 001006      MOV  R2,@#SLVPTR     ;SAVE SLAVE PTR
2614
2615 006422 105737 001130      5$:  TSTB @#ASFLG
2616 006426 001044              BNE  7$
2617 006430 112767 000001 172472  MOV  #1,ASFLG
2618
2619 006436 105037 001124      CLRB @#PRGFLG        ;CLEAR PROGRAM INDICATOR
2620 006442 000004 013114      TYPE,I.SKEW          ;ASK USER IF HE WANTS TO RUN SKEW TESTS
2621 006446 004767 174454      JSR  PC,,INPUT       ;GET USER INPUT
2622 006452 012703 001264      MOV  #INBUF,R3       ;GET REPLY
2623 006456 122713 000060      CMPB #'0,(R3)        ;BRANCH IF 'NO' (0)
2624 006462 001406              BEQ  6$
2625 006464 122713 000061      CMPB #'1,(R3)        ;CHECK IF 'YES' (1)
2626 006470 001354              BNE  5$              ;NEITHER SO ASK AGAIN
2627 006472 111337 001124      MOV  (R3),@#PRGFLG   ;SET INDICATOR
2628 006476 000420              BR   7$
2629
2630 006500 105037 001127      6$:  CLRB @#NRZFLG      ;CLEAR NRZ INDICATOR
2631 006504 000004 013155      TYPE,I.NRZ          ;ASK USER IF DRIVE 'NRZ' ONLY
2632 006510 004767 174412      JSR  PC,,INPUT       ;GET USER INPUT
2633 006514 012703 001264      MOV  #INBUF,R3       ;GET REPLY
2634 006520 122713 000060      CMPB #'0,(R3)        ;BRANCH IF 'NO' (0)
2635 006524 001405              BEQ  7$
2636 006526 122713 000061      CMPB #'1,(R3)        ;CHECK IF 'YES' (1)
2637 006532 001362              BNE  6$              ;ASK AGAIN IF NEITHER
2638 006534 111337 001127      MOV  (R3),@#NRZFLG   ;SET INDICATOR
2639 006540
2640
2641 006540 052737 000100 177560  TYPHDR: BIS  #100,@#TKS      ;SET KEYBOARD IE BIT
2642 006546 000004 013674      TYPE,L.HDR1
2643 006552 116037 001132 014054  MOV  DIGTAB(R0),@#L.DRV ;SET DRIVE #
2644 006560 116137 001132 014066  MOV  DIGTAB(R1),@#L.SLV ;AND SLAVE #
2645 006566 112737 000071 014071  MOV  #'9,@#L.CHAN      ;GET SLAVES CHANNEL TYPE
2646 006574 032765 010000 000026  BIT  #CH7,DT(R5)
2647 006602 001403              BEQ  1$
2648 006604 112737 000067 014071  MOV  #'7,@#L.CHAN      ;SET 7 CHANNEL
2649 006612 000004 014007      1$:  TYPE,L.HDR2
2650 006616 004767 175766      JSR  PC,RHINIT        ;INIT RH
2651 006622 004767 176362      JSR  PC,SNPT          ;GO PRINT SERIAL NUMBER
2652 006626 000004 014110      TYPE,L.HDR3
2653 006632 012737 006672 001002  MOV  #TST001,@#SCPADR  ;SET 'SCOPE' ADDRESS FOR FIRST TEST
2654 006640 010500              MOV  R5,R0
2655 006642 062700 000006      ADD  #FC,R0           ;R0 CONTAINS ADDRESS OF FC REG
2656 006646 010501              MOV  R5,R1
2657 006650 062701 000012      ADD  #DS,R1           ;R1 CONTAINS ADDRESS OF DS REG
2658 006654 012703 004060      MOV  #TIMER,R3        ;SET JUMP ADDRESS TO TIMER
2659 006660 105737 001124      TSTB @#PRGFLG        ;BRANCH IF NOT SKEW TESTS
2660 006664 001402              BEQ  TST001
2661 006666 000137 012314      JMP  @#SKEWTST
2662      .SBTTL START OF TESTS
2663      ;TEST 001 - WRITE FROM BOT
2664      ;THIS TEST WILL MEASURE ACCELERATION DELAY REQUIRED TO
2665      ;MOVE THE TAPE APPROXIMATELY SEVEN (7) INCHES FORWARD
2666      ;FROM DEAD STOP BEFORE STARTING TO TRANSFER DATA.
2667

```

```

2668 ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2669 006672 112737 000001 001122 TST001: MOVB #1,@TSTNUM ;SET TEST #
2670 006700 012702 006724 MOV #1$,R2 ;SET RETURN PC FROM TIMER
2671 006704 004767 176042 JSR PC,.REWIND ;REWIND SLAVE
2672 006710 102420 BVS 99$ ;BRANCH IF ERROR ON REWIND
2673 006712 004767 176116 JSR PC,WRITE ;GO SETUP WRITE COMMAND
2674 006716 004767 175046 JSR PC,TIMON ;TURN TIMER ON
2675 006722 005215 INC (R5) ;SET 'GO' BIT
2676
2677 006724 005765 000032 1$: TST TC(R5) ;BRANCH WHEN 'ACCL'=0
2678 006730 100002 BPL 2$
2679 006732 000163 004060 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2680
2681 006736 004767 175704 2$: JSR PC,WAITRDY ;WAIT FOR COMMAND TO FINISH
2682 006742 102403 BVS 99$ ;BRANCH IF ERROR
2683 006744 004767 175146 JSR PC,TIMOK ;GO CHECK TIME
2684 006750 000401 BR 100$
2685 006752 104400 99$: HLT
2686 006754 104000 100$: SCOPE
2687
2688 ;TEST 002 - WRITE START
2689 ;THIS TST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2690 006756 112737 000002 001122 TST002: MOVB #2,@TSTNUM ;SET TEST # 2
2691 006764 004767 176044 JSR PC,WRITE ;INITIATE WRITE COMMAND
2692 006770 012702 007002 MOV #1$,R2 ;SET RETURN PC FROM TIMER
2693 006774 004767 174770 JSR PC,TIMON
2694 007000 005215 INC (R5) ;SET 'GO' BIT
2695
2696 007002 005765 000032 1$: TST TC(R5) ;BRANCH WHEN 'ACCL'=0
2697 007006 100002 BPL 2$
2698 007010 000163 004060 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2699
2700 007014 004767 175626 2$: JSR PC,WAITRDY ;WAIT FOR READY
2701 007020 102403 BVS 99$ ;BRANCH IF ERROR
2702 007022 004767 175070 JSR PC,TIMOK ;GO CHECK TIME RECORDED
2703 007026 000401 BR 100$ ;EXIT VIA SCOPE
2704
2705 007030 104400 99$: HLT ;REPORT ERROR
2706 007032 104000 100$: SCOPE
2707
2708 ;TEST 003- WRITE SHUTDOWN
2709 ;THIS TEST MEASURES TIME FROM 'FC REG'=0 TO 'SDWN'=1.
2710 007034 112737 000003 001122 TST003: MOVB #3,@TSTNUM ;SET TEST#3
2711 007042 004767 175766 JSR PC,WRITE ;INITIATE WRITE COMMAND
2712 007046 005215 INC (R5) ;SET 'GO' BIT
2713
2714 007050 005710 1$: TST (R0) ;BRANCH WHEN WRITING FINISHED
2715 007052 001404 BEQ 2$
2716 007054 032711 040000 BIT #ERR,(R1) ;MONITOR ERROR BIT
2717 007060 001017 BNE 99$
2718 007062 000772 BR 1$
2719
2720 007064 2$: JSR PC,TIMON ;TURN TIMER ON
2721 007064 004767 174700 MOV PC,R2 ;LOAD RETURN PC FROM TIMER
2722 007070 010702 3$: BIT #SDWN,(R1) ;BRANCH WHEN DS <SDWN> SETS
2723 007072 032711 000020

```

```

2724 007076 001002          BNE      4$
2725 007100 000163 004060    JMP      TIMER(R3)          ;GO TO TIMER & RETURN VIA R2
2726
2727 007104 004767 175536    4$:     JSR      PC,WAITRDY   ;WAIT FOR READY
2728 007110 102403          BVS      99$
2729 007112 004767 175000    JSR      PC,TIMOK          ;GO CHECK TIME RECORDED
2730 007116 000401          BR       100$
2731 007120 104400          99$:    HLT
2732 007122 104000          100$:   SCOPE              ;REPORT ERROR
2733
2734          ;TEST 004 - WRITE SETTLEDOWN
2735          ;THIS TEST MEASURES TIME FROM 'SWDN'=1 TO 'SWDN'=0.
2736 007124 112737 000004 001122 TST004: MOVB    #4,#TSTNUM
2737 007132 004767 175676    JSR      PC,WRITE
2738 007136 005215          INC      (R5)              ;SET 'GO' BIT
2739
2740 007140 005710          1$:     TST      (R0)        ;BRANCH WHEN WRITING FINISHED
2741 007142 001404          BEQ      2$
2742 007144 032711 040000    BIT      #ERR,(R1)        ;CHECK ERROR BIT
2743 007150 001026          BNE      99$
2744 007152 000772          BR       1$
2745
2746 007154 032711 000020    2$:     BIT      #SDWN,(R1)  ;WAIT FOR ASSERTION OF 'SDWN'
2747 007160 001004          BNE      3$
2748 007162 032711 040000    BIT      #ERR,(R1)        ;MONITOR ERROR BIT
2749 007166 001017          BNE      99$
2750 007170 000771          BR       2$
2751
2752          3$:
2753 007172 004767 174572    JSR      PC,TIMON         ;TURN TIMER ON
2754 007176 010702          MOV      PC,R2           ;SET RETURN PC FROM TIMER
2755 007200 032711 000020    BIT      #SDWN,(R1)      ;BRANCH WHEN SWDN CLEARS
2756 007204 001402          BEQ      5$
2757 007206 000163 004060    JMP      TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
2758
2759 007212 004767 175430    5$:     JSR      PC,WAITRDY   ;WAIT FOR READY
2760 007216 102403          BVS      99$
2761 007220 004767 174672    JSR      PC,TIMOK
2762 007224 000401          BR       100$
2763
2764 007226 104400          99$:    HLT
2765 007230 104000          100$:   SCOPE
2766
2767          ;TEST 005 - READ FROM BOT
2768          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2769 007232 112737 000005 001122 TST005: MOVB    #5,#TSTNUM  ;SET TEST #5
2770 007240 004767 175506    JSR      PC,.REWIND      ;REWIND SLAVE
2771 007244 102422          BVS      99$            ;BRANCH IF ERROR ON REWIND
2772 007246 004767 175600    JSR      PC,READ
2773 007252 012702 007264    MOV      #1$,R2         ;SET RETURN PC FROM TIMER
2774 007256 004767 174506    JSR      PC,TIMON         ;TURN TIMER ON
2775 007262 005215          INC      (R5)           ;SET 'GO' BIT
2776
2777 007264 005765 000032    1$:     TST      TC(R5)      ;BRANCH WHEN 'ACCL' RESETS
2778 007270 100002          BPL      2$
2779 007272 000163 004060    JMP      TIMER(R3)       ;GO TO TIMER & RETURN VIA R2

```

```

2780
2781 007276 004767 175344      2$:   JSR   PC, WAITRDY      ;WAIT FOR READY
2782 007302 102403              BVS   99$                 ;BRANCH IF ERROR
2783 007304 004767 174606      JSR   PC, TIMOK           ;CHECK RECORDED TIME
2784 007310 000401              BR    100$
2785
2786 007312 104400      99$:   HLT
2787 007314 104000      100$:  SCOPE
2788
2789      ;TEST 006 - READ START
2790      ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2791 007316 112737 000006 001122 TST006: MOVB   #6, @TSTNUM      ;SET TEST #6
2792 007324 004767 175576      JSR   PC, WRT.BK         ;WRITE A RECORD & BACK SPACE
2793 007330 102422              BVS   99$
2794 007332 004767 175514      JSR   PC, READ
2795 007336 012702 007350      MOV   #1$, R2            ;SET RETURN PC FROM TIMER
2796 007342 004767 174422      JSR   PC, TIMON         ;TURN TIMER ON
2797 007346 005215              INC   (R5)               ;SET 'GO' BIT
2798
2799 007350 005765 000032      1$:   TST   TC(R5)        ;BRANCH WHEN 'ACCL' RESETS
2800 007354 100002              BPL   2$
2801 007356 000163 004060      JMP   TIMER(R3)         ;GO TO TIMER & RETURN VIA R2
2802
2803 007362 004767 175260      2$:   JSR   PC, WAITRDY
2804 007366 102403              BVS   99$
2805 007370 004767 174522      JSR   PC, TIMOK
2806 007374 000401              BR    100$
2807
2808 007376 104400      99$:   HLT
2809 007400 104000      100$:  SCOPE
2810
2811      ;TEST 007 - READ SHUTDOWN
2812      ;THIS TEST MEASURES TIME FROM 'FC REG'=FRAME COUNT TO 'SWDN'=1.
2813 007402 112737 000007 001122 TST007: MOVB   #7, @TSTNUM      ;SET TEST #7
2814 007410 004767 175512      JSR   PC, WRT.BK         ;WRITE A RECORD & BACK SPACE
2815 007414 102430              BVS   99$                 ;BRANCH IF ERROR
2816 007416 004767 175430      JSR   PC, READ
2817 007422 005215              INC   (R5)               ;SET 'GO' BIT
2818
2819 007424 022710 000400      1$:   CMP   #-FRMCNT, (R0)    ;WAIT FOR FRAME COUNT TO
2820 007430 001404              BEQ   2$                 ;= # OF FRAMES WRITTEN
2821 007432 032711 040000      BIT   #ERR, (R1)        ;MONITOR ERROR BIT
2822 007436 001017              BNE   99$
2823 007440 000771              BR    1$
2824
2825 007442              2$:
2826 007442 004767 174322      JSR   PC, TIMON         ;TURN TIMER ON
2827 007446 010702              MOV   PC, R2            ;SET RETURN PC FROM TIMER
2828 007450 032711 000020      BIT   #SDWN, (R1)       ;BRANCH WHEN SDWN SETS
2829 007454 001002              BNE   3$
2830 007456 000163 004060      JMP   TIMER(R3)         ;GO TO TIMER & RETURN VIA R2
2831
2832 007462 004767 175160      3$:   JSR   PC, WAITRDY
2833 007466 102403              BVS   99$
2834 007470 004767 174422      JSR   PC, TIMOK
2835 007474 000401              BR    100$

```

```

2836
2837 007476 104400          99$:   HLT                               ;REPORT ERROR
2838 007500 104000          100$:  SCOPE
2839
2840                      ;TEST 010 - READ SETTLEDOWN
2841                      ;THIS TEST MEASURES TIME FROM 'SWDN'=1 TO 'SWDN'=0.
2842 007502 112737 000010 001122 TST010: MOVB   #10,#TSTNUM          ;SET TEST #10
2843 007510 012702 007566          MOV    #4$,R2             ;SET RETURN PC FROM TIMER
2844 007514 004767 175406          JSR   PC,WRT.BK         ;WRITE A RECORD & BACK SPACE
2845 007520 102436          BVS   99$
2846 007522 004767 175324          JSR   PC,READ
2847 007526 005215          INC   (R5)             ;SET 'GO' BIT
2848
2849 007530 105711          1$:   TSTB  (R1)          ;WAIT FOR READY
2850 007532 100404          BMI   2$              ;BRANCH WHEN SET
2851 007534 032711 040000          BIT   #ERR,(R1)       ;CHECK ERROR BIT
2852 007540 001026          BNE   99$
2853 007542 000772          BR    1$
2854
2855 007544 032711 000020          2$:   BIT   #SDWN,(R1)   ;WAIT FOR ASSERTION OF 'SDWN'
2856 007550 001004          BNE   3$
2857 007552 032711 040000          BIT   #ERR,(R1)       ;MONITOR ERROR BIT
2858 007556 001017          BNE   99$
2859 007560 000771          BR    2$
2860
2861 007562          3$:
2862 007562 004767 174202          JSR   PC,TIMON         ;TURN TIMER ON
2863 007566 032765 000020 000012 4$:   BIT   #SDWN,DS(R5)     ;WAIT FOR NEGATION OF SDWN
2864 007574 001402          BEQ   5$
2865 007576 000163 004060          JMP   TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
2866
2867 007602 004767 175040          5$:   JSR   PC,WAITRDY
2868 007606 102403          BVS   99$
2869 007610 004767 174302          JSR   PC,TIMOK
2870 007614 000401          BR    100$
2871
2872 007616 104400          99$:   HLT
2873 007620 104000          100$:  SCOPE
2874
2875
2876                      ;TEST 011-READ REVERSE START
2877                      ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2878 007622 112737 000011 001122 TST011: MOVB   #11,#TSTNUM          ;SET RETURN PC FROM TIMER
2879 007630 012702 007666          MOV    #1$,R2             ;WRITE A RECORD
2880 007634 004767 175174          JSR   PC,WRITE         ;SET 'GO' BIT
2881 007640 005215          INC   (R5)
2882 007642 004767 175000          JSR   PC,WAITRDY
2883 007646 102422          BVS   99$
2884 007650 004767 174500          JSR   PC,DELAY         ;WAIT FOR TAPE MOTION TO STOP
2885 007654 004767 175210          JSR   PC,REVRD
2886 007660 004767 174104          JSR   PC,TIMON         ;TURN TIMER ON
2887 007664 005215          INC   (R5)             ;SET 'GO' BIT
2888
2889 007666 005765 000032          1$:   TST   TC(R5)        ;BRANCH WHEN 'ACCL' = 0
2890 007672 100002          BPL   2$
2891 007674 000163 004060          JMP   TIMER(R3)       ;GO TO TIMER & RETURN VIA R2

```

```

2892
2893 007700 004767 174742      2$:   JSR   PC, WAITRDY
2894 007704 102403                BVS   99$           ;BRANCH IF ERROR
2895 007706 004767 174204      JSR   PC, TIMOK
2896 007712 000401                BR    100$
2897
2898 007714 104400      99$:   HLT
2899 007716 104000      100$:  SCOPE
2900
2901      ;TEST 012-READ REVERSE SHUTDOWN
2902      ;THIS TEST MEASURES TIME FROM 'FC REG' = FRAME COUNT TO 'SDWN'=1.
2903 007720 112737 000012 001122 TST012: MOVB  #12, @#TSTNUM
2904 007726 012702 007776      MOV   #3$, R2           ;SET RETURN PC FROM TIMER
2905 007732 004767 175076      JSR   PC, WRITE        ;WRITE A RECORD
2906 007736 005215                INC   (R5)             ;SET 'GO' BIT
2907 007740 004767 174702      JSR   PC, WAITRDY
2908 007744 102427                BVS   99$
2909 007746 004767 175116      JSR   PC, REVRD
2910 007752 005215                INC   (R5)             ;SET 'GO' BIT
2911
2912 007754 022710 000400      1$:   CMP   #-FRMCNT, (R0)   ;BRANCH WHEN FRAME COUNT
2913 007760 001404                BEQ   2$               ;= # OF RECORD WRITTEN
2914 007762 032711 040000      BIT   #ERR, (R1)      ;MONITOR ERROR BIT IN 'DS' REG
2915 007766 001016                BNE   99$
2916 007770 000771                BR    1$
2917
2918 007772      2$:
2919 007772 004767 173772      JSR   PC, TIMON        ;TURN TIMER ON
2920 007776 032711 000020      3$:   BIT   #SDWN, (R1)   ;BRANCH WHEN SDWN SETS
2921 010002 001002                BNE   4$
2922 010004 000163 004060      JMP   TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
2923
2924 010010 004767 174632      4$:   JSR   PC, WAITRDY   ;WAIT FOR READY
2925 010014 102403                BVS   99$
2926 010016 004767 174074      JSR   PC, TIMOK
2927 010022 000401                BR    100$
2928
2929 010024 104400      99$:   HLT
2930 010026 104000      100$:  SCOPE
2931
2932      ;TEST 013-READ REVERSE SETTLEDOWN
2933      ;THIS TEST MEASURES TIME FROM 'SDWN'=1 TO 'SDWN'=0.
2934 010030 112737 000013 001122 TST013: MOVB  #13, @#TSTNUM
2935 010036 012702 010122      MOV   #4$, R2           ;SET RETURN PC FROM TIMER
2936 010042 004767 174766      JSR   PC, WRITE        ;WRITE A RECORD
2937 010046 005215                INC   (R5)             ;SET 'GO' BIT
2938 010050 004767 174572      JSR   PC, WAITRDY
2939 010054 102435                BVS   99$
2940 010056 004767 175006      JSR   PC, REVRD
2941 010062 005215                INC   (R5)             ;SET 'GO' BIT
2942
2943 010064 105711      1$:   TSTB  (R1)           ;BRANCH WHEN
2944 010066 100404                BMI   2$               ;READY SETS
2945 010070 032711 040000      BIT   #ERR, (R1)
2946 010074 001025                BNE   99$
2947 010076 000772                BR    1$

```

```

2948
2949 010100 032711 000020 2$: BIT #SDWN,(R1)
2950 010104 001004 BNE 3$
2951 010106 032711 040000 BIT #ERR,(R1)
2952 010112 001016 BNE 99$
2953 010114 000771 BR 2$
2954
2955 010116 3$: JSR PC,TIMON ;TURN TIMER ON
2956 010116 004767 173646 4$: BIT #SDWN,(R1) ;BRANCH WHEN SWDN = 0
2957 010122 032711 000020 BEQ 5$
2958 010126 001402 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2959 010130 000163 004060
2960
2961 010134 004767 174506 5$: JSR PC,WAITRDY ;WAIT FOR READY
2962 010140 102403 BVS 99$
2963 010142 004767 173750 JSR PC,TIMOK
2964 010146 000401 BR 100$
2965
2966 010150 104400 99$: HLT
2967 010152 104000 100$: SCOPE
2968
2969 ;REWIND DRIVE
2970 010154 A:
2971 010154 004767 174572 JSR PC,.REWIND ;REWIND SLAVE
2972 010160 102401 BVS 99$ ;BRANCH IF ERROR ON REWIND
2973 010162 102002 BVC 100$
2974 010164 104400 99$: HLT
2975 010166 000772 BR A
2976 010170 100$:
2977
2978 ;TEST 014-TURN AROUND DELAY (FORWARD-REVERSE)
2979 ;THIS TEST MEASURES TIME FROM 'GO'=1 (READ REVERSE) TO 'ACCL'=0
2980 010170 112737 000014 001122 TST014: MOVB #14,#TSTNUM
2981 010176 012702 010230 MOV #2$,R2 ;SET RETURN PC FROM TIMER
2982 010202 004767 174626 JSR PC,WRITE ;WRITE A RECORD
2983 010206 005215 INC (R5) ;SET 'GO' BIT
2984 010210 004767 174432 JSR PC,WAITRDY
2985 010214 102420 BVS 99$
2986
2987 010216 004767 174646 1$: JSR PC,REVRD ;READ THE RECORD (REVERSE)
2988 010222 004767 173542 JSR PC,TIMON ;TURN TIMER ON
2989 010226 005215 INC (R5) ;SET 'GO' BIT
2990
2991 010230 005765 000032 2$: TST TC(R5) ;WAIT FOR 'ACCL' = 0
2992 010234 100002 BPL 3$
2993 010236 000163 004060 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2994
2995 010242 004767 174400 3$: JSR PC,WAITRDY
2996 010246 102403 BVS 99$
2997 010250 004767 173642 JSR PC,TIMOK
2998 010254 000401 BR 100$
2999
3000 010256 104400 99$: HLT
3001 010260 104000 100$: SCOPE
3002
3003 ;TEST 015- TURN AROUND DELAY (REVERSE-FORWARD)

```

```

3004 ;THIS TEST MEASURES TIME FROM 'GO'=1 (READ) TO 'ACCL'=0.
3005 010262 112737 000015 001122 TST015: MOVB #15,@#TSTNUM
3006 010270 012702 010336 MOV #2$,R2 ;SET RETURN PC FROM TIMER
3007 010274 004767 174534 JSR PC,WRITE ;WRITE A RECORD
3008 010300 005215 INC (R5) ;SET 'GO' BIT
3009 010302 004767 174340 JSR PC,WAITRDY ;WAIT FOR READY
3010 010306 102426 BVS 99$
3011 010310 004767 174554 JSR PC,REVRD ;READ A RECORD IN THE
3012 010314 005215 INC (R5) ;SET 'GO' BIT
3013
3014 010316 004767 174324 JSR PC,WAITRDY
3015 010322 102420 BVS 99$
3016
3017 010324 004767 174522 1$: JSR PC,READ ;READ RECORD FORWARD
3018 010330 004767 173434 JSR PC,TIMON ;TURN TIMER ON
3019 010334 005215 INC (R5) ;SET 'GO' BIT
3020
3021 010336 005765 000032 2$: TST TC(R5) ;WAIT FOR 'ACCL' = 0
3022 010342 100002 BPL 3$
3023 010344 000163 004060 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3024
3025 010350 004767 174272 3$: JSR PC,WAITRDY
3026 010354 102403 BVS 99$
3027 010356 004767 173534 JSR PC,TIMOK
3028 010362 000401 BR 100$
3029
3030 010364 104400 99$: HLT
3031 010366 104000 100$: SCOPE
3032
3033 ;TEST 016-GAP SIZE (STOP HALF)
3034 010370 112737 000016 001122 TST016: MOVB #16,@#TSTNUM
3035 010376 012702 010434 MOV #1$,R2 ;SET RETURN PC FROM TIMER
3036 010402 004767 174426 JSR PC,WRITE ;WRITE A RECORD
3037 010406 005215 INC (R5) ;SET 'GO' BIT
3038 010410 004767 174232 JSR PC,WAITRDY
3039 010414 102421 BVS 99$
3040 010416 004767 173732 JSR PC,DELAY ;DELAY 350 MS
3041 010422 004767 174442 JSR PC,REVRD ;READ REVERSE RECORD
3042 010426 004767 173336 JSR PC,TIMON ;TURN TIMER ON
3043 010432 005215 INC (R5) ;SET 'GO' BIT
3044
3045 010434 005710 1$: TST (R0) ;WAIT FOR FRAME COUNT > 0
3046 010436 001002 BNE 2$
3047 010440 000163 004060 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3048
3049 010444 004767 174176 2$: JSR PC,WAITRDY ;WAIT FOR READY BIT TO SET
3050 010450 102403 BVS 99$
3051 010452 004767 173440 JSR PC,TIMOK ;CHECK TIME
3052 010456 000401 BR 100$
3053
3054 010460 104400 99$: HLT
3055 010462 104000 100$: SCOPE
3056
3057 ;TEST 017-GAP SIZE (START HALF)
3058 010464 112737 000017 001122 TST017: MOVB #17,@#TSTNUM
3059 010472 012702 010544 MOV #1$,R2 ;SET RETURN PC FROM TIMER

```

```

3060 010476 004767 174332      JSR   PC,WRITE      ;WRITE A RECORD
3061 010502 005215              INC   (R5)          ;SET 'GO' BIT
3062 010504 004767 174136      JSR   PC,WAITRDY    ;WAIT FOR READY
3063 010510 102427              BVS   99$
3064 010512 004767 174352      JSR   PC,REVRD      ;READ REVERSE THE RECORD
3065 010516 005215              INC   (R5)          ;SET 'GO' BIT
3066 010520 004767 174122      JSR   PC,WAITRDY    ;WAIT FOR READY
3067 010524 102421              BVS   99$          ;BRANCH ON ERROR
3068 010526 004767 173622      JSR   PC,DELAY      ;WAIT FOR TAPE MOTION TO STOP
3069 010532 004767 174314      JSR   PC,READ       ;READ RECORD
3070 010536 004767 173226      JSR   PC,TIMON      ;TURN TIMER ON
3071 010542 005215              INC   (R5)          ;SET 'GO' BIT
3072
3073 010544 005710              1$:   TST   (R0)      ;WAIT FOR FRAME COUNT > 0
3074 010546 001002              BNE   2$
3075 010550 000163 004060      JMP   TIMER(R3)     ;GO TO TIMER & RETURN VIA R2
3076
3077 010554 004767 174066      2$:   JSR   PC,WAITRDY ;WAIT FOR READY
3078 010560 102403              BVS   99$
3079 010562 004767 173330      JSR   PC,TIMOK      ;CHECK TIME
3080 010566 000401              BR    100$
3081
3082 010570 104400              99$:  HLT
3083 010572 104000              100$: SCOPE
3084
3085 ;TEST 020- GAP SIZE (INTERRECORD)
3086 ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'FC REG' >0.
3087 010574 112737 000020 001122 TST020: MOVB  #20,#TSTNUM
3088 010602 012702 010664      MOV   #1$,R2        ;SET RETURN PC FROM TIMER
3089 010606 004767 174222      JSR   PC,WRITE      ;WRITE A RECORD
3090 010612 005215              INC   (R5)          ;SET 'GO' BIT
3091 010614 004767 174026      JSR   PC,WAITRDY    ;WAIT FOR READY
3092 010620 102433              BVS   99$
3093 010622 004767 174206      JSR   PC,WRITE      ;WRITE SECOND RECORD
3094 010626 005215              INC   (R5)          ;SET 'GO' BIT
3095 010630 004767 174012      JSR   PC,WAITRDY    ;WAIT FOR READY
3096 010634 102425              BVS   99$
3097 010636 004767 174226      JSR   PC,REVRD      ;READ REVERSE SECOND RECORD
3098 010642 005215              INC   (R5)          ;SET 'GO' BIT
3099 010644 004767 173776      JSR   PC,WAITRDY    ;WAIT FOR READY
3100 010650 102417              BVS   99$
3101 010652 004767 174212      JSR   PC,REVRD      ;READ REVERSE FIRST RECORD
3102 010656 004767 173106      JSR   PC,TIMON      ;TURN TIMER ON
3103 010662 005215              INC   (R5)          ;SET 'GO' BIT
3104
3105 010664 005710              1$:   TST   (R0)      ;WAIT FOR FRAME COUNT > 0
3106 010666 001002              BNE   2$
3107 010670 000163 004060      JMP   TIMER(R3)     ;GO TO TIMER & RETURN VIA R2
3108
3109 010674 004767 173746      2$:   JSR   PC,WAITRDY ;WAIT FOR READY
3110 010700 102403              BVS   99$
3111 010702 004767 173210      JSR   PC,TIMOK      ;CHECK TIME
3112 010706 000401              BR    100$
3113
3114 010710 104400              99$:  HLT
3115 010712 104000              100$: SCOPE

```

```

3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129 010714 112737 000021 001122 TST021: MOVB #21,@#TSTNUM
3130 010722 012702 011060 MOV #4$,R2 ;SET RETURN PC FROM TIMER
3131 010726 004767 174020 JSR PC,.REWIND ;REWIND SLAVE
3132 010732 102530 BVS 99$ ;BRANCH IF ERROR ON REWIND
3133 010734 005067 170154 CLR DELTIM ;CLEAR VARIABLE DELAY TIME
3134 010740 012700 000021 MOV #17.,R0 ;SET # OF RECORDS TO WRITE
3135 010744 004767 174064 1$: JSR PC,WRITE ;WRITE 17. RECORDS
3136 010750 005215 INC (R5) ;SET 'GO' BIT
3137 010752 004767 173670 JSR PC,WAITRDY ;WAIT FOR READY
3138 010756 102516 BVS 99$
3139 010760 004767 173420 JSR PC,DELAYV ;DELAY BEFORE WRITING,NEXT REC.
3140 010764 062767 000022 170122 ADD #18.,DELTIM ;SET NEXT DELAY TIME
3141 010772 005300 DEC R0 ;DECREMENT RECORDS WRITTEN COUNT
3142 010774 001363 BNE 1$
3143
3144 010776 012700 000021 MOV #17.,R0 ;SET # OF RECS. TO REVERSE READ
3145 011002 004767 174062 2$: JSR PC,REVRD ;REVERSE READ 17. RECORDS
3146 011006 005215 INC (R5) ;SET 'GO' BIT
3147 011010 004767 173632 JSR PC,WAITRDY ;WAIT FOR READY
3148 011014 102477 BVS 99$
3149 011016 005300 DEC R0 ;DECREMENT RECORD COUNT
3150 011020 001370 BNE 2$
3151
3152 011022 012700 000020 MOV #16.,R0 ;SET # OF RECORDS TO READ
3153 011026 012701 001054 MOV #GAPTBL,R1 ;SET PTR TO GAP TABLE FOR TEST
3154 011032 004767 174014 JSR PC,READ ;READ A RECORD
3155 011036 005215 INC (R5) ;SET 'GO' BIT
3156
3157 011040 004767 173602 3$: JSR PC,WAITRDY ;WAIT FOR READY
3158 011044 102463 BVS 99$
3159 011046 004767 174000 JSR PC,READ ;READ NEXT RECORD
3160 011052 004767 172712 JSR PC,TIMON ;TURN TIMER ON
3161 011056 005215 INC (R5) ;SET 'GO' BIT
3162
3163 011060 005765 000006 4$: TST FC(R5) ;WAIT FOR FRAME COUNT > 0
3164 011064 001002 BNE 5$
3165 011066 000163 004060 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3166
3167 011072 004767 173550 5$: JSR PC,WAITRDY ;WAIT FOR READY
3168 011076 102446 BVS 99$
3169 011100 010421 MOV R4,(R1)+ ;STORE TIME IN GAPTBL
3170 011102 005300 DEC R0 ;DECREMENT # OF RECORDS READ
3171 011104 001355 BNE 3$

```

```

: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.

```

```

3172
3173 011106 105037 001120          CLRB   @#GAP           ;SET GAP # 0
3174 011112 012700 000020          MOV    #16.,R0
3175 011116 012701 001054          MOV    #GAPTBL,R1
3176
3177 011122 012104          6$:   MOV    (R1)+,R4       ;GET GAP TICK COUNT
3178 011124 004767 173076          JSR    PC,GAPOK        ;CHECK TIME
3179 011130 105237 001120          INCB   @#GAP           ;INCREMENT GAP #
3180 011134 122737 000020 001120  CMPB   #16.,@#GAP      ;BRANCH IF ALL GAPS NOT CHECKED
3181 011142 001367
3182
3183 011144 012700 000020          MOV    #16.,R0         ;SETUP TO AVERAGE GAP SIZES
3184 011150 012701 001054          MOV    #GAPTBL,R1     ;SET PTR TO TABLE
3185 011154 005002          CLR    R2              ;CLEAR 'SUM' REGISTERS
3186 011156 005003          CLR    R3
3187 011160 062102          7$:   ADD    (R1)+,R2       ;ADD ALL GAP SIZES TOGETHER
3188 011162 005503          ADC    R3
3189 011164 005300          DEC    R0
3190 011166 001374          BNE    7$
3191 011170 012700 000004          MOV    #4,R0           ;NOW DIVIDE BY 16.
3192 011174 006203          8$:   ASR    R3              ;BY SHIFTING 4 PLACES RIGHT
3193 011176 006002          ROR    R2
3194 011200 005300          DEC    R0
3195 011202 001374          BNE    8$
3196 011204 010204          MOV    R2,R4           ;MOVE AVERAGED TIMES TO R4
3197 011206 004767 172704          JSR    PC,TIMOK        ;CHECK AVERAGED TIMES
3198 011212 000401          BR     100$
3199
3200 011214 104400          99$:  HLT
3201 011216 104000          100$: SCOPE
3202
3203          ;TEST 022-DUMMY TEST
3204          ;THIS TEST MEASURES NOTHING
3205 011220 112737 000022 001122' TST022: MOVB   #22,@#TSTNUM
3206
3207          ;TEST 023-DATA TIME (200BPI)
3208          ;THIS TEST MEASURES TIME FROM 'FC REG' CHANGES TO 'RDY'=1.
3209 011226 112737 000023 001122 TST023: MOVB   #23,@#TSTNUM
3210 011234 012702 011306          MOV    #3$,R2         ;SET RETURN PC FROM TIMER
3211 011240 004767 173506          JSR    PC,.REWIND     ;REWIND SLAVE
3212 011244 102437          BVS    99$            ;BRANCH IF ERROR ON REWIND
3213 011246 004367 173716          JSR    R3,TMCMO      ;WRITE 800 WORD RECORD
3214 011252 015176          .WORD  WTBUFF        ;SET WRITE BUFFER ADDRESS
3215 011254 176340          .WORD  -800.         ;WORD COUNT
3216 011256 174700          .WORD  -1600.        ;FRAME COUNT
3217 011260 000060          .WORD  WFWO          ;WRITE COMMAND
3218 011262 005215          INC    (R5)           ;SET 'GO' BIT
3219
3220 011264 022710 174700          1$:   CMP    #-1600.,(R0)   ;WAIT FOR FRAME COUNT TO CHANGE
3221 011270 001004          BNE    2$
3222 011272 032711 040000          BIT    #ERR,(R1)     ;MONITOR ERROR BIT
3223 011276 001022          BNE    99$
3224 011300 000771          BR     1$
3225
3226 011302          2$:   JSR    PC,TIMON      ;TURN TIMER ON
3227 011302 004767 172462

```

3228	011306	105711			3\$:	TSTB	(R1)		;WAIT FOR READY TO SET
3229	011310	100402				BMI	4\$		
3230	011312	000163	004060			JMP	TIMER(R3)		;GO TO TIMER & RETURN VIA R2
3231	011316	012700	000003		4\$:	MOV	#3,R0		;SET TO DIVIDE BY 8
3232	011322	006204			5\$:	ASR	R4		;BY SHIFTING RIGHT 3 PLACES
3233	011324	005300				DEC	R0		
3234	011326	001375				BNE	5\$		
3235	011330	004767	173312			JSR	PC,WAITRDY		
3236	011334	102403				BVS	99\$		
3237	011336	004767	172554			JSR	PC,TIMOK		;CHECK TIME
3238	011342	000401				BR	100\$		
3239									
3240	011344	104400			99\$:	HLT			
3241	011346	104000			100\$:	SCOPE			
3242									
3243									
3244	011350	112737	000024	001122					;TEST 024-DATA TIME (556BPI)
3245	011356	012702	011436		TST024:	MOVB	#24,@TSTNUM		
3246	011362	004767	173364			MOV	#3\$,R2		;SET RETURN PC FROM TIMER
3247	011366	102442				JSR	PC,.REWIND		;REWIND SLAVE
3248	011370	052765	000700	000032		BVS	99\$		;BRANCH IF ERROR ON REWIND
3249	011376	004367	173566			BIS	#BPI556+NORM11,TC(R5)		;LOAD TAPE CONTROL REGISTER
3250	011402	015176				JSR	R3,TMCMO		;WRITE 2224. WORD RECORD
3251	011404	173520				.WORD	WTBUF		
3252	011406	167240				.WORD	-2224.		
3253	011410	000060				.WORD	-4448.		
3254	011412	005215				.WORD	WFD		
3255						INC	(R5)		;SET 'GO' BIT
3256	011414	022710	167240		1\$:	CMP	#-4448.,(R0)		;BRANCH WHEN WRITING BEGINS
3257	011420	001004				BNE	2\$		
3258	011422	032711	040000			BIT	#ERR,(R1)		;MONITOR ERROR BIT
3259	011426	001022				BNE	99\$		
3260	011430	000771				BR	1\$		
3261									
3262	011432				2\$:				
3263	011432	004767	172332			JSR	PC,TIMON		;TURN TIMER ON
3264	011436	105711			3\$:	TSTB	(R1)		;BRANCH WHEN READY SETS
3265	011440	100402				BMI	4\$		
3266	011442	000163	004060			JMP	TIMER(R3)		;GO TO TIMER & RETURN VIA R2
3267									
3268	011446	012700	000003		4\$:	MOV	#3,R0		;SET SHIFT COUNT
3269	011452	006204			5\$:	ASR	R4		
3270	011454	005300				DEC	R0		
3271	011456	001375				BNE	5\$		
3272	011460	004767	173162			JSR	PC,WAITRDY		
3273	011464	102403				BVS	99\$		
3274	011466	004767	172424			JSR	PC,TIMOK		;CHECK TIME
3275	011472	000401				BR	100\$		
3276									
3277	011474	104400			99\$:	HLT			
3278	011476	104000			100\$:	SCOPE			
3279									
3280									
3281	011500	112737	000025	001122					;TEST 025-DATA TIME (800BPI)
3282	011506	012702	011566		TST025:	MOVB	#025,@TSTNUM		
3283	011512	004767	173234			MOV	#3\$,R2		;SET RETURN PC FROM TIMER
						JSR	PC,.REWIND		;REWIND SLAVE

3284	011516	102442			BVS	99\$		;BRANCH IF ERROR ON REWIND
3285	011520	052765	001300	000032	BIS	#BPI800+NORM11,TC(R5)		;SET 800 BPI
3286	011526	004367	173436		JSR	R3,TMCMD		;WRITE 3200. WORD RECORD
3287	011532	015176			.WORD	WTBUF		
3288	011534	171600			.WORD	-3200.		
3289	011536	163400			.WORD	-6400.		
3290	011540	000060			.WORD	WFWD		
3291	011542	005215			INC	(R5)		;SET 'GO' BIT
3292								
3293	011544	022710	163400		1\$:	CMP	#-6400.,(R0)	;WAIT FOR WRITING TO START
3294	011550	001004				BNE	2\$	
3295	011552	032711	040000			BIT	#ERR,(R1)	;MONITOR ERROR BIT
3296	011556	001022				BNE	99\$	
3297	011560	000771				BR	1\$	
3298								
3299	011562				2\$:			
3300	011562	004767	172202			JSR	PC,TIMON	;TURN TIMER ON
3301	011566	105711			3\$:	TSTB	(R1)	;BRANCH WHEN READY SETS
3302	011570	100402				BMI	4\$	
3303	011572	000163	004060			JMP	TIMER(R3)	;GO TO TIMER & RETURN VIA R2
3304								
3305	011576	012700	000003		4\$:	MOV	#3,R0	;SET SHIFT COUNT
3306	011602	006204			5\$:	ASR	R4	
3307	011604	005300				DEC	R0	
3308	011606	001375				BNE	5\$	
3309	011610	004767	173032			JSR	PC,WAITRDY	
3310	011614	102403				BVS	99\$	
3311	011616	004767	172274			JSR	PC,TIMOK	;CHECK TIME
3312	011622	000401				BR	100\$	
3313								
3314	011624	104400			99\$:	HLT		
3315	011626	104000			100\$:	SCOPE		
3316								
3317								
3318	011630	112737	000026	001122				
3319	011636	105737	001127					
3320	011642	001046						
3321	011644	012702	011724					
3322	011650	004767	173076					
3323	011654	102437						
3324	011656	052765	002300	000032				
3325	011664	004367	173300					
3326	011670	015176						
3327	011672	171600						
3328	011674	163400						
3329	011676	000060						
3330	011700	005215						
3331								
3332	011702	022710	163400		1\$:	CMP	#-6400.,(R0)	;BRANCH WHEN WRITING STARTS
3333	011706	001004				BNE	2\$	
3334	011710	032711	040000			BIT	#ERR,(R1)	;MONITOR ERROR BIT
3335	011714	001017				BNE	99\$	
3336	011716	000771				BR	1\$	
3337								
3338	011720				2\$:			
3339	011720	004767	172044			JSR	PC,TIMON	;TURN TIMER ON

```

3340 011724 105711          3$:   TSTB   (R1)           ;BRANCH WHEN READY SETS
3341 011726 100402          BMI    4$
3342 011730 000163 004060   JMP    TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
3343
3344 011734 006204          4$:   ASR    R4           ;DIVIDE TIME BY 4
3345 011736 006204          ASR    R4
3346 011740 004767 172702   JSR    PC, WAITRDY
3347 011744 102403          BVS    99$
3348 011746 004767 172144   JSR    PC, TIMOK       ;CHECK TIME
3349 011752 000401          BR     100$
3350
3351 011754 104400          99$:  HLT
3352 011756 104000          100$: SCOPE
3353
3354          ;TEST 027-ERASE
3355          ;THIS TST MEASURES TIME FROM 'GO'=1 TO 'RDY'=1.
3356 011760 112737 000027 001122 TST027: MOVB   #27, @#TSTNUM
3357 011766 012702 012014      MOV    #1$, R2           ;SET RETURN PC FROM TIMER
3358 011772 004337 005170      JSR    R3, @#TMCMD
3359 011776 000000          .WORD 0
3360 012000 000000          .WORD 0
3361 012002 000000          .WORD 0
3362 012004 000024          .WORD ERASE
3363 012006 004767 171756      JSR    PC, TIMON       ;TURN TIMER ON
3364 012012 005215          INC    (R5)           ;SET 'GO' BIT
3365
3366 012014 105711          1$:   TSTB   (R1)           ;BRANCH WHEN READY SETS
3367 012016 100402          BMI    2$
3368 012020 000163 004060   JMP    TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
3369
3370 012024 004767 172616      2$:   JSR    PC, WAITRDY
3371 012030 102403          BVS    99$
3372 012032 004767 172060      JSR    PC, TIMOK
3373 012036 000401          BR     100$
3374
3375 012040 104400          99$:  HLT
3376 012042 104000          100$: SCOPE
3377
3378          ;TEST-030 TAPE MARK
3379          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'RDY'=1.
3380 012044 112737 000030 001122 TST030: MOVB   #30, @#TSTNUM
3381 012052 012702 012114      MOV    #1$, R2           ;SET RETURN PC FROM TIMER
3382 012056 004767 172752      JSR    PC, WRITE       ;WRITE A RECORD
3383 012062 005215          INC    (R5)           ;SET 'GO' BIT
3384 012064 004767 172556      JSR    PC, WAITRDY
3385 012070 102423          BVS    99$
3386 012072 004337 005170      JSR    R3, @#TMCMD
3387 012076 000000          .WORD 0
3388 012100 000000          .WORD 0
3389 012102 000000          .WORD 0
3390 012104 000026          .WORD WFMK
3391 012106 004767 171656      JSR    PC, TIMON       ;TURN TIMER ON
3392 012112 005215          INC    (R5)           ;SET 'GO' BIT
3393
3394 012114 105711          1$:   TSTB   (R1)           ;BRANCH WHEN READY SETS
3395 012116 100402          BMI    2$

```

```

3396 012120 000163 004060          JMP     TIMER(R3)          ;GO TO TIMER & RETURN VIA R2
3397
3398 012124 004767 172516          2$:   JSR     PC, WAITRDY
3399 012130 102403                    BVS     99$
3400 012132 004767 171760          JSR     PC, TIMOK
3401 012136 000401                    BR      100$
3402
3403 012140 104400                    99$:   HLT
3404 012142                    100$:
3405 012142 004767 172604          JSR     PC, .REWIND        ;REWIND SLAVE
3406 012146 102774                    BVS     99$                ;BRANCH IF ERROR ON REWIND
3407 012150 104000                    SCOPE
3408
3409 012152 012700 000012          FINISH: MOV    #10., R0        ;SET LINE FEED COUNT
3410 012156 000004 001374          1$:   TYPE, CRLF
3411 012162 005300                    DEC     R0
3412 012164 001374                    BNE     1$
3413 012166 032777 000100 166604          BIT     #SW06, @SWR
3414 012174 001410                    BEQ     2$
3415 012176 113700 001004          MOVB   @#DRVNUM, R0
3416 012202 113701 001005          MOVB   @#SLVNUM, R1
3417 012206 113702 001006          MOVB   @#SLVPTR, R2
3418 012212 000137 006540          JMP     @#TYPHDR
3419 012216 105237 001005          2$:   INCB   @#SLVNUM        ;SET NEXT SLAVE #
3420 012222 005237 001006          INC     @#SLVPTR        ;AND ITS POINTER
3421 012226 122737 000010 001005          CMPB   #8., @#SLVNUM    ;BRANCH IF LAST SLAVE (7)
3422 012234 001402                    BEQ     3$
3423 012236 000137 006314          JMP     @#BEGIN        ;BEGIN TEST ON NEXT SLAVE
3424 012242 105037 001005          3$:   CLRB   @#SLVNUM        ;SET SLAVE #0
3425 012246 105237 001004          INCB   @#DRVNUM        ;AND INCREMENT DRIVE #
3426 012252 122737 000010 001004          CMPB   #8., @#DRVNUM    ;AND CHECK IF LAST DRIVE
3427 012260 001402                    BEQ     END
3428 012262 000137 006314          JMP     @#BEGIN
3429
3430 012266 105737 001125          END:   TSTB   @#UNTFND    ;BRANCH IF A UNIT WAS FOUND
3431 012272 001004                    BNE     1$
3432 012274 000004 013377          TYPE, E.UNIT
3433 012300 000137 005340          JMP     @#INIT
3434 012304 000000                    1$:   HALT
3435 012306 000005                    RESET
3436 012310 000137 005340          JMP     @#INIT        ;RESTART
3437
3438 ;SKEW TAPE TIMING TESTS
3439 012314 012737 012322 001002          SKWST:MOV  #TST031, @#SCPADR ;SET SCOPE POINTER
3440
3441 ;TEST 031- SKEW TAPE SPEED TEST-FORWARD
3442 ;THIS TEST READS 32" OF TAPE (26400.-800. = 25600. FRAMES), THEN
3443 ;DIVIDES TIME BY 32. TO GET TIME TO READ 1" (800. FRAMES) OF TAPE.
3444 012322 112737 000031 001122          TST031: MOVB  #31, @#TSTNUM
3445 012330 012702 012406          MOV     #2$, R2        ;SET RETURN PC FROM TIMER
3446 012334 004767 172412          JSR     PC, .REWIND    ;REWIND SLAVE
3447 012340 102441                    BVS     99$            ;BRANCH IF ERROR ON REWIND
3448 012342 052765 001300 000032          BIS     #BPI800+NORM11, TC(R5) ;SET 800 BPI
3449 012350 052765 000010 000010          BIS     #BAI, CS2(R5)  ;INHIBIT BUS ADDRESS INCREMENT
3450 012356 004337 005170          JSR     R3, @#TMCMD   ;READ 32" OF TAPE-FORWARD
3451 012362 015176          .WORD  RDBUF

```

```

3452 012364 177777
3453 012366 063440
3454 012370 000070
3455 012372 005215
3456
3457 012374 022710 001440
3458 012400 101375
3459
3460 012402 004767 171362
3461 012406 023710 012366
3462 012412 103402
3463 012414 000163 004060
3464
3465 012420 012700 000005
3466 012424 006204
3467 012426 005300
3468 012430 001375
3469 012432 004767 172152
3470 012436 004767 171454
3471 012442 000401
3472
3473 012444 104400
3474 012446 104000
3475
3476
3477
3478
3479
3480 012450 112737 000032 001122
3481 012456 012702 012604
3482 012467 004767 172264
3483 012466 102465
3484 012470 052765 001300 000032
3485 012476 052765 000010 000010
3486 012504 004337 005170
3487 012510 015176
3488 012512 177777
3489 012514 076400
3490 012516 000070
3491 012520 005215
3492
3493 012522 023710 012514
3494 012526 101375
3495
3496 012530 004767 172054
3497 012534 004767 171614
3498 012540 052765 001300 000032
3499 012546 052765 000010 000010
3500 012554 004337 005170
3501 012560 015176
3502 012562 177777
3503 012564 063440
3504 012566 000076
3505 012570 005215
3506
3507 012572 022710 001440

```

10\$: .WORD -1. ;FRAME COUNT  
.WORD 26400.  
.WORD RDEFWD ;SET 'GO' BIT  
INC (R5)  
1\$: CMP #800.,(R0) ;WAIT FOR FIRST 800 FRAMES  
BHI 1\$ ;TO BE READ  
2\$: JSR PC,TIMON ;TURN TIMER ON  
CMP @#10\$,(R0) ;WAIT FOR READING TO FINISH  
BLO 3\$  
JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2  
3\$: MOV #5,R0 ;DIVIDE TIME BY 32.  
4\$: ASR R4  
DEC R0  
BNE 4\$  
JSR PC,RHINIT ;INIT DRIVE  
JSR PC,TIMOK ;CHECK TIME  
BR 100\$  
99\$: HLT  
100\$: SCOPE  
;TEST 032-SKEW TAPE SPEED TEST-REVERSE  
;THIS TEST READS FORWARD 40" (32000. FRAMES) OF TAPE, THEN READS REVERSE  
;32" (26400.-800. = 25600. FRAMES) OF TAPE. THE TIME IS THEN DIVIDED BY  
;32. TO GET TIME TO READ 1" (800. FRAMES) OF TAPE.  
TST032: MOVB #32,@#TSTNUM  
MOV #3\$,R2 ;SET RETURN PC FROM TIMER  
JSR PC,.REWIND ;REWIND SLAVE  
BVS 99\$ ;BRANCH IF ERROR ON REWIND  
BIS #BPI800+NORM11,TC(R5)  
BIS #BAI,CS2(R5)  
JSR R3,@#TMCMD ;READ FORWARD 32000. FRAMES  
.WORD RDBUF  
.WORD -1. ;WORD COUNT  
10\$: .WORD 32000. ;FRAME COUNT  
.WORD RDEFWD ;READ FORWARD  
INC (R5) ;SET 'GO' BIT  
1\$: CMP @#10\$,(R0)  
BHI 1\$  
JSR PC,RHINIT ;INIT DRIVE  
JSR PC,DELAY ;WAIT FOR TAPE MOTION TO STOP  
BIS #BPI800+NORM11,TC(R5) ;SET 800 BPI  
BIS #BAI,CS2(R5) ;INHIBIT BUS ADDRESS INCREMENT  
JSR R3,@#TMCMD ;READ REVERSE 32" OF TAPE  
.WORD RDBUF ;READ BUFFER  
.WORD -1. ;WORD COUNT  
11\$: .WORD 26400. ;FRAME COUNT  
.WORD RDREV ;READ REVERSE  
INC (R5) ;SET 'GO' BIT  
2\$: CMP ,#800.,(R0) ;WAIT FOR FIRST 800 FRAMES

```

3508 012576 101375          BHI      2$          ;TO BE READ
3509
3510 012600 004767 171164    JSR      PC,TIMON    ;TURN TIMER ON
3511 012604 023710 012564    3$:     CMP      @#11$, (R0) ;WAIT FOR ALL FRAMES TO BE READ
3512 012610 103402          BLO      4$
3513 012612 000163 004060    JMP      TIMER(R3)   ;GO TO TIMER & RETURN VIA R2
3514
3515 012616 012700 000005    4$:     MOV      #5,R0   ;DIVIDE TIME BY 32.
3516 012620 006204    5$:     ASR      R4
3517 012624 005300          DEC      R0
3518 012626 001375          BNE      5$
3519 012630 004767 171754    JSR      PC,RHINIT
3520 012634 004767 171256    JSR      PC,TIMOK
3521 012640 000401          BR       100$
3522
3523 012642 104400    99$:    HLT
3524 012644    100$:
3525 012644 004767 172102    JSR      PC,.REWIND ;REWIND SLAVE
3526 012650 102774          BVS     99$         ;BRANCH IF ERROR ON REWIND
3527 012652 104000          SCOPE
3528
3529 012654 000137 012152    JMP      @#FINISH
3530
3531
3532
3533
3534          .SBTTL   PROGRAM MESSAGES
3535 012660 005015 052524 032464    ;OPERATOR INSTRUCTIONS
3536 012666 042040 044522 042526    M.NAM:  .ASCIZ  <CR><LF>'TU45 DRIVE FUNCTION TIMER (CZTULAO) '
3537 012674 043040 047125 052103
3538 012702 047511 020116 044524
3539 012710 042515 020122 041450
3540 012716 052132 046125 030101
3541 012724 000051
3542 012726 005015 054524 042520    I.REG:  .ASCIZ  <CR><LF>'TYPE FIRST ADDRESS OF CONTROLLER '
3543 012734 043040 051111 052123
3544 012742 040440 042104 042522
3545 012750 051523 047440 020106
3546 012756 047503 052116 047522
3547 012764 046114 051105 020040
3548 012772 000
3549 012773 0124 050131 020105    I.DRVS: .ASCIZ  %TYPE TMO2 DRIVE #'S TO BE TESTED %
3550 013000 046524 031060 042040
3551 013006 044522 042526 021440
3552 013014 051447 052040 020117
3553 013022 042502 052040 051505
3554 013030 042524 020104 000
3555 013035 0106 051117 052040    I.SLVS: .ASCII  'FOR TMO2 DRIVE '
3556 013042 030115 020062 051104
3557 013050 053111 020105
3558 013054 026460 052040 050131    I.DRV:  .ASCIZ  %0- TYPE SLAVE #'S TO BE TESTED %
3559 013062 020105 046123 053101
3560 013070 020105 023443 020123
3561 013076 047524 041040 020105
3562 013104 042524 052123 042105
3563 013112 000040

```

3564	013114	050123	042505	020104	I.SKEW: .ASCIZ 'SPEED TESTS ONLY? (YES/NO = 1/0)'
3565	013122	042524	052123	020123	
3566	013130	047117	054514	020077	
3567	013136	054450	051505	047057	
3568	013144	020117	020075	027461	
3569	013152	024460	000		
3570	013155	116	055122	047440	I.NRZ: .ASCIZ 'NRZ ONLY? (YES/NO = 1/0)'
3571	013162	046116	037531	024040	
3572	013170	042531	027523	047516	
3573	013176	036440	030440	030057	
3574	013204	000051			
3575	013206	005015	047105	020104	M.EOT: .ASCIZ <CR><LF>'END OF TAPE'<CR><LF>
3576	013214	043117	052040	050101	
3577	013222	006505	000012		
3578					
3579					:ERROR MESSAGES
3580	013226	005015	051124	050101	E.TRP4: .ASCIZ <CR><LF>'TRAPPED TO 4'
3581	013234	042520	020104	047524	
3582	013242	032040	000		
3583	013245	116	020117	047503	E.NCON: .ASCIZ 'NO CONTROLLER AT ADDRESS SPECIFIED'<CR><LF>
3584	013252	052116	047522	046114	
3585	013260	051105	040440	020124	
3586	013266	042101	051104	051505	
3587	013274	020123	050123	041505	
3588	013302	043111	042511	006504	
3589	013310	000012			
3590	013312	046524	031060	042040	E.NDRV: .ASCIZ 'TMO2 DRIVE '
3591	013320	044522	042526	000040	
3592	013326	051104	053111	020105	E.NSLV: .ASCII 'DRIVE '
3593	013334	020060	046123	053101	E.DRV: .ASCII '0 SLAVE '
3594	013342	020105			
3595	013344	020060	047516	020124	E.NAVA: .ASCIZ '0 NOT AVAILABLE FOR TEST'<CR><LF>
3596	013352	053101	044501	040514	
3597	013360	046102	020105	047506	
3598	013366	020122	042524	052123	
3599	013374	005015	000		
3600	013377	116	020117	046524	E.UNIT: .ASCIZ 'NO TMO2/TU45 UNIT FOUND TO TEST'<CR><LF>
3601	013404	031060	052057	032125	
3602	013412	020065	047125	052111	
3603	013420	043040	052517	042116	
3604	013426	052040	020117	042524	
3605	013434	052123	005015	000	
3606	013441	123	043117	020124	E.SFT: .ASCIZ 'SOFT ERROR (DATA)'<CR><LF>
3607	013446	051105	047522	020122	
3608	013454	042050	052101	024501	
3609	013462	005015	000		
3610	013465	124	051505	020124	E.HDR: .ASCIZ 'TEST # '
3611	013472	020043	000		
3612	013475	040	042504	044526	E.HDR1: .ASCII ' DEVICE ERROR'<CR><LF>
3613	013502	042503	042440	051122	
3614	013510	051117	005015		
3615	013514	051503	004461	041527	.ASCIZ 'CS1'<HT>'WC'<HT>'BA'<HT>'FC'<HT>'CS2'<HT>'DS'<HT>'ER'<HT>'TC'<CR><LF>
3616	013522	041011	004501	041506	
3617	013530	041411	031123	042011	
3618	013536	004523	051105	052011	
3619	013544	006503	000012		

3620	013550	047440	052125	047440	E.HDR2: .ASCIZ ' OUT OF RANGE ERROR'<CR><LF>
3621	013556	020106	040522	043516	
3622	013564	020105	051105	047522	
3623	013572	006522	000012		
3624	013576	005015	044524	042515	E.TIMOV: .ASCIZ <CR><LF>'TIMER OVERFLOWED'<CR><LF>
3625	013604	020122	053117	051105	
3626	013612	046106	053517	042105	
3627	013620	005015	000		
3628	013623	015	052012	046511	E.TIMEX: .ASCIZ <CR><LF>'TIME EXPIRED WAITING FOR RDY'<CR><LF>
3629	013630	020105	054105	044520	
3630	013636	042522	020104	040527	
3631	013644	052111	047111	020107	
3632	013652	047506	020122	042122	
3633	013660	006531	000012		
3634	013664	043440	050101	021440	E.GAP: .ASCIZ ' GAP # '
3635	013672	000040			
3636					
3637					:TIME DOCUMENT LINES
3638	013674	025052	025052	025052	L.HDR1: .ASCIZ '*****'
3639	013702	025052	025052	025052	
3640	013710	025052	025052	025052	
3641	013716	025052	025052	025052	
3642	013724	025052	025052	025052	
3643	013732	025052	025052	025052	
3644	013740	025052	025052	025052	
3645	013746	025052	025052	025052	
3646	013754	025052	025052	025052	
3647	013762	025052	025052	025052	
3648	013770	025052	025052	025052	
3649	013776	025052	025052	025052	
3650	014004	005015	000		
3651	014007	052	052040	030115	L.HDR2: .ASCII '* TMO2 DRIVE FUNCTION TIMES- DRIVE # '
3652	014014	020062	051104	053111	
3653	014022	020105	052506	041516	
3654	014030	044524	047117	052040	
3655	014036	046511	051505	020055	
3656	014044	051104	053111	020105	
3657	014052	020043			
3658	014054	020060	046123	053101	L.DRV: .ASCII '0 SLAVE # '
3659	014062	020105	020043		
3660	014066	020060	040		L.SLV: .ASCII '0 '
3661	014071	071	041440	040510	L.CHAN: .ASCIZ '9 CHAN. SER # '
3662	014076	027116	051440	051105	
3663	014104	021440	000040		
3664	014110	006440	025012	005015	L.HDR3: .ASCII ' '<CR><LF>' '<CR><LF>
3665	014116	020052	052506	041516	.ASCIZ '* FUNCTION'<HT><HT>'TIME(SPECIFICATION)'<HT>'TIME(ACTUAL)'<CR><LF>
3666	014124	044524	047117	004411	
3667	014132	044524	042515	051450	
3668	014140	042520	044503	044506	
3669	014146	040503	044524	047117	
3670	014154	004451	044524	042515	
3671	014162	040450	052103	040525	
3672	014170	024514	005015	000	
3673					
3674	014175	122	047101	042507	L.RNG: .ASCIZ 'RANGE=<
3675	014202	036075	000		

3676	014205	101	052103	040525	L.ACT: .ASCIZ 'ACTUAL='
3677	014212	036514	000		
3678					
3679					;TEST DESCRIPTOR HEADERS
3680	014215	052	053440	044522	A.T001: .ASCIZ '* WRITE FROM BOT'<HT>
3681	014222	042524	043040	047522	
3682	014230	020115	047502	004524	
3683	014236	000			
3684	014237	052	053440	044522	A.T002: .ASCIZ '* WRITE START'<HT><HT>
3685	014244	042524	051440	040524	
3686	014252	052122	004411	000	
3687	014257	052	053440	044522	A.T003: .ASCIZ '* WRITE SHUTDOWN'<HT>
3688	014264	042524	051440	052510	
3689	014272	042124	053517	004516	
3690	014300	000			
3691	014301	052	053440	044522	A.T004: .ASCIZ '* WRITE SETTLEDOWN'<HT>
3692	014306	042524	051440	052105	
3693	014314	046124	042105	053517	
3694	014322	004516	000		
3695	014325	052	051040	040505	A.T005: .ASCIZ '* READ FROM BOT'<HT><HT>
3696	014332	020104	051106	046517	
3697	014340	041040	052117	004411	
3698	014346	000			
3699	014347	052	051040	040505	A.T006: .ASCIZ '* READ START'<HT><HT>
3700	014354	020104	052123	051101	
3701	014362	004524	000011		
3702	014366	020052	042522	042101	A.T007: .ASCIZ '* READ SHUTDOWN'<HT><HT>
3703	014374	051440	052510	042124	
3704	014402	053517	004516	000011	
3705	014410	020052	042522	042101	A.T010: .ASCIZ '* READ SETTLEDOWN'<HT>
3706	014416	051440	052105	046124	
3707	014424	042105	053517	004516	
3708	014432	000			
3709	014433	052	051040	040505	A.T011: .ASCIZ '* READ REV START'<HT>
3710	014440	020104	042522	020126	
3711	014446	052123	051101	004524	
3712	014454	000			
3713	014455	052	051040	040505	A.T012: .ASCIZ '* READ REV SHUTDOWN'<HT>
3714	014462	020104	042522	020126	
3715	014470	044123	052125	047504	
3716	014476	047127	000011		
3717	014502	020052	042522	042101	A.T013: .ASCIZ '* READ REV SETTLEDOWN'<HT>
3718	014510	051040	053105	051440	
3719	014516	052105	046124	042105	
3720	014524	053517	004516	000	
3721	014531	052	052040	051125	A.T014: .ASCIZ '* TURN AROUND DELAY F-R'<HT>
3722	014536	020116	051101	052517	
3723	014544	042116	042040	046105	
3724	014552	054501	043040	051055	
3725	014560	000011			
3726	014562	020052	052524	047122	A.T015: .ASCIZ '* TURN AROUND DELAY R-F'<HT>
3727	014570	040440	047522	047125	
3728	014576	020104	042504	040514	
3729	014604	020131	026522	004506	
3730	014612	000			
3731	014613	052	043440	050101	A.T016: .ASCIZ '* GAP SIZE-STOP HALF'<HT>

3732	014620	051440	055111	026505	
3733	014626	052123	050117	044040	
3734	014634	046101	004506	000	
3735	014641	052	043440	050101	A.T017: .ASCIZ '* GAP SIZE-START HALF'<HT>
3736	014646	051440	055111	026505	
3737	014654	052123	051101	020124	
3738	014662	040510	043114	000011	
3739	014670	020052	040507	020120	A.T020: .ASCIZ '* GAP SIZE-INTERRECORD'<HT>
3740	014676	044523	042532	044455	
3741	014704	052116	051105	042522	
3742	014712	047503	042122	000011	
3743	014720	020052	040507	020120	A.T021: .ASCIZ '* GAP CONSISTANCY'<HT>
3744	014726	047503	051516	051511	
3745	014734	040524	041516	004531	
3746	014742	000			
3747	014743	052	042040	052101	A.T023: .ASCIZ '* DATA TIME-200BPI'<HT>
3748	014750	020101	044524	042515	
3749	014756	031055	030060	050102	
3750	014764	004511	000		
3751	014767	052	042040	052101	A.T024: .ASCIZ '* DATA TIME-556BPI'<HT>
3752	014774	020101	044524	042515	
3753	015002	032455	033065	050102	
3754	015010	004511	000		
3755	015013	052	042040	052101	A.T025: .ASCIZ '* DATA TIME-800BPI'<HT>
3756	015020	020101	044524	042515	
3757	015026	034055	030060	050102	
3758	015034	004511	000		
3759	015037	052	042040	052101	A.T026: .ASCIZ '* DATA TIME-1600BPI'<HT>
3760	015044	020101	044524	042515	
3761	015052	030455	030066	041060	
3762	015060	044520	000011		
3763	015064	020052	051105	051501	A.T027: .ASCIZ '* ERASE GAP TIME'<HT>
3764	015072	020105	040507	020120	
3765	015100	044524	042515	000011	
3766	015106	020052	051127	052111	A.T030: .ASCIZ '* WRITE FILE MARK'<HT>
3767	015114	020105	044506	042514	
3768	015122	046440	051101	004513	
3769	015130	000			
3770	015131	052	052040	050101	A.T031: .ASCIZ '* TAPE SPEED-FWD'<HT>
3771	015136	020105	050123	042505	
3772	015144	026504	053506	004504	
3773	015152	000			
3774	015153	052	052040	050101	A.T032: .ASCIZ '* TAPE SPEED-REV'<HT>
3775	015160	020105	050123	042505	
3776	015166	026504	042522	004526	
3777	015174	000			
3778		015176			
3779		015176			
3780		015176			
3781	015176	000200			
3782		000001			

.EVEN  
RDBUF=  
WTBUF=  
.BLKW 128.  
.END

;THAT'S ALL FOLKS!

A	010154	CNVDEC	002362	E.HDR2	013550	L.SLV	014066	R11	=X000001
ACCL	= 100000	CNV OCT	002254	E.NAVA	013344	MCPE	= 020000	R12	=X000002
ANGTAB	001412	CNVTAO	002712	E.NCON	013245	MDPE	= 000400	R13	=X000003
AS	= 000016	CNVTD	002374	E.NDRV	013312	MMVEC	= 000250	R14	=X000004
ASFLG	001130	CNVTO	002266	E.NSLV	013326	MOL	= 010000	R15	=X000005
ATA	= 100000	CR	= 000015	E.SFT	013441	MR	= 000024	SC	= 100000
ATIME	001012	CRLF	001374	E.TIME	013623	MXF	= 001000	SCOPE	= 104000
ATIMTB	001014	CSITM	= 002000	E.TIMO	013576	M.EOT	013206	SCPADR	001002
A.T001	014215	CS1	= 000000	E.TRP4	013226	M.NAM	012660	SDWN	= 000020
A.T002	014237	CS2	= 000010	E.UNIT	013377	NAMPTR	001672	SKEWTS	012314
A.T003	014257	DASH	001405	FC	= 000006	NED	= 010000	SLA	= 000001
A.T004	014301	DB	= 000022	FCE	= 001000	NEF	= 004000	SLAVES	005720
A.T005	014325	DCONST	002470	FINISH	012152	NEM	= 004000	SLR	= 177774
A.T006	014347	DELAY	004354	FMT	= 000020	NOP	= 000000	SLVAVA	004560
A.T007	014366	DELAYV	004404	FPEVEC	= 000244	NORM11	= 000300	SLVNUM	001005
A.T010	014410	DELTIM	001114	FRMCNT	= 177400	NRZFLG	001127	SLVPTR	001006
A.T011	014433	DIGTAB	001132	FWDSPC	005106	MSG	= 000400	SLVTBL	001164
A.T012	014455	DISPLA	= 177570	GAP	001120	OCTALO	001116	SN	= 000030
A.T013	014502	DIVIDE	004442	GAPOK	004226	ODIGIT	001144	SNPT	005210
A.T014	014531	DLT	= 100000	GAPTBL	001054	OPI	= 020000	SPACE	001410
A.T015	014562	DPR	= 000400	GO	= 000001	OR	= 000200	SPACE2	001407
A.T016	014613	DRIVES	005470	GTIMTB	001572	OSC	= 000100	SPCFWD	= 000030
A.T017	014641	DRVAVA	004532	HLT	= 104400	OUTBUF	= 005340	SPCREV	= 000032
A.T020	014670	DRVNUM	001004	HSWR	= 177570	OUTGAP	002600	SPR	= 002000
A.T021	014720	DRVTBL	001154	HT	= 000011	OUTSPC	002504	SSC	= 000100
A.T023	014743	DRY	= 000200	IDB	= 000010	PARVEC	= 000114	STIMTB	001416
A.T024	014767	DRYCLR	= 000010	IE	= 000100	PAT	= 000020	STKPTR	= 000600
A.T025	015013	DS	= 000012	ILF	= 000001	PEFLRC	= 000200	SWR	001000
A.T026	015037	DT	= 000026	ILR	= 000002	PES	= 000040	SW06	= 000100
A.T027	015064	LTE	= 010000	INBUF	001264	PE1600	= 002000	SW07	= 000200
A.T030	015106	DVA	= 004000	INCVAE	= 000100	PFVEC	= 000024	SW08	= 000400
A.T031	015131	DVO	= 000000	INIT	005340	PGE	= 002000	SW09	= 001000
A.T032	015153	DV1	= 000001	IOTVEC	= 000020	PIP	= 020000	SW10	= 002000
A16	= 000400	DV2	= 000002	IR	= 000100	PIRQ	= 177772	SW11	= 004000
A17	= 001000	DV3	= 000003	ITCNT	001121	PIRVEC	= 000240	SW13	= 020000
BA	= 000004	DV4	= 000004	I.DRV	013054	PLKCSR	= 172540	SW14	= 040000
BA1	= 000010	DV5	= 000005	I.DRVS	012773	PLKVEC	= 000104	SW15	= 100000
BEGIN	006314	DV6	= 000006	I.NRZ	013155	PRGFLG	001124	TAP	= 040000
BELL	001403	DV7	= 000007	I.REG	012726	PSEL	= 002000	TBITVE	= 000014
BKSLSH	001377	ECHO	001401	I.SKEW	013114	PSW	= 177776	TC	= 000032
BOT	= 000002	EMTVEC	= 000030	I.SLVS	013035	PUBLIS	003000	TIMER	004060
BPI200	= 000000	END	012266	LF	= 000012	RDBUF	= 015176	TIMERR	004070
BPI556	= 000400	EOT	= 002000	LKS	= 177546	RDFWD	= 000070	TIMERO	004104
BPI800	= 001000	ER	= 000014	LKVEC	= 000100	RDREV	= 000076	TIMER1	004034
BPTVEC	= 000014	ERASE	= 000024	LPB	= 177516	RDY	= 000200	TIMOK	004116
CDM11	= 000320	ERFLG	001123	LPS	= 177514	READ	005052	TIMON	003770
CHKDRV	005620	ERR	= 040000	L.ACT	014205	RESVEC	= 000010	TKB	= 177562
CHKSLV	006124	ERRTRP	003302	L.CHAN	014071	REVRD	005070	TKISR	003240
CH7	= 010000	ERRVEC	= 000004	L.DRV	014054	RHINIT	004610	TKS	= 177560
CLR	= 000040	E.DRV	013334	L.HDR1	013674	RMR	= 000004	TKVEC	= 000060
CNTRLC	= 000003	E.GAP	013664	L.HDR2	014007	RWD	= 000006	TMBASE	001010
CNTRLO	= 000017	E.HDR	013465	L.HDR3	014110	RWDOFF	= 000002	TMCMO	005170
CNTRLU	= 000025	E.HDR1	013475	L.RNG	014175	R10	=X000000	TMCS1	= 172440

TMK = 000004	TST010 007502	TST027 011760	WC = 000002	SHT = 000011
TPB = 177566	TST011 007622	TST030 012044	WCE = 040000	\$NULL 001760
TPS = 177564	TST012 007720	TST031 012322	WCHKF = 000050	\$TKFLG 001763
TPVEC = 000064	TST013 010030	TST032 012450	WCHKR = 000056	\$TPB 001766
TRAPVE = 000034	TST014 010170	TYPDEC 002370	WFMK = 000026	\$TPFLG 001762
TRE = 040000	TST015 010262	TYPE = 000004	WFWD = 000060	\$TPS 001764
TRTVEC = 000014	TST016 010370	TYPFLG 001126	WRDCNT = 177600	. = 015576
TSTNUM 001122	TST017 010464	TYPHDR 006540	WRITE 005034	.HLT 003304
TST001 006672	TST020 010574	TYPOCT 002262	WRL = 004000	.INPUT 003126
TST002 006756	TST021 010714	UBREAK = 177770	WRT.BK 005126	.RESTO 002232
TST003 007034	TST022 011220	UNS = 040000	WTBUF = 015176	.REWIND 004752
TST004 007124	TST023 011226	UNTFND 001125	\$CHARC 001770	.SAVE 002210
TST005 007232	TST024 011350	UPE = 020000	\$CNTRL 001771	.SCOPE 003552
TST006 007316	TST025 011500	WAITRD 004646	\$CRLF 001772	.TYPE 001776
TST007 007402	TST026 011630	WAITTI 004650	\$FILL 001761	

. ABS. 015576 000

ERRORS DETECTED: 0

.CZTULA.SEQ/SOL\_CZTULA.P11  
RUN-TIME: 18 37.2 SECONDS  
RUN-TIME RATIO: 92/58=1.5  
CORE USED: 5K (10 PAGES)