

TM03

TM03/TE16,TU77 DFT
CZTEECO

AH-A804C-MC

COPYRIGHT 77-80

FICHE 1 OF 1

JAN 1980

digital

MADE IN USA

The image shows a grid of 10 columns and 10 rows of small, illegible data tables or charts. Each cell in the grid contains a small table with multiple columns and rows of text, which is too small to read. The grid is located on the left side of the page, and the right side is mostly blank.

.RFM %

IDENTIFICATION

PRODUCT CODE: AC-A803C-MC
PRODUCT NAME: CZTEECO TMO3-TE16/TU77 DRIVE FUNCTION TIMER
DATE CREATED: 20 MAR 79
MAINTAINER: DIAGNOSTIC GROUP
AUTHOR: J. G. ADAMS

REVISED TO REV C MIKE PAGE
20 MAR 79
;+C SHOWS CODE ADDED TO REV C.

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

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

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

COPYRIGHT (C) 1977, 1979 BY DIGITAL EQUIPMENT CORPORATION

TMO3 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 SUBTEST DESCRIPTIONS

TMO3 DRIVE FUNCTION TIMER
ABSTRACT

PAGE 3

ABSTRACT

PROGRAM DZTEE MEASURES THE TIME REQUIRED AND GAP
SIZES PRODUCED BY THE TMO3-TE16/TU77 MAGTAPE
DRIVE/SLAVE.

THE TEST WILL CHECK BOTH THE LOGIC GENERATED
TIME DELAYS, AND THE DISTANCES TRAVELED BY THE
TAPE.

ACTUAL TAPE SPEED MAY ALSO BE CHECKED BY USING
THE SPEED TESTS WITH AN 300 BPI SKFW TAPE.

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

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

TM03 DRIVE FUNCTION TIMER
REQUIREMENTS

PAGE 4

CHAPTER 1
REQUIREMENTS

PDP-11 FAMILY CENTRAL PROCESSOR WITH 4K MEMORY WITH UP TO 64 TM03-TE16/TU77
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)

MAINDEC-11-DZTEA CONTROL LOGIC TEST(PART 1)
MAINDEC-11-DZTEC BASIC FUNCTION TEST

TMO3 DRIVE FUNCTION TIMER
LOADING AND STARTING PROCEDURE

PAGE 5

CHAPTER 2

LOADING AND STARTING PROCEDURE

2.0 LOAD & START PROCEDURE:

LOAD PROGRAM USING THE ABSOLUTE LOADER
LOAD ADDRESS = 200
SET OPERATING SWITCHES SEE CHAPT 3 SWITCH SETTINGS
PRESS START

THE PROGRAM WILL THEN REQUEST THE FIRST BUS ADDRESS OF THE RHXX CONTROLLER, TMO3 DRIVES TO BE TESTED, TE16/TU77 SLAVES TO BE TESTED, AND IF SPEED TESTS ARE TO BE RUN. IN ADDITON TO EACH REQUEST A DEFAULT ANSWER WILL BE TYPED. TO INVOKE THE DEFAULT TYPE A CARRIAGE RETURN.

THE REQUESTS & THEIR DEFAULTS ARE:

TYPE FIRST ADDRESS OF CONTROLLER:172440
TYPE TMO3 DRIVE #'S TO BE TESTED:ALL
FOR TMO3 DRIVE X-TYPE SLAVE #'S TO BE TESTED:ALL
SPEED TESTS?(YES/NO):NO

NOTES: SLAVE #'S ARE NOT REQUESTED IF DEFAULT TO DRIVE REQUEST IS INVOKED.

IF MORE THAN 1 DRIVE OR SLAVE IS TO BE TESTED, TYPE ', ' BETWEEN EACH DRIVE OR SLAVE # TO BE TESTED.

SPEED TESTS CAN & WILL ONLY BE RUN BY ANSWERING YES TO THE REQUEST.

TYPE CONTROL U (^U) TO DELETE LINE TYPED;TYPE 'RUBOUT' TO DELETE LAST CHARACTER(S).
PROGRAM WILL REPORT ERRORS, AND END OF PASS.

2.1 RESTART PROCEDURE

THE PROGRAM MAY BE RESTARTED USING START UP PARAMETERS AT ADDRESS 210.

THE PROGRAM MAY ALSO BE RESTARTED BY TYPING A CONTROL C (^C).
A ^C RESTART WILL REQUEST PARAMETERS.

NOTE: AFTER RESTARTING THE SWITCH REGISTER SHOULD
BE SET TO PROGRAM SWITCH SETTINGS. IF 210
IS LEFT AS THE SWITCH SETTING THE PROGRAM
WILL SELECT & RUN TEST 10 ONLY. SEE SWITCH
SETTINGS FOR EXPLANATION.

2.2 AUTOMATIC MODE OPERATION

IF THE PROGRAM IS LOADED AND RUN IN AUTOMATIC (CHAIN) MODE
DEFAULT RESPONSES TO OPERATOR REQUESTS ARE USED, AND ALL AVAIL-
ABLE TMO3-TE16/TU77 COMBINATIONS ARE TESTED. ADDITIONALLY THE SOFTWARE
SWR IS INVOKED WITH A SWITCH SETTING OF 000000 IF LOADED VIA ACT11.
NO OPERATOR INTERVENTION IS REQUIRED

** EXCEPTION: IF LOADED VIA TMDP TMO3 DRIVE 0 TE16/TU77 SLAVE 0 IS
NOT TESTED.

**NOTE: IN ORDER TO CHANGE THE DEFAULT SETTING OF THE SOFTWARE
SWR, SET LOC: 176(SWREG:) TO THE DESIRED SETTING.

TMO3 DRIVE FUNCTION TIMER
SWITCH SETTINGS

PAGE 6

CHAPTER 3
SWITCH SETTINGS

CONTROL :

- 1) CONTROL G <^G>:
SELECTS THE SOFTWARE SWR AND ALLOWS NEW SWITCH SETTINGS.
THE MACHINE WILL THEN TYPE: SWR=XXXXXXNEW-
WHERE: XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWR.
AFTER THE ''NEW='' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE
OF THE FOLLOWING AT THE TTY:
A) TYPE A NUMBER TO BE LOADED INTO THE SOFTWARE SWR.
B) IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH
REGISTER CONTENTS WILL NOT BE CHANGED.
- 2) CONTROL A <^A>:
ALTERNATES USAGE OF THE SWR BETWEEN HARDWARE & SOFTWARE.
- 3) CONTROL C <^C>:
RESTARTS PROGRAM AT 200
- 4) CONTROL U <^U>:
DELETES ALL CHARACTERS TYPED IN RESPONSE TO A REQUEST.

SW15 (100000)		HALT ON ERROR THIS SWITCH WHEN SET WILL HALT THE PROCESSOR WHEN AN ERROR IS DETECTED. THE PC+2 AND PSW AT THE TIME OF THE ERROR IS STORED ON THE STACK. PRESSING CONTINUE WILL CAUSE THE ERROR TO BE TYPED (IF SELECTED) AND FURTHER TESTING RESUMED.
SW14 (040000)	LOOP SUBTEST	THIS SWITCH WHEN SET LOOPS THE CURRENT SUBTEST REGARDLESS OF ERROR CONDITION.
SW13 (020000)	INHIBIT ERROR TYPEOUT	THIS SWITCH WHEN SET INHIBITS ERROR TYPEOUT.
SW11 (004000)	INHIBIT SUB- TEST ITERATION	THIS SWITCH WHEN SET CAUSES EACH SUBTEST TO BE EXECUTED ONLY ONCE.
SW10 (002000)	INHIBIT FUNCTION TIME PUBLICATION	THIS SWITCH WHEN SET WILL INHIBIT THE PRINTING OF THE FUNCTION TIMES. (SEE CHAPTER 8.)
SW09 (001000)	RING BELL ON ERROR	THIS SWITCH WHEN SET WILL RING THE BELL ON THE TTY WHEN AN ERROR IS DETECTED.
SW06 (000100)	CONTINUOUS CYCLE	THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO RUN CONTINUOUSLY UNTIL STOPPED BY THE OPERATOR.
SW5-0	TEST SELECT	RUN SUBTEST SELECTED

NOTE: A TEST CAN ONLY BE SELECTED DURING STARTUP (OR RESTART).
DO NOT INHIBIT SUBTEST ITERATIONS WHEN PROGRAM IS RUNNING.

TMO3 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 ERI
AAAAAA BBBB BB CCCCC DDDDD EEEEE FFFFF GGGGG

WHERE:

XXXXXX = TEST NUMBER
AAAAAA-III IIII CONTENTS OF TAPE REGISTER 172440-172454

4.2 ERROR TYPEOUT FORMAT (FUNCTION TIME OUT OF RANGE)

TEST # XXXXXX OUT OF RANGE ERROR

RANGE <AAAAAA-BBBBB> ACTUAL = CCCCC

TMO3 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. PROVIDES CONTINUOUS LOOP <SW14>
3. MOVES FUNCTION TIME INTO TABLE
4. OUTPUTS LINE ITEM <SW10> 1
5. DELAYS 350MS BEFORE STARTING TEST
6. INIT'S DRIVE/SLAVE
7. CLEARS THE ERROR FLAG (ERFLG)
8. CHECK FOR CONTROL G (^G)

THE ROUTINE MONITORS SW14, SW11, SW10, 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 THE TIME RECORDED BY THE SUBTEST.

TMO3 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.

TMO3 DRIVE FUNCTION TIMER
MISCELLANEOUS

PAGE 10

CHAPTER 6
MISCELLANEOUS

6.1 STACK POINTER

THE STACK POINTER IS INITAILLY SET TO 500.

6.2 EXECUTION TIME

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

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

TMO3 DRIVE FUNCTION TIMER
PROGRAM DESCRIPTION

CHAPTER 7
PROGRAM DESCRIPTION

7.1 SAMPLE TIME DOCUMENT

TYPE FIRST ADDRESS OF CONTROLLER:172440
TYPE TMO3 DRIVE #'S TO BE TESTED:ALL 0
FOR TMO3 DRIVE J- TYPE SLAVE #'S TO BE TESTED:ALL 0
TAPE SPEED TESTS ONLY? (YES/NO):NO

* FUNCTION	TIME (SPECIFICATION)	TIME (ACTUAL)
* TMO3 DRIVE FUNCTION TIMES- DRIVE # 0 SLAVE # 7 TE16 SER. # 5009		
* FUNCTION	TIME (SPECIFICATION)	TIME (ACTUAL)
* WRITE FROM BOT	RANGE=<176000-172000>	ACTUAL=174740
* WRITE START	RANGE=<009500-008700>	ACTUAL=009120
* WRITE SHUTDOWN	RANGE=<008500-007500>	ACTUAL=008840
* WRITE SETTLEDOWN	RANGE=<013500-007300>	ACTUAL=010970
* READ FROM BOT	RANGE=<045000-041000>	ACTUAL=043580
* READ START	RANGE=<003200-002400>	ACTUAL=002740
* READ SHUTDOWN	RANGE=<004100-003050>	ACTUAL=004360
* READ SETTLEDOWN	RANGE=<013500-007300>	ACTUAL=010970
* READ REV START	RANGE=<003200-002400>	ACTUAL=002740
* READ REV SHUTDOWN	RANGE=<003700-003300>	ACTUAL=003520
* READ REV SETTLEDOWN	RANGE=<013500-007300>	ACTUAL=010970
* TURN AROUND DELAY F-R	RANGE=<016700-010700>	ACTUAL=013600
* TURN AROUND DELAY R-F	RANGE=<016700-010700>	ACTUAL=013660
* GAP SIZE-STOP HALF	RANGE=<012900-009500>	ACTUAL=012200
* GAP SIZE-START HALF	RANGE=<011800-008500>	ACTUAL=010520
* GAP SIZE-INTERRECORD	RANGE=<014300-012600>	ACTUAL=014500
* GAP CONSISANCY	RANGE=<014000-011800>	ACTUAL=013040-
* DATA TIME-800BPI	RANGE=<024000-022000>	ACTUAL=023400
* DATA TIME-1600BPI	RANGE=<025100-024100>	ACTUAL=024470
* ERASE GAP TIME	RANGE=<101000-098000>	ACTUAL=099510
* WRITE FILE MARK	RANGE=<104000-102000>	ACTUAL=103990

TMO3 DRIVE FUNCTION TIMER

7.1.1 SAMPLE TIME DOCUMENT FOR TAPE SPEED TESTS

TYPE FIRST ADDRESS OF CONTROLLER 172440:
TYPE TMO3 DRIVE #'S TO BE TESTED:ALL 0
FOR TMO3 DRIVE 0- TYPE SLAVE #'S TO BE TESTED:ALL 7
SPEED TESTS ONLY? (YES/NO):NO Y

*TMO3 DRIVE FUNCTION TIMES- DRIVE # 0 SLAVE # 7 TE16 SERIAL # 5009

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

TMO3 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	*M8931/M8940 ROM*M8933 ACCL CNTR
2. WRITE START	* ''	* '' * ''
3. WRITE SHUTDOWN	* ''	* '' * ''
4. WRITE SETTLEDOWN	* ''	*M8916/M8940 (TU77) SETTLEDOWN O
5. READ FROM BOT	* ''	*M8931/M8940 ROM*M8933 ACCL CNTR
6. READ START	* ''	* '' * ''
7. READ SHUTDOWN	* ''	* '' * ''
10. READ SETTLEDOWN	* ''	*M8916/M8940 (TU77) SETTLEDOWN O
11. READ REVERSE START	* ''	*M8931/M8940 ROM*M8933 ACCL CNTR
12. READ REVERSE SHUTDOWN	* ''	* '' * ''
13. READ REVERSE SETTLEDOWN	* ''	*M8916/M8940 (TU77) SETTLEDOWN O
14. TURN AROUND F-R	* ''	*M8931/M8940 ROM*M8933 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	* '' '' ''

TMO3 DRIVE FUNCTION TIMER

PAGE 14

- | | | |
|------------------------|---------------------|----------------------------------|
| 21. GAP CONSISTENCY | *SAME AS IN TEST 16 | *WRITE CLOCK |
| 22. DATA TIME 800 BPI | *NONE | * '' '' |
| 23. DATA TIME 1600 BPI | * '' | * '' '' |
| 24. ERASE GAP TIME | * '' | *M8931/M8940 ROM*M8933 ACCL CNTR |
| 25. WRITE FILE MARK | * '' | * '' '' * '' '' '' |
| 26. TAPE SPEED-FORWARD | *FWD SPEED | *CAPSTAN SERVO LOOP |
| 27. TAPE SPEED-REVERSE | *REVERSE SPEED | *CAPSTAN SERVO LOOP |

**** NOTE: IF TIME PROBLEMS APPEAR IN T1 THRU T25, RUN TAPE SPEED TESTS FIRST*****
TEST 26 & 27 REQUIRE AN 800 BPI SKEW TAPE

TM03 DRIVE FUNCTION TIMER

PAGE 15

7.3 SUBTEST DESCRIPTIONS:

THE FIRST THIRTEEN (13) TESTS (T1 - T15) ARE CHECKS OF THE ROM CIRCUITS IN THE TE16 (M8931/M8940), THE ACCL COUNTER IN THE TM03 (M8933), AND THE SETTLEDOWN ONE SHOT (M8916/M8940 (TU77)).

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

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

TMO3 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

TM03 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.

TMO3 DRIVE FUNCTION TIMER

PAGE 19

GAP MEASUREMENTS:

THE PREVIOUS THIRTEEN (13) TESTS WERE MEASUREMENTS OF LOGIC DELAYS PERFORMED BY THE TMO3 OR TE16/TU77 IN ORDER TO ALLOW FOR PROPER ACCELERATION AND DECELERATION OF TAPE ACCORDING TO THE DESIRED INTERCORD GAP (.6 INCHES). THIS TEST, HOWEVER, WILL MEASURE THE PHYSICAL SIZE OF THE INTERCORD GAP THAT EXISTS ON TAPE AS A RESULT OF THE START/STOP TIMES OF THE CAPSTAN ITSELF. BECAUSE THE INTERCORD GAP IS CREATED BY TWO ACTIONS, THE START OF MOTION AND THE STOP OF MOTION IT IS NECESSARY TO MAKE TWO SEPERATE MEASUREMENTS. A THIRD MEASUREMENT, MADE ON THE FLY, OF THE ENTIRE LENGTH OF THE GAP WILL ALSO BE MADE.

T16. GAP SIZE (STOP HALF)

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

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

T17. GAP SIZE (START HALF)

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

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

T20. GAP SIZE (INTERRECORD)

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

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

TMO3 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 INTERCORD 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 LISTING FOR GAP TIMES) **

T22. DATA TIME AT 800 BPI:

THIS TEST WILL MEASURE THE TIME REQUIRED TO WRITE ONE (1) INCH OF TAPE AT 800 BPI. BY WRITING A RECORD OF ENOUGH FRAMES TO MOVE THE TAPE 1 INCH (800 FRAMES), DATA RATE CAN BE VARIFIED.

1. REWIND TO BOT AND ALLOW TAPE TO STOP
2. WRITE A RECORD AT 800 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

T23. DATA TIME AT 1600 BPI (PE):
REPEAT STEPS 1 THRU 5 AT 1600 BPI.

TM03 DRIVE FUNCTION TIMER

PAGE 21

T24. 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

T25. 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

TMO3 DRIVE FUNCTION TIMER

PAGE 22

T26. 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 = 26400(10)
5. TIME FROM FC = 800 TO FC = 26400 IS THE TIME REQUIRED FOR TAPE TO TRAVEL 32 INCHES
6. DIVIDE THE TIME FOR 32 INCHES BY 32.
7. THE RESULT IS AN AVERAGE SPEED FOR 1 INCH.
8. STOP.

T27. 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.

%

```

982          .LIST  BIN,LOC,SEQ
983          .NLIST MC
984          .NLIST TOC
985          .LIST  ME
986          .ENABLE ABS,AMA
987          .MCALL $CPVEC,$CPREG,$CATCH,$TYPE,..$ACT11,..$EOP,$CHAIN
988          .TITLE  CZTEECO TM03-TE16/TU77 DFT
989          ;DRIVE FUNCTION TIMER
990          .SBTTL  STARTING INSTRUCTIONS
991          ;LOADING AND STARTING PROCEDURE
992          LOAD PROGRAM USING ABS LOADER
993          LOAD ADDRESS 200
994          SET SWITCH OPTIONS
995          PRESS START
996
997          ;RESTART PROCEDURE
998          LOAD ADDRESS 210
999          SET SWITCH OPTIONS
1000         PRESS START
1001
1002         ;SWITCH REGISTER SWITCH ASSIGNMENTS
1003         100000 SW15= 100000 ;HALT ON ERROR
1004         040000 SW14= 040000 ;LOOP SUBTEST
1005         020000 SW13= 020000 ;INHIBIT ERROR TYPE OUT
1006         004000 SW11= 004000 ;INHIBIT SUBTEST ITERATION
1007         002000 SW10= 002000 ;INHIBIT PUBLISHING TIME SPECIFICATION
1008         001000 SW09= 001000 ;RING BELL ON ERROR
1009         000400 SW08= 000400
1010         000200 SW07= 00200 ;NOT USED
1011         000100 SW06= 000100 ;CONTINUOUS CYCLE
1012         SW05-SW00 ;RUN TEST SELECTED
1013         ;**NOTE: IF <SW15-SW00> = 177777 AT STARTUP USE SOFTWARE
1014         ;SWITCH REGISTER.
1015
1016         ;CONSOLE COMMANDS
1017         CONTROL C ;RESTART PROGRAM (SAME AS START @ 200)
1018         CONTROL G ;SET NEW SOFTWARE SWITCH REGISTER
1019         CONTROL U ;DELETE LINE TYPED
1020         RUBOUT (DELETE) ;DELETE LAST CHAR TYPED
1021
1022         ;GENERAL REGISTER USAGE:
1023         R0=ADDRESS OF 'FC' REGISTER (SET BY SCOPE)
1024         R1=ADDRESS OF 'DS' REGISTER (SET BY SCOPE)
1025         R2=RETURN PC FROM TIMER (SET BY EACH TEST)
1026         R3=INDEX INDICATING PREVIOUS OSCILLATOR POLARITY (SET BY TIMER)
1027         R4=CONTAINS 'TICK' COUNT WHEN TIMER IS RUNNING (SET BY TIMER)
1028         R5=ADDRESS OF CS1 (SET BY SCOPE)
1029
1030         .SBTTL  MACRO DEFINITIONS
1031         .MACRO  SAVE
1032         JSR    PC,..SAVE ;SAVE REGISTERS ON THE STACK
1033         .ENDM  SAVE
1034         .MACRO  RESTORE
1035         JSR    PC,..RESTORE ;RESTORE REGISTERS FROM THE STACK
1036         .ENDM  RESTORE
1037         .MACRO  INPUT
  
```

```
1038 JSR PC, INPUT ;GET USER INPUT
1039 .ENDM INPUT
1040 .MACRO REWIND
1041 JSR PC, REWIND ;REWIND SLAVE
1042 BVS 99$ ;BRANCH IF ERROR ON REWIND
1043 .ENDM REWIND
1044 .MACRO TIMEON
1045 JSR PC, TIMON ;TURN TIMER ON
1046 .ENDM
1047 .MACRO TIMCHK
1048 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
1049 .ENDM
1050 .MACRO SETGO
1051 INC (R5) ;SET 'GO' BIT
1052 .ENDM
```

```
1054 .SBTTL REGISTER ASSIGNMENTS
1055 ;;DEFINITIONS AND REGISTER ASSIGNMENTS
1056 ;;GENERAL REGISTER ASSIGNMENTS
```

```
1057 000000 R0=%0
1058 000001 R1=%1
1059 000002 R2=%2
1060 000003 R3=%3
1061 000004 R4=%4
1062 000005 R5=%5
1063 000006 SP=%6
1064 000007 PC=%7
1065 000000 R10=%0
1066 000001 R11=%1
1067 000002 R12=%2
1068 000003 R13=%3
1069 000004 R14=%4
1070 000005 R15=%5
```

```
1072 ;;REGISTER ADDRESSES
```

```
1073 177776 PSW= 177776 ;;PROCESSER STATUS WORD
1074 177774 SLR= 177774 ;;STACK LIMIT REGISTER (11/40,11/45)
1075 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQ. (11/45)
1076 177770 UBREAK= 177770 ;;MICRO-BREAK REGISTER (11/45)
1077 177560 TKS= 177560 ;;KEYBOARD CSR
1078 177562 TKB= 177562 ;;KEYBOARD DATA BUFFER REGISTER
1079 177564 TPS= 177564 ;;TELEPRINTER CSR
1080 177566 TPB= 177566 ;;TELEPRINTER DATA BUFFER REGISTER
```

```
1082 ;;VECTOR ADDRESSES
```

```
1083 000004 ERRVEC=4 ;;ADDRESS OF ERROR VECTOR
1084 000010 RESVEC=10 ;;ADDRESS OF RESERVED INST. TRAP VECTOR
1085 000014 TBITVEC=14 ;;ADDRESS OF 'T' BIT TRAP VECTOR
1086 000014 TRTVEC=14 ;;ADDRESS OF 'TRACE' TRAP VECTOR
1087 000014 BPTVEC=14 ;;ADDRESS OF 'BREAKPOINT' TRAP VECTOR
1088 000020 IOTVEC=20 ;;ADDRESS OF IOT TRAP VECTOR
1089 000024 PFVEC=24 ;;ADDRESS OF POWER FAIL TRAP VECTOR
1090 000030 EMTVEC=30 ;;ADDRESS OF EMT VECTOR
1091 000034 TRAPVEC=34 ;;ADDRESS OF TRAP VECTOR
1092 000060 TKVEC= 60 ;;ADDRESS OF TTY KEYBOARD IN*. VECTOR
1093 000064 TPVEC=64 ;;ADDRESS OF TTY PRINTER INTERRUPT VECTOR
```

1094	000114	PARVEC= 114	::ADDRESS OF MA/MF PARITY ERROR VECTOR
1095	000240	PIRVEC=240	::ADDRESS OF PIRQ VECTOR
1096	000244	FPEVEC=244	::ADDRESS OF FLOATING POINT INT. VECTOR
1097	000250	MMVEC=250	::ADDRESS OF MEM MGMT ERROR TRAP VECTOR
1098			

1099
1100 172440
1101
1102
1103 000000
1104 000002
1105 000004
1106 000006
1107 000010
1108 000012
1109 000014
1110 000016
1111 000022
1112 000024
1113 000026
1114 000030
1115 000032
1116
1117
1118
1119 000001
1120 000000
1121 000002
1122 000006
1123 000010
1124 000026
1125 000024
1126 000030
1127 000032
1128 000050
1129 000056
1130 000060
1131 000070
1132 000076
1133 000100
1134 000200
1135 000400
1136 001000
1137 002000
1138 004000
1139 020000
1140 040000
1141 100000
1142
1143 000000
1144 000001
1145 000002
1146 000003
1147 000004
1148 000005
1149 000006
1150 000007
1151 000010
1152 000020
1153 000040
1154 000100

:RH, TM03-TE16/TU77 REGISTERS
TMCS1= 172440

:TM03-TE16/TU77 INDEX VALUES

CS1=	00	:CONTROL STATUS #1
WC=	02	
BA=	04	:BUS ADDRESS REGISTER
FC=	06	:FRAME COUNT
CS2=	10	:CONTROL STATUS #2
DS=	12	:DRIVE STATUS
ER=	14	:ERROR REG #1
AS=	16	:ATTENTION SUMMARY
DB=	22	:DATA BUFFER REG
MR=	24	:MAINTENANCE REG
DT=	26	:DRIVE TYPE REG
SN=	30	:SERIAL NUMBER REGISTER
TC=	32	:TAPE CONTROL REG

.SBTTL TM03-TE16/TU77 REGISTER BITS
:RHCS1-CS1(R5)

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

:RHCS2-CS2(R5)

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

1155	000200	OR=	200	
1156	000400	MDPE=	400	
1157	001000	MXF=	1000	
1158	002000	PGE=	2000	
1159	004000	NEM=	4000	
1160	010000	NED=	10000	
1161	020000	UPE=	20000	
1162	040000	WCE=	40000	
1163	100000	DLT=	100000	
1164		;RHDS-DS(R5)		
1165	000001	SLA=	1	
1166	000002	BOT=	2	
1167	000004	TMK=	4	
1168	000010	IDB=	10	
1169	000020	SDWN=	20	
1170	000040	PES=	40	
1171	000100	SSC=	100	
1172	000200	DRY=	200	
1173	000400	DPR=	400	
1174	002000	EOT=	2000	
1175	004000	WRL=	4000	
1176	010000	MOL=	10000	
1177	020000	PIP=	20000	
1178	040000	ERR=	40000	
1179	100000	ATA=	100000	
1180		;RHER-ER(R5)		
1181	000001	ILF=	1	
1182	000002	ILR=	2	
1183	000004	RMR=	4	
1184				
1185	000020	FMT=	20	
1186	000100	INCVAE=	100	
1187	000200	PEFLRC=	200	
1188	000400	NSG=	400	
1189	001000	FCE=	1000	
1190	002000	CSITM=	2000	
1191	004000	NEF=	4000	
1192	010000	DTE=	10000	
1193	020000	OPI=	20000	
1194	040000	UNS=	40000	
1195	075027	HRDERR=	UNS.OPI.DTE.NEF.FCE.FMT.RMR.ILR!ILF	;HARDERROR BITS
1196				
1197		;RHMR-MR(R5)		
1198	000100	OSC=	100	
1199				
1200		;RHDT-DT(R5)		
1201	002000	SPR=	2000	
1202	010000	CH7=	10000	
1203	040000	TAP=	40000	
1204				
1205		;RHTC-TC(R5)		
1206	001700	NORM11=	1700	
1207	000320	CDM11=	320	
1208	000000	BPI200=	0	
1209	000400	BPI556=	000400	
1210	001000	BPI800=	001000	

1211 002300
 1212 100000
 1213
 1214
 1215
 1216
 1217 104400
 1218 104000
 1219 000004
 1220
 1221
 1222 006274
 1223 177400
 1224 177600
 1225
 1226 000001
 1227 000003
 1228 000007
 1229 000011
 1230 000012
 1231 000015
 1232 000017
 1233 000025

PE1600= 002300
 ACCL= 100000

; INSTRUCTION EQUATES

HLT= TRAP
 SCOPE= EMT
 TYPE= IOT

; MISCELLANEOUS EQUATES

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

; ASCII EQUATES

CNTRLA= 1
 CNTRLC= 3
 CNTRLG= 7
 HT= 11
 LF= 12
 CR= 15
 CNTRLO= 17
 CNTRLU= 25

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

; ASCII CODE FOR CONTROL A (^A)
 ; ASCII CODE FOR CONTROL C (^C)
 ; ASCII CODE FOR CONTROL G (^G)
 ; 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)

```
1234 ;SETUP TRAP VECTORS
1235      . =TBITVEC
1236 000014 000016      .WORD      .+2      ;SET 'T' TRAP TO TIMER ROUTINE
1237 000016 000000      .WORD      HALT      ;PRIORITY LEVEL 7
1238 000020 002636      .WORD      .TYPE     ;SET IOT TRAP TO .TYPE ROUTINE
1239 000022 000000      .WORD      0         ;PRIORITY LEVEL 0
1240 000024 000026      .WORD      PFVEC+2   ;POWER FAIL TRAP TO HALT
1241 000026 000000      .WORD      HALT      ;AT PFVEC+2
1242 000030 004516      .WORD      .SCOPE    ;SET EMT TRAP TO .SCOPE ROUTINE
1243 000032 000340      .WORD      340       ;PRIORITY LEVEL 7
1244 000034 004252      .WORD      .HLT      ;SET TRAP TRAP TO .HLT ROUTINE
1245 000036 000340      .WORD      340       ;PRIORITY LEVEL 7
1246
1247 ;ACT11 HOOK *****
1248      $SVPC=.      ;SAVE CURRENT LOCATION CTR
1249      . =42
1250 000042 000000      .WORD      0
1251      . =46
1252 000046 013504      .WORD      $ENDAD    ;SET LOCATION 46
1253      . =52
1254 000052 000000      .WORD      0         ;SET LOCATION 52 = 0
1255      . = $SVPC    ;RESTORE LOCATION CTR
1256
1257      . =TKVEC
1258 000060 004126      .WORD      TKISR
1259 000062 000200      .WORD      200
1260
1261 ;SOFTWARE SWITCH REGISTER LOC. 176
1262      . =176
1263 000176 000000      SWREG: .WORD      0      ;SOFTWARE SWITCH REGISTER
1264
1265      . =200
1266 000200 000137 006274      JMP      @#INIT      ;GO TO START OF PROGRAM
1267      . =210
1268 000210 000137 007414      JMP      @#RSTRT    ;RESTART ADDRESS
1269
1270      . =500
1271      STKPTR= 600      ;STACK
1272
1273      . =-1000
1274 ;PROGRAM TAGS
1275 001000 177570      SWR:      177570      ;SWITCH REGISTER
1276 001002 000000      SCPADR: .WORD      0
1277 001004 000      DRVNUM: .BYTE      0      ;TM03 DRIVE UNDER TEST
1278 001005 000      SLVNUM: .BYTE      0      ;TE16/TU77 SLAVE UNDER TEST
1279 001006 000000      SLVPTR: .WORD      0      ;POINTER TO SLAVE TABLE (SLVTBL) BELOW
1280 001010 172440      TMBASE: .WORD      TMCS1 ;BASE ADDRESS OF TM03-TE16/TU77 REGISTERS
1281 001012 000000      OSCTIM: .WORD      0      ;US/TICK (56/80 FOR TE16/TU77)
1282 001014 000000      GAPDEL: .WORD      0      ;TICKS/MS (18/6 FOR TE16/TU77)
1283 001016 000000      ATIME:  .WORD      0      ;CONTAINS 'TICK' COUNT
1284 001020 000020      ATIMTBL: .BLKW     16.    ;EACH ENTRY CONTAINS TIME FOR FUNCTION
1285      ;ENTRIES ARE MADE BY 'SCOPE' ROUTINE
1286 001060 000020      GAPTBL: .BLKW     16.    ;TIMES RECORDED BY 'GAP CONSISTANCY' TEST
1287 001120 000000      DELTIM: .WORD      0      ;VARIABLE DELAY
1288 001122 000000      OCTALO: .WORD      0
1289 001124 000      GAP:      .BYTE      0      ;CONTAINS GAP # (USED FOR TST 021)
```

1290	001125	000	ITCNT:	.BYTE	0	; ITERATION COUNT
1291	001126	000	TSTNUM:	.BYTE	0	; TEST #
1292	001127	000	ERFLG:	.BYTE	0	; ERROR FLAG
1293	001130	000	SKEWFLG:	.BYTE	0	; 0/1 = DO NOT/DO SKEW (SPEED) TESTS
1294	001131	000	PRGFLG:	.BYTE	0	; PROGRAM FLAG
1295	001132	000	UNTFND:	.BYTE	0	; UNIT FOUND INDICATOR
1296	001133	000	TYPFLG:	.BYTE	0	
1297	001134	000	PSCNT:	.BYTE	0	; CONTAINS PASS COUNT
1298	001135	000	ASFLG:	.BYTE	0	; 1/0 = YES/NO.
1299	001136	000	TE16:	.BYTE	0	; 0/1 = TE16/TU77
1300		001140		.EVEN		
1301	001140	030460	DIGTAB:	'01		
1302	001142	031462		'23		
1303	001144	032464		'45		
1304	001146	033466		'67		
1305	001150	034470		'89		
1306	001152	000006	ODIGITS:	.BLKB	6	; RESERVE SPACE FOR CONVERTED DIGITS
1307	001160	000		.BYTE	0	; TERMINATOR
1308		001162		.EVEN		
1309	001162	000010	DRVTBL:	.BLKB	8	; A 0/-1 = DRIVE NOT TO BE/TO BE TESTED
1310	001172	000100	SLVTBL:	.BLKB	64	; A 0/-1 = SLAVE NOT TO BE/TO BE TESTED
1311	001272	000110	INBUF:	.BLKB	72	; TELETYPE INPUT BUFFER
1312	001402	005015	CRLF:	.ASCIZ	<CR><LF>	; MISCELLANEOUS ASCII CHARACTERS
1313	001405	134	BACKSLSH:	.ASCIZ	'\'	
1314	001407	060	FCHO:	.ASCIZ	'0'	
1315	001411	007	BELL:	.ASCIZ	<7>	
1316	001413	055	DASH:	.ASCIZ	'-'	
1317	001415	040	SPACE2:	.ASCII	' '	
1318	001416	000040	SPACE:	.ASCIZ	' '	
1319	001420	004476	ANGTAB:	.ASCIZ	'>'<HT>	
1320		001424		.EVEN		

1321
 1322
 1323
 1324
 1325
 1326
 1327
 1328 001424 000000 000000
 1329 001430 036050 035230
 1330 001434 001666 001546
 1331 001440 001522 001356
 1332 001444 002506 001332
 1333 001450 007164 006344
 1334 001454 000500 000360
 1335 001460 000632 000454
 1336 001464 002506 001332
 1337 001470 000500 000360
 1338 001474 000562 000512
 1339 001500 002506 001332
 1340 001504 003206 002056
 1341 001510 003206 002056
 1342 001514 002412 001666
 1343 001520 002234 001522
 1344 001524 002626 002354
 1345 001530 002570 002234
 1346 001534 004540 004230
 1347 001540 004716 004552
 1348 001544 023564 023110
 1349 001550 024240 023730
 1350 001554 004336 004172
 1351 001560 004336 004172

.SBTTL TE16 TIME SPECIFICATION TABLE
 :THE BELOW TABLE CONTAINS THE TE16 SPECIFIED FUNCTION TIMES IN TENS OF
 :MICROSECONDS. NOTE THAT WHEN TIMES ARE TYPED THAT THEY ARE TYPED IN
 :MICROSECONDS (BY APPENDING A 0).
 :FORMAT IS

: .WORD	MAX,MIN	: TIME IN MS	FUNCTION	TEST #
TE16TTBL: .WORD	0,0	: SFARE		
.WORD	15400.,15000.	:154.0-150.0	WRITE FROM BOT	TST001
.WORD	00950.,00870.	:9.5-8.7	WRITE START	TST002
.WORD	00850.,00750.	:8.9-8.5	WRITE SHUTDOWN	TST003
.WORD	01350.,00730.	:13.5-7.3	WRITE STLDOWN	TST004
.WORD	03700.,03300.	:37.0-33.0	READ FROM BOT	TST005
.WORD	00320.,00240.	:3.2-2.4	READ START	TST006
.WORD	00410.,00300.	:4.1-3.00	READ SHUTDOWN	TST007
.WORD	01350.,00730.	:13.5-7.3	READ SETTLEDOWN	TST010
.WORD	00320.,00240.	:3.2-2.4	RD REV START	TST011
.WORD	00370.,00330.	:3.7-3.3	RD REV SHTDWN	TST012
.WORD	01350.,00730.	:13.5-7.3	RD REV STLDWN	TST013
.WORD	01670.,01070.	:16.7-10.7	TRN RND DLY F-R	TST014
.WORD	01670.,01070.	:16.7-10.7	TRN RND DLY R-F	TST015
.WORD	01290.,00950.	:12.9-9.5	GAP SIZE STOP	TST016
.WORD	01180.,00850.	:11.8-8.5	GAP SIZE STRT	TST017
.WORD	01430.,01260.	:14.3-12.6	GAP SIZE INTER	TST020
.WORD	01400.,01180.	:14.0-11.8	GAP CONSISANCY	TST021
.WORD	02400.,02200.	:24.0-22.0	DAT TIME 800BPI	TST022
.WORD	02510.,02410.	:25.1-24.1	DAT TIME 1600PE	TST023
.WORD	10100.,09800.	:101.0-98.0	ERASE	TST024
.WORD	10400.,10200.	:104.0-102.0	WRT FILE MARK	TST025
.WORD	02270.,02170.	:22.7-21.7	TAPE SPEED FWD	TST026
.WORD	02270.,02170.	:22.7-21.7	TAPE SPEED REV	TST027

:NOTE: TEST 26 AND 27 REQUIRE PRERECORDED 800BPI SKEW TAPE.

.SBTTL TE16 GAP TIME SPECIFICATION TABLE
 :THIS TABLE CONTAINS THE TE16 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.

1352
 1353
 1354
 1355
 1356
 1357
 1358
 1359
 1360
 1361 001564 002602 002412
 1362 001570 002652 002506
 1363 001574 002734 002532
 1364 001600 003016 002424
 1365 001604 003016 002304
 1366 001610 002734 002176
 1367 001614 002652 002176
 1368 001620 002652 002176
 1369 001624 002570 002176
 1370 001630 002570 002176
 1371 001634 002570 002176
 1372 001640 002570 002176
 1373 001644 002570 002176
 1374 001650 002570 002176
 1375 001654 002570 002176
 1376 001660 002570 002176

: .WORD	MAX,MIN(10)	: TIME IN MS(10)	GAP #	DELAY IN MS(10)
TE16GTBL: .WORD	01410.,01290.	:14.10-12.9	GAP-0	0 MS
.WORD	01450.,01350.	:14.5-13.5	GAP-1	1.0 MS
.WORD	01500.,01370.	:15.0-13.7	GAP-2	2.0 MS
.WORD	01550.,01300.	:15.5-13.0	GAP-3	3.0 MS
.WORD	01550.,01220.	:15.5-12.2	GAP-4	4.0 MS
.WORD	01500.,01150.	:15.0-11.5	GAP-5	5.0 MS
.WORD	01450.,01150.	:14.5-11.5	GAP-6	6.0 MS
.WORD	01450.,01150.	:14.5-11.5	GAP-7	7.0 MS
.WORD	01400.,01150.	:14.0-11.5	GAP-10	8.0 MS
.WORD	01400.,01150.	:14.0-11.5	GAP-11	9.0 MS
.WORD	01400.,01150.	:14.0-11.5	GAP-12	10.0 MS
.WORD	01400.,01150.	:14.0-11.5	GAP-13	11.1 MS
.WORD	01400.,01150.	:14.0-11.5	GAP-14	12.1 MS
.WORD	01400.,01150.	:14.0-11.5	GAP-15	13.1 MS
.WORD	01400.,01150.	:14.0-11.5	GAP-16	14.1 MS
.WORD	01400.,01150.	:14.0-11.5	GAP-17	15.1 MS

1377
 1378
 1379
 1380
 1381
 1382
 1383
 1384
 1385
 1386
 1387
 1388
 1389
 1390
 1391
 1392
 1393
 1394
 1395
 1396
 1397
 1398
 1399
 1400
 1401
 1402
 1403
 1404
 1405
 1406
 1407
 1408
 1409
 1410
 1411
 1412
 1413
 1414
 1415
 1416
 1417
 1418
 1419
 1420
 1421
 1422
 1423
 1424
 1425
 1426
 1427
 1428
 1429
 1430
 1431
 1432

001664 000000 000000
 001670 012606 011571
 001674 000520 000460
 001700 000346 000313
 001704 003410 001717
 001710 001315 001112
 001714 000111 000067
 001720 000111 000067
 001724 003410 001717
 001730 000111 000067
 001734 000101 000057
 001740 003410 001717
 001744 003500 002006
 001750 003510 002017
 001754 000702 000545
 001760 000642 000505
 001764 001012 000646
 001770 001034 000641
 001774 001515 001434
 002000 001536 001434
 002004 007020 006476
 002010 007176 006642
 002014 001560 001320
 002020 001560 001320

.SBTTL TU77 TIME SPECIFICATION TABLE
 :THE BELOW TABLE CONTAINS THE TU77 SPECIFIED FUNCTION TIMES IN TENS OF
 :MICROSECONDS. NOTE THAT WHEN TIMES ARE TYPED THAT THEY ARE TYPED IN
 :MICROSECONDS (BY APPENDING A 0).
 :FORMAT IS
 :
 : .WORD MAX,MIN ;TIME IN MS FUNCTION TEST #

.WORD	MAX,MIN	TIME IN MS	FUNCTION	TEST #
TU77TTBL: .WORD	0,0	:SPARE		
.WORD	5510.,4985.	:55.1-49.85	WRITE FROM BOT	TST001
.WORD	00336.,00304.	:3.36-3.04	WRITE START	TST002
.WORD	00230.,00203.	:2.3-2.03	WRITE SHUTDOWN	TST003
.WORD	01800.,00975.	:18.0-9.75	WRITE STLDOWN	TST004
.WORD	00717.,00586.	:7.17-5.86	READ FROM BOT	TST005
.WORD	00073.,00055.	:.73-.55	READ START	TST006
.WORD	00073.,00055.	:.73-.55	READ SHUTDOWN	TST007
.WORD	01800.,00975.	:18.0-9.75	READ SETTLEDOWN	TST010
.WORD	00073.,00055.	:0.73-0.55	RD REV START	TST011
.WORD	00065.,00047.	:0.65-0.47	RD REV SHTDWN	TST012
.WORD	01800.,00975.	:18.0-9.75	RD REV STLDWN	TST013
.WORD	01856.,01030.	:18.56-10.3	TRN RND DLY F-R	TST014
.WORD	01864.,01039.	:18.64-10.39	TRN RND DLY R-F	TST015
.WORD	0450.,00357.	:4.50-3.57	GAP SIZE STOP	TST016
.WORD	0418.,00325.	:4.18-3.25	GAP SIZE STRT	TST017
.WORD	0522.,0422.	:5.22-4.22	GAP SIZE INTER	TST020
.WORD	0540.,0417.	:5.40-4.17	GAP CONSISANCY	TST021
.WORD	0845.,0796.	:8.45-7.96	DAT TIME 800BPI	TST022
.WORD	0862.,0796.	:8.62-7.96	DAT TIME 1600PE	TST023
.WORD	3600.,03390.	:36.00-33.90	ERASE	TST024
.WORD	3710.,3490.	:37.10-34.90	WRT FILE MARK	TST025
.WORD	00880.,00720.	:8.8-7.2	TAPE SPEED FWD	TST026
.WORD	00880.,00720.	:8.8-7.2	TAPE SPEED REV	TST027

:NOTE: TEST 26 AND 27 REQUIRE PRERECORDED 800BPI SKEW TAPE.

.SBTTL TU77 GAP TIME SPECIFICATION TABLE
 :THIS TABLE CONTAINS THE TU77 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)
TU77GTBL: .WORD	0504.,0446.	:5.04-4.46	GAP-0	0 MS
.WORD	0523.,0463.	:5.23-4.63	GAP-1	0.24 MS
.WORD	0539.,0474.	:5.39-4.74	GAP-2	0.48 MS
.WORD	0552.,0482.	:5.52-4.82	GAP-3	0.72 MS
.WORD	0562.,0486.	:5.62-4.86	GAP-4	0.96 MS
.WORD	0567.,0482.	:5.67-4.82	GAP-5	1.20 MS
.WORD	0570.,0478.	:5.70-4.78	GAP-6	1.44 MS
.WORD	0570.,0468.	:5.70-4.68	GAP-7	1.68 MS
.WORD	0566.,0452.	:5.66-4.52	GAP-10	1.92 MS
.WORD	0565.,0432.	:5.65-4.32	GAP-11	2.16 MS
.WORD	0550.,0407.	:5.50-4.07	GAP-12	2.40 MS
.WORD	0535.,0378.	:5.35-3.78	GAP-13	2.64 MS
.WORD	0510.,0410.	:5.10-4.10	GAP-14	2.88 MS
.WORD	0510.,0410.	:5.10-4.10	GAP-15	3.12 MS
.WORD	0510.,0410.	:5.10-4.10	GAP-16	3.36 MS
.WORD	0510.,0410.	:5.10-4.10	GAP-17	3.60 MS

1433
1434
1435
1436
1437
1438
1439 002124
1440 002124 000000 000000
1441 002130 036050 035230
1442 002134 001666 001546
1443 002140 001522 001356
1444 002144 002506 001332
1445 002150 007164 006344
1446 002154 000500 000360
1447 002160 000632 000454
1448 002164 002506 001332
1449 002170 000500 000360
1450 002174 000562 000512
1451 002200 002506 001332
1452 002204 003206 002056
1453 002210 003206 002056
1454 002214 002412 001666
1455 002220 002234 001522
1456 002224 002626 002354
1457 002230 002506 002234
1458 002234 004540 004230
1459 002240 004716 004552
1460 002244 023564 023110
1461 002250 024240 023730
1462 002254 004336 004172
1463 002260 004336 004172
1464
1465
1466
1467
1468
1469
1470
1471 002264 002544 002354
1472 002270 002652 002506
1473 002274 002722 002506
1474 002300 002710 002474
1475 002304 002570 002260
1476 002310 002556 002114
1477 002314 002532 002114
1478 002320 002532 002114
1479 002324 002506 002176
1480 002330 002474 002176
1481 002334 002474 002176
1482 002340 002474 002176
1483 002344 002474 002176
1484 002350 002474 002176
1485 002354 002474 002176
1486 002360 002474 002176
1487 002364

.SBTTL TIME SPECIFICATION TABLE
;THE BELOW TABLE WILL CONTAIN THE SPECIFIED FUNCTION TIMES IN TENS OF
;MICROSECONDS. NOTE THAT WHEN TIMES ARE TYPED THAT THEY ARE TYPED IN
;MICROSECONDS (BY APPENDING A 0).
;FORMAT IS
; .WORD MAX,MIN ;TIME IN MS FUNCTION TEST #
STTBL:
STIMTBL: .WORD 0 0 ;SPARE
.WORD 15400.,15000. ;154.0-150.0 WRITE FROM BOT TST001
.WORD 00950.,00870. ;9.5-8.7 WRITE START TST002
.WORD 00850.,00750. ;8.9-8.5 WRITE SHUTDOWN TST003
.WORD 01350.,00730. ;13.5-7.3 WRITE STLDOWN TST004
.WORD 03700.,03300. ;37.0-33.0 READ FROM BOT TST005
.WORD 00320.,00240. ;3.2-2.4 READ START TST006
.WORD 00410.,00300. ;4.1-3.00 READ SHUTDOWN TST007
.WORD 01350.,00730. ;13.5-7.3 READ SETTLEDOWN TST010
.WORD 00320.,00240. ;3.2-2.4 RD REV START TST011
.WORD 00370.,00330. ;3.7-3.3 RD REV SHTDWN TST012
.WORD 01350.,00730. ;13.5-7.3 RD REV STLDWN TST013
.WORD 01670.,01070. ;16.7-10.7 TRN RND DLY F-R TST014
.WORD 01670.,01070. ;16.7-10.7 TRN RND DLY R-F TST015
.WORD 01290.,00950. ;12.9-9.5 GAP SIZE STOP TST016
.WORD 01180.,00850. ;11.8-8.5 GAP SIZE STRT TST017
.WORD 01430.,01260. ;14.3-12.6 GAP SIZE INTER TST020
.WORD 01350.,01180. ;13.5-11.8 GAP CONSISANCY TST021
.WORD 02400.,02200. ;24.0-22.0 DAT TIME 800BPI TST022
.WORD 02510.,02410. ;25.1-24.1 DAT TIME 1600PE TST023
.WORD 10100.,09800. ;101.0-98.0 ERASE TST024
.WORD 10400.,10200. ;104.0-102.0 WRT FILE MARK TST025
.WORD 02270.,02170. ;22.7-21.7 TAPE SPEED FWD TST026
.WORD 02270.,02170. ;22.7-21.7 TAPE SPEED REV TST027

;NOTE: TEST 26 AND 27 REQUIRE PRERECORDED 800BPI SKEW TAPE.

.SBTTL GAP TIME SPECIFICATION TABLE
;THIS TABLE WILL CONTAIN 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.

GTIMTBL: .WORD MAX,MIN(10) ;TIME IN MS(10) GAP # DELAY IN MS(10)
.WORD 01380.,01260. ;13.8-12.6 GAP-0 0 MS
.WORD 01450.,01350. ;14.5-13.5 GAP-1 1.0 MS
.WORD 01490.,01350. ;14.9-13.5 GAP-2 2.0 MS
.WORD 01480.,01340. ;14.8-13.4 GAP-3 3.0 MS
.WORD 01400.,01200. ;14.0-12.0 GAP-4 4.0 MS
.WORD 01390.,01100. ;13.9-11.0 GAP-5 5.0 MS
.WORD 01370.,01100. ;13.7-11.0 GAP-6 6.0 MS
.WORD 01370.,01100. ;13.7-11.0 GAP-7 7.0 MS
.WORD 01350.,01150. ;13.5-11.5 GAP-10 8.0 MS
.WORD 01340.,01150. ;13.4-11.5 GAP-11 9.0 MS
.WORD 01340.,01150. ;13.4-11.5 GAP-12 10.0 MS
.WORD 01340.,01150. ;13.4-11.5 GAP-13 11.0 MS
.WORD 01340.,01150. ;13.4-11.5 GAP-14 12.0 MS
.WORD 01340.,01150. ;13.4-11.5 GAP-15 13.1 MS
.WORD 01340.,01150. ;13.4-11.5 GAP-16 14.1 MS
.WORD 01340.,01150. ;13.4-11.5 GAP-17 15.1 MS

ENDTBL:

1488
 1489
 1490 002364 015701
 1491 002366 015731
 1492 002370 015753
 1493 002372 015773
 1494 002374 016015
 1495 002376 016041
 1496 002400 016063
 1497 002402 016102
 1498 002404 016124
 1499 002406 016147
 1500 002410 016171
 1501 002412 016216
 1502 002414 016245
 1503 002416 016276
 1504 002420 016327
 1505 002422 016355
 1506 002424 016404
 1507 002426 016434
 1508 002430 016457
 1509 002432 016503
 1510 002434 016530
 1511 002436 016552
 1512 002440 016575
 1513 002442 016617
 1514
 1515
 1516 002444 010000
 1517 002446 010274
 1518 002450 010360
 1519 002452 010436
 1520 002454 010526
 1521 002456 010634
 1522 002460 010720
 1523 002462 011004
 1524 002464 011104
 1525 002466 011224
 1526 002470 011322
 1527 002472 011432
 1528 002474 011572
 1529 002476 011664
 1530 002500 011772
 1531 002502 012066
 1532 002504 012176
 1533 002506 012316
 1534 002510 012622
 1535 002512 012752
 1536 002514 013102
 1537 002516 013222
 1538 002520 013556
 1539 002522 013714

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

.WORD A.T000
 .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 A.T022
 .WORD A.T023
 .WORD A.T024
 .WORD A.T025
 .WORD A.T026
 .WORD A.T027

;TABLE OF TEST STARTING ADDRESSES
 TSTTBL:

.WORD TST000
 .WORD TST001
 .WORD TST002
 .WORD TST003
 .WORD TST004
 .WORD TST005
 .WORD TST006
 .WORD TST007
 .WORD TST010
 .WORD TST011
 .WORD TST012
 .WORD TST013
 .WORD TST014
 .WORD TST015
 .WORD TST016
 .WORD TST017
 .WORD TST020
 .WORD TST021
 .WORD TST022
 .WORD TST023
 .WORD TST024
 .WORD TST025
 .WORD TST026
 .WORD TST027

```

154J 002524 000000      TIB:      .WORD      0
1541      ;ROUTINE TO LOAD SOFTWARE SWR
1542
1543 002526 022737 000176 001000 GTSWR:  CMP      #SWREG,SWR      ;BRANCH IF SOFTWARE SWR
1544 002534 001027      BNE      2$              ;NOT INVOKED
1545 002536 004737 003062      JSR      PC,..SAVE      ;SAVE REGISTERS ON THE STACK
1546 002542 000004 016646      TYPE,L.SWR
1547 002546 017702 176226      MOV      @SWR,R2
1548 002552 004737 003134      JSR      PC,TYPOCT
1549 002556 000004 016655      TYPE,L.NEW
1550 002562 004737 004000      JSR      PC,..INPUT      ;GET USER INPUT
1551 002566 122737 000015 001272 CMPB     #CR,@INBUF      ;EXIT IF FIRST CHAR IS <CR>
1552 002574 001405      BEQ     1$
1553 002576 004737 003564      JSR      PC,CNVTAO      ;CONERT ASCII TO OCTAL
1554 002602 013777 001122 176170 MOV      @OCTALO,@SWR    ;SET NEW SWITCH REG CONTENTS
1555 002610 004737 003104      1$:     JSR      PC,..RESTORE
1556 002614 000207      2$:     RTS      PC
1557
1558
1559      .SBTTL  PROGRAM SUBROUTINES
1560      .SBTTL  TYPE SUBROUTINE
1561      ;;ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1562      ;;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1563      ;;CALL: TYPE      ;;A TRAP TYPE INSTRUCTION
1564      ;;      MESADR      ;;MESADR IS FIRST ADDRESS OF ASCIZ STRING
1565
1566      ;;TAGS USED BY THE TYPE ROUTINE BELOW
1567      $HT=11      ;;HORIZONTAL TAB
1568 002616 000      $NULL: .BYTE 0      ;;CONTAINS NULL CHARACTER
1569 002617 002      $FILL: .BYTE 2      ;;CONTAINS # OF FILLER CHARACTERS
1570 002620 000      $TPFLG: .BYTE 0      ;;CONTAINS TELEPRINTER AVAILABLE FLAG
1571      ;;0/377 = AVAIL/NOT AVAIL
1572 002621 000      $TKFLG: .BYTE 0      ;;CONTAINS KEYBOARD AVAILABLE FLAG
1573 002622 177564      $TPS:   .WORD 177564      ;;ADDRESS OF TELEPRINTER STATUS REGISTER
1574 002624 177566      $TPB:   .WORD 177566      ;;ADDRESS OF TELEPRINTER DATA BUFFER
1575 002626 000      $CHARCNT: .BYTE 0      ;;CONTAINS # OF CHARS TYPED
1576 002627 000      $CNTRLO: .BYTE 0      ;;CONTAINS CONTROL 0 CHAR (IF TYPED)
1577 002630 005015 000      $CRLF:  .ASCIZ <15><12>
1578      .EVEN
1579 002634 000000      RDSW:   .WORD 0
1580
1581 002636 010046      .TYPE:  MOV      R0,-(SP)      ;;SAVE R0
1582 002640 017600 000002      MOV      @2(SP),R0      ;;GET MESSAGE ADDRESS
1583 002644 062766 000002 000002 ADD      #2,2(SP)      ;;ADJUST RETURN PC
1584 002652 105037 002627      CLRB    $CNTRLO
1585
1586 002656 105737 002627      TYPE1:  TSTB    $CNTRLO      ;;BRANCH IF CONTROL 0(^O) WASN'T TYPED
1587 002662 001410      BEQ     TYPE2
1588 002664 000004 002630      TCRLF:  TYPE,$CRLF      ;;TYPE <CR><LF>
1589 002670 105737 002634      TSTB    RDSW
1590 002674 100006      BPL     TYPE3
1591 002676 005037 002634      CLR     RDSW
1592 002702 000207      RTS     PC
1593 002704 112046      TYPE2:  MOVB   (R0)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1594 002706 001003      BNE     TYPE4      ;;BRANCH IF NOT THE TERMINATOR
1595 002710 005726      TST     (SP)+      ;;POP TERMINATOR CHAR OFF THE STACK

```

```

1596 002712 012600          TYPE3: MOV      (SP)+,R0          ;;RESTORE R0
1597 002714 000002          RTI                          ;;RETURN TO CALLER
1598
1599 002716 122716 000011  TYPE4: CMPB     #SHT,(SP)          ;;BRANCH IF HORIZONTAL TAB <HT>
1600 002722 001445          BEQ      9$
1601 002724 004737 002756  JSR      PC,5$          ;;TYPE CHARACTER
1602 002730 122726 000012  3$:  CMPB     #12,(SP)+          ;;CHECK IF CHARACTER WAS A LINE FEED
1603 002734 001350          BNE     TYPE1          ;;BRANCH IF NOT LINE FEED
1604 002736 013746 002616  MOV      $NULL,-(SP)      ;;GET # OF FILLERS REQUIRED AND FILLER
1605                                     ;;CHARACTER.
1606
1607 002742 105366 000001  4$:  DECB     1(SP)          ;;DECREMENT FILLERS REQ. COUNT
1608 002746 002770          BLT     3$              ;;BRANCH IF NO MORE FILLERS ARE REQUIRED
1609 002750 004737 002756  JSR      PC,5$          ;;TYPE FILLER CHARACTER
1610 002754 000772          BR      4$
1611
1612 002756 105777 177640  5$:  TSTB     @STPS          ;;WAIT FOR OUTPUT DEVICE
1613 002762 100375          BPL     -4
1614 002764 122737 000017 002627  CMPB     #17,@$CNTRLO      ;;CHECK IF CONTROL O WAS TYPED
1615 002772 001403          BEQ     6$              ;;STOP TYPING MESSAGE IF ^O WAS TYPED
1616 002774 116677 000002 177622  MOVB     2(SP),@STPB        ;;OUTPUT CHARACTER
1617 003002 122766 000015 000002  6$:  CMPB     #15,2(SP)          ;;BRANCH IF NOT <CR>
1618 003010 001003          BNE     7$
1619 003012 105037 002626  CLRB     $CHARCNT          ;;CLEAR CHARACTERS TYPED COUNT
1620 003016 000406          BR      8$
1621 003020 122766 000012 000002  7$:  CMPB     #12,2(SP)          ;;BRANCH IF <LF> OR 'NULL'
1622 003026 002002          BGE     8$
1623 003030 105237 002626  INCB     $CHARCNT          ;;INCREMENT CHARACTER TYPED COUNT
1624 003034 000207          RTS      PC
1625
1626                                     ;;HORIZONTAL TAB <HT> PROCESSER
1627 003036 112716 000040  9$:  MOVB     #40,(SP)          ;;LOAD 'SPACE'
1628 003042 004737 002756  10$: JSR      PC,5$          ;;TYPE 'SPACE'
1629 003046 132737 000007 002626  BITB     #7,$CHARCNT        ;;TYPE SPACES UNTIL A MULTIPLE
1630 003054 001372          BNE     10$             ;;OF 8 CHARACTERS HAVE BEEN TYPED
1631 003056 105726          TSTB     (SP)+          ;;POP SPACE
1632 003060 000676          BR      TYPE1          ;;GET NEXT CHARACTER
1633
1634                                     ;SUBROUTINE TO SAVE GENERAL REGISTERS ON THE STACK
1635                                     ;CALL: SAVE
1636 003062 010546          .SAVE: MOV     R5,-(SP)          ;SAVE REGISTERS ON THE STACK
1637 003064 010446          MOV     R4,-(SP)
1638 003066 010346          MOV     R3,-(SP)
1639 003070 010246          MOV     R2,-(SP)
1640 003072 010146          MOV     R1,-(SP)
1641 003074 010046          MOV     R0,-(SP)
1642 003076 016646 000014          MOV     14(SP),-(SP)        ;GET RETURN PC
1643 003102 000207          RTS      PC              ;RETURN
1644
1645                                     ;SUBROUTINE TO RESTORE GENERAL REGISTERS FROM THE STACK
1646                                     ;CALL: RESTORE
1647 003104 012666 000014          .RESTORE:MOV   (SP)+,14(SP)    ;MOVE RETURN PC
1648 003110 012600          MOV     (SP)+,R0          ;RESTORE REGISTERS
1649 003112 012601          MOV     (SP)+,R1
1650 003114 012602          MOV     (SP)+,R2
1651 003116 012603          MOV     (SP)+,R3

```

```

1652 003120 012604      MOV      (SP)+,R4
1653 003122 012605      MOV      (SP)+,R5
1654 003124 000207      RTS      PC                ;RETURN
1655
1656      ;SUBROUTINE TO CONVERT OCTAL DATA TO ASCII
1657      ;CALL: MOV      NUMBER,R2          ;MOVE NUMBER TO R2
1658      ;      JSR      PC,CNVOCT
1659
1660 003126 110637 001133  CNVOCT: MOVB     SP,TYPFLG          ;SET DO NOT TYPE FLAG
1661 003132 000402      BR      CNVTO
1662
1663      .SBTTL      OCTAL TO ASCII & TYPE ROUTINE
1664      ;SUBROUTINE TO CONVERT OCTAL NUMBER TO ASCII AND TYPE IT OUT
1665      ;CALL: MOV      NUMBER,R2          ;PUT # IN R2
1666      ;      JSR      PC,TYPOCT         ;CALL ROUTINE
1667
1668 003134 105037 001133  TYPOCT: CLRB     @TYPFLG          ;SET TYPE FLAG
1669 003140  CNVTO:
1670 003140 004737 003062      JSR      PC,.SAVE              ;SAVE REGISTERS ON THE STACK
1671 003144 012704 001152      MOV      #ODIGITS,R4          ;SET PTR TO OUTPUT
1672 003150 005003      CLR      R3                  ;R3 WILL CONTAIN OCTAL DIGIT
1673 003152 010201      MOV      R2,R1              ;GET # TO BE TYPED
1674 003154 006302 1$:      ASL      R2                  ;SHIFT #
1675 003156 006103      ROL      R3
1676 003160 012700 000006      MOV      #6,R0              ;SET DIGIT COUNTER
1677 003164 000404      BR      3$
1678
1679 003166 006302 2$:      ASL      R2                  ;SHIFT # 3 PLACES LEFT
1680 003170 006103      ROL      R3
1681 003172 005301      DEC      R1
1682 003174 001374      BNE     2$
1683 003176 012701 000003 3$:      MOV      #3,R1              ;SET SHIFT COUNTER
1684 003202 116324 001140      MOVB     DIGTAB(R3),(R4)+     ;MOVE ASCII EQUIV TO OUTPUT
1685 003206 005003      CLR      R3
1686 003210 005300      DEC      R0                  ;DECREMENT DIGIT COUNT
1687 003212 001365      BNE     2$                  ;GET NEXT DIGIT
1688 003214 105737 001133      TSTB     @TYPFLG            ;BRANCH IF ASCII IS
1689 003220 001002      BNE     4$                  ;NOT TO BE TYPED
1690 003222 000004 001152      TYPE,ODIGITS
1691 003226      4$:
1692 003226 004737 003104      JSR      PC,.RESTORE         ;RESTORE REGISTERS FROM THE STACK
1693 003232 000207      RTS      PC
1694
1695
1696      ;SUBROUTINE TO CONVERT OCTAL DATA TO DECIMAL ASCII
1697      ;CALL: MOV      NUMBER,R2          ;MOVE NUMBER TO R2
1698      ;      JSR      PC,CNVDEC
1699
1700 003234 110637 001133  CNVDEC: MOVB     SP,@TYPFLG          ;SET DO NOT TYPE FLAG
1701 003240 000402      BR      CNVTD
1702
1703      .SBTTL      OCTAL TO DECIMAL & TYPE ROUTINE
1704      ;THIS ROUTINE CONVERTS AN OCTAL # TO DECIMAL ASCII AND TYPES IT OUT
1705      ;CALL: MOV      NUMBER,R2          ;PUT # IN R2
1706      ;      JSR      PC,TYPDEC         ;CALL ROUTINE
1707 003242 105037 001133  TYPDEC: CLRB     @TYPFLG          ;SET TYPE FLAG
  
```

```
1708 003246          CNVTD:
1709 003246 004737 003062      JSR   PC,..SAVE          ;SAVE REGISTERS ON THE STACK
1710 003252 005000          CLR   R0                ;R0 IS INDEX TO DECIMAL CONSTANT
1711 003254 012704 001152      MOV   #ODIGITS,R4       ;SET OUTPUT PTR
1712 003260 005003          1$: CLR   R3                ;R3 CONTAINS DECIMAL DIGIT
1713 003262 166002 003342      2$: SUB  DCONST(R0),R2   ;SUBTRACT DECIMAL CONSTANT UNTIL
1714 003266 103402          BLO  3$                ;INPUT # GOES NEGATIVE
1715 003270 005203          INC  R3                ;KEEPING TRACK OF SUBTRACTIONS
1716 003272 000773          BR   2$
1717 003274 066002 003342      3$: ADD  DCONST(R0),R2   ;ADD BACK CONSTANT WHEN NEGATIVE
1718 003300 116324 001140      MOVB  DIGTAB(R3),(R4)+  ;MOVE ASCII EQUIVALENT
1719 003304 062700 000002      ADD  #2,R0              ;NEXT CONSTANT
1720 003310 005760 003342      TST  DCONST(R0)        ;UNTIL ALL CONSTANTS DONE
1721 003314 001361          BNE  1$
1722 003316 112724 000060      MOVB  #'0,(R4)+        ;LAST DIGIT IS 0
1723 003322 105737 001133      TSTB  @#TYPFLG         ;BRANCH IF ASCII IS
1724 003326 001002          BNE  4$                ;NOT TO BE TYPED
1725 003330 000004 001152      TYPE,ODIGITS
1726 003334          4$:
1727 003334 004737 003104      JSR   PC,..RESTORE     ;RESTORE REGISTERS FROM THE STACK
1728 003340 000207          RTS   PC
1729
1730 003342 023420          DCONST: .WORD 10000.
1731 003344 001750          .WORD 1000.
1732 003346 000144          .WORD 100.
1733 003350 000012          .WORD 10.
1734 003352 000001          .WORD 1.
1735 003354 000000          .WORD 0                ;TERMINATOR
1736
1737          .SBTTL          TYPE SPECIFIED TIMES ROUTINE
1738          ;THIS SUBROUTINE OUTPUTS THE TIME SPECIFICATIONS FOR THE TEST
1739          ;AND ALSO THE ACTUAL TIME RECORDED (ATIME)
1740          ;FORMAT OF LINE TYPED
1741          ;RANGE=<AAAAAA-BBBBBB>          ACTUAL=CCCCCC
1742          ;WHERE:          AAAAAA IS MAXIMUM TIME FOR TEST (STIMTBL(TSTNUMX4)).
1743          ;          BBBBBB IS MINIMUM TIME FOR TEST (STIMTBL(TSTNUMX4+2)).
1744          ;          CCCCCC IS ACTUAL TIME RECORDED BY TEST (ATIME).
1745          ;CALL:  MOVB  TEST NUMBER,R2 ;LOAD TEST NUMBER
1746          ;          MOV   #TIME,@#ATIME ;MOVE TIME TO ATIME
1747          ;          JSR   PC,OUTSPC
1748          OUTSPC: MOV   R2,-(SP) ;SAVE R2 & R3 ON THE STACK
1749          MOV   R3,-(SP)
1750          ASL  R2 ;MULTIPLY TEST # TIMES 4
1751          ASL  R2 ;TO FORM INDEX INTO STIMTBL
1752          MOV  R2,R3 ;R3 CONTAINS INDEX INTO TABLE
1753          TYPE,L.RNG
1754          MOV  STIMTBL(R3),R2 ;GET MAXIMUM SPEC TIME
1755          JSR  PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
1756          TYPE,DASH
1757          MOV  STIMTBL+2(R3),R2 ;GET MINIMUM TIME
1758          JSR  PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
1759          TYPE,ANGTAB
1760          TYPE,L.ACT
1761          MOV  @#ATIME,R2 ;GET ACTUAL TIME
1762          JSR  PC,TYPDEC ;CONVERT TO DECIMAL & TYPE
1763          TYPE,CRLF
```

1764 003444 012603
 1765 003446 012602
 1766 003450 000207
 1767
 1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776 003452 010246
 1777 003454 010346
 1778 003456 113703 001124
 1779 003462 006303
 1780 003464 006303
 1781 003466 000004 015661
 1782 003472 016302 002264
 1783 003476 004737 003242
 1784 003502 000004 001413
 1785 003506 016302 002266
 1786 003512 004737 003242
 1787 003516 000004 001420
 1788 003522 000004 015671
 1789 003526 013702 001016
 1790 003532 004737 003242
 1791 003536 000004 015336
 1792 003542 113702 001124
 1793 003546 004737 003134
 1794 003552 000004 001402
 1795 003556 012603
 1796 003560 012602
 1797 003562 000207
 1798
 1799
 1800
 1801
 1802 003564
 1803 003564 004737 003062
 1804 003570 012700 001272
 1805 003574 012701 001122
 1806 003600 005011
 1807 003602 005061 000002
 1808 003606 122710 000015
 1809 003612 001414
 1810 003614 112002
 1811 003616 042702 177770
 1812 003622 012703 000003
 1813 003626 006311
 1814 003630 006161 000002
 1815 003634 005303
 1816 003636 001373
 1817 003640 050211
 1818 003642 000761
 1819 003644

```

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:  MOVB  #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 ,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
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$:

```

```
1820 003644 004737 003104 JSR PC,RESTORE ;RESTORE REGISTERS FROM THE STACK
1821 003650 000207 RTS PC ;RETURN
1822
1823 .SBTTL PUBLISH SUBROUTINE
1824 ;THE PUBLISH SUBROUTINE AVERAGES THE RECORDED TIMES FOR EACH TEST IT-
1825 ;ERATION (IF 16. ITERATIONS) AND PLACES THE AVERAGE RESULT IN 'ATIME'.
1826 ;IT TYPES THE NAME OF THE FUNCTION THAT WAS TIMED,THE TIME SPEC-
1827 ;IFICATION AND THE ACTUAL TIME .
1828
1829 PUBLISH:
1830 003652 004737 003062 JSR PC,SAVE ;SAVE REGISTERS ON THE STACK
1831 003656 012700 001020 MOV #ATIMTBL,R0 ;GET TABLE ADDRESS CONTAINING TIMES
1832 003662 113701 001125 MOV #ITCNT,R1 ;GET # OF ENTRIES (GIVEN BY ITERATION COUNT)
1833 003666 122701 000001 CMPB #1,R1 ;BRANCH IF SINGLE ITERATION
1834 003672 001423 BEQ 4$
1835 003674 005002 CLR R2 ;CLEAR 'SUM' REGISTERS
1836 003676 005003 CLR R3
1837 003700 122701 000020 CMPB #16.,R1 ;BRANCH IF 16. ITERATIONS
1838 003704 001402 BEQ 1$
1839 003706 000000 HALT ;ITERATION COUNT MUST BE 1 OR 16.
1840 003710 000777 BR . ;DO NOT CHANGE POSIT OF SW11
1841 ;WHEN TEST IS RUNNING.
1842
1843 003712 062002 1$: ADD (R0)+,R2 ;SUM INDIVIDUAL TIMES
1844 003714 005503 ADC R3
1845 003716 005301 DEC R1
1846 003720 001374 BNE 1$
1847
1848 003722 012700 000004 2$: MOV #4,R0
1849 003726 006203 3$: ASR R3 ;SHIFT TIME IN R3 & R2 4 PLACES
1850 003730 006002 ROR R2 ;RIGHT = DIVIDE BY 16.
1851 003732 005300 DEC R0
1852 003734 001374 BNE 3$
1853 003736 010237 001016 MOV R2,#ATIME ;MOVE AVERAGED TIMES
1854
1855 003742 113700 001126 4$: MOV #TSTNUM,R0 ;GET TEST #
1856 003746 006300 ASL R0
1857 003750 016037 002364 003760 MOV #NAMPTR(R0),5$ ;GET TEST NAME STRING ADDRESS
1858 003756 000004 TYPE
1859 003760 000000 5$: .WORD 0
1860 003762 113702 001126 MOV #TSTNUM,R2 ;GET TEST #
1861 003766 004737 003356 JSR PC,OUTSPC ;OUTPUT TIMES
1862 003772 004737 003104 JSR PC,RESTORE ;RESTORE REGISTERS FROM THE STACK
1863 003776 000207 RTS PC
1864
1865 .SBTTL INPUT SUBROUTINE
1866 ;SUBROUTINE TO GET TTY INPUT
1867 ;CALL: JSR PC,INPUT
1868 ;INPUT DATA IS RETURNED IN BUFFER BEGINNING AT INBUF.
1869
1870 004000 010046 .INPUT: MOV R0,-(SP) ;SAVE R0 ON THE STACK
1871 004002 012700 001272 1$: MOV #INBUF,R0
1872 004006 105737 177560 2$: TSTB #TKS
1873 004012 100375 BPL 2$
1874
1875 004014 113746 177562 MOVB #TKB,-(SP) ;GET CHARACTER
```

```
1876 004020 042716 000200      BIC      #200,(SP)
1877 004024 122716 000177      CMPB     #177,(SP)      ;CHECK RUBOUT
1878 004030 001004      BNE      3$
1879 004032 124026      CMPB     -(RO),(SP)+    ;REMOVE CHARACTER FROM INPUT
1880 004034 000004 001405      TYPE,BKSLSH
1881 004040 000762      BR       2$            ;WAIT FOR NEXT CHARACTER
1882 004042 122716 000025      3$:      CMPB     #CNTRLU,(SP) ;CHECK CONTROL U (^U)
1883 004046 001004      BNE      4$
1884 004050 005726      TST     (SP)+
1885 004052 000004 001402      TYPE,CRLF
1886 004056 000751      BR       1$
1887 004060 122716 000003      4$:      CMPB     #CNTRLC,(S') ;BRANCH IF NOT CONTROL C
1888 004064 001003      BNE      40$
1889 004066 000005      RESET
1890 004070 000137 006274      JMP      @#INIT        ;RESTART PROGRAM
1891 004074 111637 001407      40$:     MOVB     (SP),@#ECHO
1892 004100 111620      MOVB     (SP),(RO)+
1893 004102 122726 000015      CMPB     #CR,(SP)+
1894 004106 001403      BEQ     5$
1895 004110 000004 001407      TYPE,ECHO
1896 004114 000734      BR       2$
1897 004116 000004 001402      5$:      TYPE,CRLF
1898 004122 012600      MOV     (SP)+,RO
1899 004124 000207      RTS     PC

1901      ;KEYBOARD INTERRUPT SERVICE ROUTINE
1902 004126 113746 177562      TKISR:  MOVB     @#TKB,-(SP) ;GET TYPED CHARACTER
1903 004132 042716 000200      BIC     #200,(SP)      ;STRIP PARITY BIT
1904 004136 122716 000017      CMPB     #CNTRLO,(SP) ;BRANCH IF NOT CONTROL O (^O)
1905 004142 001002      BNE     1$
1906 004144 111637 002627      MOVB     (SP),#CNTRLO ;SET CONTROL O INDICATOR IN TYPE ROUTINE
1907
1908 004150 122716 000003      1$:      CMPB     #3,(SP)      ;BRANCH IF NOT CONTROL C (^C)
1909 004154 001007      BNE     2$
1910 004156 023727 000042 013504      CMP     @#42,#SENDAD ;INHIBIT ^C IF ACT11 QV OR AA
1911 004164 001403      BEQ     2$
1912 004166 000005      RESET
1913 004170 000137 006274      JMP     @#INIT        ;RESTART PROGRAM
1914
1915 004174 122716 000001      2$:      CMPB     #CNTRLA,(SP) ;BRANCH IF NOT ^A
1916 004200 001011      BNE     3$
1917 004202 022737 000176 001000      CMP     #SWREG,SWR    ;BRANCH IF HARDWARE SWR IS INVOKED
1918 004210 001010      BNE     4$
1919 004212 012737 177570 001000      MOV     #177570,SWR   ;INVOKE HARDWARE SWR
1920 004220 000004 014314      TYPE,M.HSWR
1921 004224 122716 000007      3$:      CMPB     #CNTRLG,(SP) ;BRANCH IF NOT ^G
1922 004230 001005      BNE     5$
1923 004232 012737 000176 001000      4$:      MOV     #SWREG,SWR   ;INVOKE SOFTWARE SWR
1924 004240 004737 002526      JSR     PC,GTSWR     ;GET NEW SWITCH REGISTER
1925 004244 005726      5$:      TST     (SP)+
1926 004246 000002      RTI
;RETURN TO CALLER
```

```

1927          .SBTTL          ERROR SERVICE ROUTINES
1928          ;ROUTINE TO PROCESS ERROR TRAPS (TRAPS TO 4)
1929 004250 000000  ERRTRP: HALT
1930
1931          ;ERROR SERVICE ROUTINE
1932          ;THIS ROUTINE PROCESSES TWO TYPES OF ERRORS (OUT OF RANGE AND HARDWARE)
1933          ;THE CALLS FOR AN OUT OF RANGE ERROR ARE <HLT+1>,<HLT+2> AND, FOR A
1934          ;HARDWARE ERROR THE CALL IS <HLT>.
1935
1936 004252 004737 003062  .HLT:  JSR    PC,SAVE          ;SAVE REGISTERS ON THE STACK
1937 004256 110637 001127 1$:    MOVB   SP,@MERFLG      ;SET ERROR FLAG
1938 004262 032777 020000 174510  BIT    #SW13,@SWR        ;BRANCH IF NO TYP0UT
1939 004270 001075          BNE    4$
1940 004272 000004 015137          TYPE,E.HDR
1941 004276 113702 001126          MOVB   @#TSTNUM,R2        ;GET TEST #
1942 004302 004737 003134          JSR    PC,TYP0CT        ;AND TYPE IT
1943 004306 016600 000016          MOV    16(SP),R0        ;GET RETURN PC
1944 004312 162700 000002          SUB    #2,R0           ;NOW PC OF HLT CALL
1945 004316 111000          MOVB   (R0),R0          ;NOW HLT CALL ITSELF
1946 004320 001417          BEQ   2$               ;BRANCH IF HLT
1947 004322 000004 015222          TYPE,E.HDR2
1948 004326 122700 000002          CMPB  #2,R0            ;BRANCH IF NOT HLT+2
1949 004332 001005          BNE   10$
1950 004334 004737 003452          JSR    PC,OUTGAP        ;TYPE GAP SPECIFIED TIMES
1951 004340 000004 001402          TYPE,CRLF
1952 004344 000447          BR    4$
1953 004346 004737 003356 10$:   JSR    PC,OUTSPC        ;TYPE SPECIFIED TIMES
1954 004352 000004 001402          TYPE,CRLF
1955 004356 000442          BR    4$
1956 004360 016500 000014 2$:   MOV    ER(R5),R0
1957 004364 032765 002300 000032  BIT    #PE1600,TC(R5)
1958 004372 001403          BEQ   20$
1959 004374 042700 102100          BIC   #102100,R0
1960 004400 000402          BR    21$
1961 004402 042700 102300 20$:   BIC   #102300,R0
1962 004406 005700 21$:   TST   R0
1963 004410 001003          BNE   22$
1964 004412 000004 015113          TYPE,E.SFT            ;TYPE SOFT ERROR MESSAGE
1965 004416 000434          BR    6$
1966
1967 004420 000004 015147 22$:   TYPE,E.HDR1
1968 004424 010500          MOV    R5,R0            ;GET FIRST ADDRESS OF REGS.
1969 004426 012701 000007          MOV    #7,R1           ;TYPE FIRST 7 REGS.
1970 004432 012002 3$:   MOV    (R0)+,R2        ;GET REG CONTENTS
1971 004434 004737 003134          JSR    PC,TYP0CT        ;AND TYPE IT
1972 004440 000004 001415          TYPE,SPACE2
1973 004444 005301          DEC   R1
1974 004446 001371          BNE   3$
1975 004450 016502 000032          MOV    TC(R5),R2        ;GET CONTENTS OF TC REGISTER
1976 004454 004737 003134          JSR    PC,TYP0CT
1977 004460 000004 001402          TYPE,CRLF
1978
1979 004464 032777 001000 174306 4$:   BIT    #SW09,@SWR        ;BRANCH IF NO RING THE BELL
1980 004472 001402          BEQ   5$
1981 004474 000004 001411          TYPE,BELL
1982 004500 005777 174274 5$:   TST   @SWR            ;HALT ON ERROR?

```

LZTEECO TMO3-TE16/TU77 DFT
CZTEFC.P11 22-AUG-79 08:40

MACY11 30A(1052) 22-AUG-79 09:00 PAGE 46
ERROR SERVICE ROUTINES

SEQ 0046

1983 004504 100001
1984 004506 000000
1985 004510
1986 004510 004737 003104
1987 004514 000002
1988
1989

6S:

BPL 6S
HALT

JSR PC, .RESTORE
RTI

;RESTORE REGISTERS FROM THE STACK
;RETURN

1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045

```
.SBTTL          SCOPE SUBROUTINE
:SCOPE ROUTINE
:THIS ROUTINE IS ENTERED UPON COMPLETION OF EACH SUBTEST
:THE SCOPE ROUTINE:
:   REPEATS TEST IF SW14 IS SET
:   STORES ACTUAL TIME FOR FUNCTION IN TIME TABLE (ATIMTBL)
:   PUBLISHES TIME IF SW10=0
:   UPDATES ITERATION COUNT AND IF ITERATIONS COMPLETE CONTINUES
:   TO NEXT TEST, OTHERWISE REPEATS TEST.
:   DELAYS BEFORE CONTINUING OR REPEATING TEST.
:   INITIALIZES DRIVE
:RETURNS:      R5=BASE ADDRESS OF TM03 REGISTERS (ADDRESS OF CS1)
:              R1='DS' REG ADDRESS
:              R0='FC' REG ADDRESS

.SCOPE: MOV     @#TMBASE,R5          ;SET R5 TO FIRST TM REG
        BIT     #SW14,@SWR         ;BRANCH IF CONTINUOUS LOOP
        BEQ     2$                 ;NOT DESIRED
1$:     MOV     @SWR,R1             ;GET SWITCHES
        BIC     #177740,R1         ;CLEAR ALL BUT TEST #
        BEQ     11$                ;BRANCH IF ALL SELECTED
        CMPB   R1,@#TSTNUM         ;BRANCH IF RUNNING SELECTED TEST
        BEQ     11$
11$:    MOV     #TST000,SCPADR      ;RESTART AT TST000
        JSR    PC,DELAY            ;DELAY 350 MS
        JSR    PC,RHINIT           ;INIT
        CLRB   @#ERFLG            ;CLEAR ERROR FLAG
        MOV    SCPADR,(SP)
        MOV    R5,R1
        ADD    #DS,R1              ;ADDRESS OF 'DS' REG IS IN R1
        MOV    R5,R0
        ADD    #FC,R0              ;ADDRESS OF 'FC' REG IS IN R0
        RTI

2$:     TSTB   @#ERFLG             ;BRANCH IF ERROR FLAG IS SET
        BNE    3$
        MOVB  @#ITCNT,R0           ;GET ITERATION COUNT
        ASL   R0                   ;STORE TIME IN TABLE
        MOV   @#ATIME,ATIMTBL(R0)
3$:     INCB   @#ITCNT             ;INCREMENT ITERATION COUNT
        TSTB  @#PSCNT             ;INHIBIT ITERATIONS ON
        BEQ   4$                   ;ON FIRST PASS
        BIT   #SW11,@SWR          ;BRANCH IF SINGLE ITERATION DESIRED
        BNE   4$
        CMPB  #16,@#ITCNT         ;BRANCH IF ITERATIONS INCOMPLETE
        BNE   1$
4$:     BIT   #37,@SWR             ;IF TEST SELECTED IS TEST 0
        BNE   42$                 ;TREAT AS ALL TESTS
        MOV   (SP),@#SCPADR        ;SET SCOPE ADDRESS TO NEXT TEST
        BIT   #SW10,@SWR          ;BRANCH IF NO PUBLICATION DESIRED
        BNE   5$
5$:     JSR   PC,PUBLISH           ;GO PUBLISH TEST DATA
        CLRB  @#ITCNT             ;RESET ITERATION COUNT
        BR    1$

.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      004730 005004      TIMON: CLR      R4              ;CLEAR TIME COUNT
2056      004732 012703 000024      MOV      #24,R3          ;SET POLARITY TO '0' STATE
2057      004736 032765 000100 000024      BIT      #OSC,MR(R5)      ;BRANCH IF POLARITY IS '0'
2058      004744 001405      BEQ      2$
2059      004746 032765 000100 000024 1$: BIT      #OSC,MR(R5)      ;WAIT FOR OSCILLATOR TO RETURN
2060      004754 001374      BNE      1$
2061      004756 000405      BR       4$
2062
2063      004760 005403      2$: NEG      R3              ;NEGATE PREV POLARITY INDICATOR
2064      004762 032765 000100 000024 3$: BI*     #OSC,MR(R5)      ;WAIT FOR OSCILLATOR TO RETURN
2065      004770 001774      BEQ      3$              ;TO '1' STATE
2066      004772 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      004774 032765 000100 000024 TIMER1: BIT      #OSC,MR(R5)      ;BRANCH IF CURRENT STATE IS '0'
2079      005002 001406      BEQ      TIMER          ;GO INCREMENT TIME
2080      005004 000112      JMP      (R2)          ;RETURN TO TEST
2081
2082      .=TIMER1+24
2083      005020 005403      TIMER: NEG      R3              ;NEGATE PREV STATE INDICATOR
2084      005022 005204      INC      R4              ;INCREMENT 'TICK' COUNT
2085      005024 100401      BMI      TIMERR          ;BRANCH ON OVERFLOW
2086      005026 000112      JMP      (R2)          ;RETURN TO TEST
2087      005030 000004 015250      TIMERR: TYPE,E.TIMOV      ;TYPE 'TIMER OVERFLOWED'
2088      005034 104400      HLT
2089      005036 000177 173740      JMP      @SCPADR        ;REPORT HARDWARE ERROR
2090      ;              ;RETURN TO BEGINNING OF TEST
2091      .=TIMER+24
2092      ;ENTER HERE VIA JMP TIMER(R3) WHEN R3 +24 (PREV STATE 0)
2093      005044 032765 000100 000024 TIMERO: BIT      #OSC,MR(R5)      ;BRANCH IF CURRENT STATE '1'
2094      005052 001362      BNE      TIMFR
2095      005054 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(RO)) AND THE LOW LIMIT (STIMTBL+2(RO)).
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      TIMOK:
2106      005056 004737 003062      JSR      PC,SAVE      ;SAVE REGISTERS ON THE STACK
2107      005062 013700 001012      MOV      @#OSCTIM,R0   ;GET TIME PER TICK
2108      005066 010401              MOV      R4,R1        ;GET TICKS COUNT
2109      005070 005002              CLR      R2           ;CLEAR SUMMING REGISTERS
2110      005072 005003              CLR      R3
2111      005074 060002      1$:      ADD      R0,R2        ;MULTIPLY TIME PER TICK
2112      005076 005503              ADC      R3           ;BY TICK COUNT
2113      005100 005301              DEC      R1
2114      005102 001374              BNE      1$
2115      005104 010246              MOV      R2,-(SP)     ;DIVIDE COUNT BY 10.
2116
2117      005106 010346              MOV      R3,-(SP)
2118      005110 012746 000012      MOV      #10,-(SP)
2119      005114 004737 005376      JSR      PC,DIVIDE
2120      005120 005726              TST      (SP)+        ;DISCARD REMAINDER
2121      005122 012637 001016      MOV      (SP)+,@#ATIME ;STORE QUOTIENT
2122      005126 113700 001126      MOVB    @#TSTN*1,R0   ;GET TEST #
2123      005132 006300              ASL      R0
2124      005134 006300              ASL      R0
2125      005136 023760 001016 002124  CMP      @#ATIME,STIMTBL(R0) ;CHECK THAT TIME IS WITHIN
2126      005144 101004              BHI      2$           ;LIMITS SPECIFIED
2127      005146 023760 001016 002126  CMP      @#ATIME,STIMTBL+2(R0)
2128      005154 101001              BHI      3$
2129      005156 104401      2$:      HLT+1              ;CALL ERROR ROUTINE
2130      005160
2131      005160 004737 003104      3$:      JSR      PC,.RESTORE  ;RESTORE REGISTERS FROM THE STACK
2132      005164 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      :      MOVB    #GAP,@#GAP          ;LOCATION GAP CONTAINS GAP #
2141      :      JSR      PC,GAPOK
2142
2143      GAPOK:
2144      005166 004737 003062      JSR      PC,SAVE      ;SAVE REGISTERS ON THE STACK
2145      005172 013700 001012      MOV      @#OSCTIM,R0   ;GET TIME PER TICK
2146      005176 010401              MOV      R4,R1        ;GET TICK COUNT
2147      005200 005002              CLR      R2           ;CLEAR SUMMING REGISTERS
2148      005202 005003              CLR      R3
2149      005204 060002      1$:      ADD      R0,R2        ;MULTIPLY TICK COUNT
2150      005206 005503              ADC      R3           ;BY TIME PER TICK
2151      005210 005301              DEC      R1
2152      005212 001374              BNE      1$
2153
2154      005214 010246              MOV      R2,-(SP)     ;DIVIDE TIME BY 10.
2155      005216 010346              MOV      R3,-(SP)
2156      005220 012746 000012      MOV      #10,-(SP)
2157      005224 004737 005376      JSR      PC,DIVIDE

```

```

2158 005230 005726          TST      (SP)+          ;DISCARD REMAINDER
2159 005232 012637 001016  MOV      (SP)+,@#ATIME  ;STORE QUOTIENT
2160 005236 113703 001124  MOVB    @#GAP,R3        ;GET GAP #
2161 005242 006303          ASL      R3              ;MULTPLY BY 4
2162 005244 006303          ASL      R3              ;TO GET AT TABLE ENTRY
2163 005246 023763 001016 002264  CMP     @#ATIME,GTIMTBL(R3) ;CHECK TIME (MAX)
2164 005254 101004          BHI     2$              ;
2165 005256 023763 001016 002266  CMP     @#ATIME,GTIMTBL+2(R3) ;CHECK TIME (MIN)
2166 005264 101001          BHI     3$              ;
2167 005266 104402          HLT+2    ;REPORT OUT OF RANGE ERROR
2168 005270 032777 002000 173502 3$: BIT   #SW10,@SWR      ;BRANCH IF TIMES NOT WANTED
2169 005276 001001          BNE     100$           ;
2170 005300 000240          NOP
2171
2172 005302          100$:
2173 005302 004737 003104  JSR     PC,.RESTORE    ;RESTORE REGISTERS FROM THE STACK
2174 005306 000207          RTS     PC              ;RETURN TO TEST
2175
2176          .SBTTL      DELAY SUBROUTINES
2177          ;THIS SUBROUTINE CAUSES A DELAY OF 115 MS.
2178 005310 004737 004730  DELAY: JSR     PC,TIMON
2179 005314 010246          MOV     R2,-(SP)      ;SAVE R2 ON THE STACK
2180 005316 012702 005326  MOV     #2$,R2        ;SET RETURN ADDRESS FOR TIMER
2181 005322          1$:
2182 005322 000163 005020  JMP     TIMER(R3)     ;GO TO TIMER & RETURN VIA R2
2183 005326 032704 004000  2$: BIT   #4000,R4
2184 005332 001773          BEQ     1$
2185 005334 012602          MOV     (SP)+,R2     ;RESTORE R2
2186 005336 000207          RTS     PC
2187
2188          ;THIS SUBROUTINE ALLOWS A CALLER SPECIFIED DELAY.
2189          ;CALL: MOV   DELAY TIME,DELTIM ;LOAD DELAY TIME (# OF TICKS)
2190          ; JSR   PC,DELAYV
2191 005340 005737 001120  DELAYV: TST   DELTIM ;BRANCH IF 0 DELAY
2192 005344 001413          BEQ     3$
2193 005346 004737 004730  JSR     PC,TIMON     ;TURN TIMER ON
2194 005352 010246          MOV     R2,-(SP)     ;SAVE R2 ON THE STACK
2195 005354 012702 005364  MOV     #2$,R2        ;SET RETURN ADDRESS FROM TIMER
2196 005360          1$:
2197 005360 000163 005020  JMP     TIMER(R3)     ;GO TO TIMER & RETURN VIA R2
2198 005364 023704 001120  2$: CMP   @#DELTIM,R4
2199 005370 101373          BHI     1$
2200 005372 012602          MOV     (SP)+,R2     ;RESTORE R2
2201 005374 000207          3$: RTS     PC
2202
2203          .SBTTL      DIVIDE SUBROUTINE
2204          ;THIS SUBROUTINE DIVIDES A DOUBLE PRECISION # AND RETURNS THE RESULT
2205          ;TO THE CALLER ON THE STACK. BOTH DIVIDEND & DIVISOR MUST BE POSITIVE.
2206          ;CALL: MOV   LEAST SIGNIFICANT HALF DIVIDEND,-(SP)
2207          ; MOV   #MOST SIGNIFICANT HALF DIVIDEND,-(SP)
2208          ; MOV   #DIVISOR,-(SP)
2209          ; JSR   PC,DIVIDE
2210          ;RETURN
2211          ; (SP)=REMAINDER ON STACK
2212          ; 2(SP)=QUOTIENT
2213

```

```

2214 ;NOTE: THIS SUBROUTINE DESTROYS PREVIOUS CONTENTS OF R0,R1,R2 & R3.
2215
2216 005376 005046 DIVIDE: CLR -(SP) ;SAVE LOC FOR SIGNS
2217 005400 012746 000021 MOV #17,-(SP) ;SET ITERATION COUNT
2218 005404 016601 000012 MOV 12(SP),R1 ;GET LSH DIVIDEND
2219 005410 016600 000010 MOV 10(SP),R0 ;GET MSH DIVIDEND
2220 005414 016602 000006 MOV 6(SP),R2 ;GET DIVISOR
2221 005420 005402 NEG R2 ;NEGATE DIVISOR
2222 005422 000241 CLC ;CLEAR 'C' BIT IN PSW
2223 005424 000405 BR 2$
2224 005426 006100 1$: ROL R0 ;ROTATE MSH DIVIDEND
2225 005430 010003 MOV R0,R3 ;SAVE IN R3
2226 005432 060203 ADD R2,R3 ;SUBTRACT DIVISOR FROM MSH DIVIDEND
2227 005434 103001 BCC 2$ ;BRANCH IF DIVIDEND > DIVISOR
2228 005436 010300 MOV R3,R0 ;SAVE REMAINDER IN R0
2229 005440 006101 2$: ROL R1 ;ROTATE LSH DIVIDEND
2230 005442 005316 DEC (SP) ;DECREMENT ITERATION COUNT
2231 005444 001370 BNE 1$
2232 005446 005726 TST (SP)+ ;POP ITERATION COUNTER
2233 005450 005726 TST (SP)+ ;POP SIGN CORRECTION
2234 005452 010166 000006 MOV R1,6(SP) ;PUSH REMAINDER ON STACK
2235 005456 010066 000004 MOV R0,4(SP) ;PUSH QUOTIENT ONTO STACK
2236 005462 012616 MOV (SP)+,(SP)
2237 005464 000207 RTS PC
2238
2239 .SBTTL DRIVE SUBROUTINES
2240 ;SUBROUTINE TO CHECK IF DRIVE IS AVAILABLE
2241 ;CALL: MOV B DRIVE#,DRVNUM
2242 ; JSR PC,DRVAVA
2243 ;RETURN: 'C' BIT SET IF NOT AVAILABLE
2244 005466 113765 001004 000010 DRVAVA: MOV B @DRVNUM,CS2(R5) ;LOAD DRIVE #
2245 005474 032765 040000 000026 BIT #TAP,DT(R5) ;CHECK IF TAPE UNIT
2246 005502 001003 BNE 1$
2247 005504 004737 005544 JSR PC,RHINIT
2248 005510 000262 SEV ;SET 'V' TO IND NOT AVAIL
2249 005512 000207 1$: RTS PC ;RETURN
2250
2251 ;SUBROUTINE TO CHECK IF TE16/TU77 SLAVE IS AVAILABLE FOR TEST
2252 ;CALL: MOV B DRIVE #,@DRVNUM ;PASS DRIVE # VIA DRVNUM
2253 ; MOV B SLAVE #,@SLVNUM ;PASS SLAVE # VIA SLVNUM
2254 ; JSR PC,SLVAVA ;CALL SUBROUTINE
2255 005514 113765 001004 000010 SLVAVA: MOV B @DRVNUM,CS2(R5) ;LOAD DRIVE #
2256 005522 113765 001005 000032 MOV B @SLVNUM,TC(R5) ;AND SLAVE #
2257 005530 032765 002000 000026 BIT #SPR,DT(R5) ;BRANCH IF SLAVE PRESENT
2258 005536 001001 BNE 1$
2259 005540 000262 SEV ;SET 'V' TO INDICATE NO SLAVE
2260 005542 000207 1$: RTS PC
2261
2262 ;SUBROUTINE TO INITIALIZE RH CONTROLLER
2263 ;CALL: JSR PC,RHINIT
2264
2265 005544 012765 000040 000010 RHINIT: MOV #40,CS2(R5)
2266 005552 113765 001004 000010 MOV B @DRVNUM,CS2(R5)
2267 005560 005046 CLR -(SP)
2268 005562 113716 001005 MOV B @SLVNUM,(SP)
2269 005566 012665 000032 MOV (SP)+,TC(R5) ;LOAD SLAVE # INTO TC REG

```

```

2270 005572 052765 001700 000032      BIS      #NORM11,TC(R5)
2271 005600 000207      RTS      PC
2272
2273      ;SUBROUTINE TO WAIT FOR DRIVE READY (DRY)
2274 005602 005027      WAITRDY:CLR      (PC)+      ;CLEAR WAIT TIMER
2275 005604 000000      WAITTIM:.WORD   0
2276 005606 105765 000012      1$:      TSTB      DS(R5)      ;WAIT FOR READY TO SET
2277 005612 100406      BMI      2$
2278 005614 005237 005604      INC      WAITTIM      ;INCREMENT WAIT TIMER
2279 005620 001372      BNE      1$           ;BRANCH IF TIME HAS NOT EXPIRED
2280 005622 000004 015275      TYPE,E.TIMEXP      ;TYPE 'TIME EXPIRED WAITING FOR RDY'
2281 005626 000425      BR      99$
2282 005630 032765 002000 000012 2$:      BIT      #EOT,DS(R5)      ;CHECK FOR END OF TAPE
2283 005636 001415      BEQ      3$           ;BRANCH IF NO EOT
2284 005640 000004 014134      TYPE,M.NAM
2285 005644 000004 014653      TYPE,M.EOT      ;TYPE 'END OF TAPE'
2286 005650 004737 005706      JSR      PC,.REWIND      ;REWIND SLAVE
2287 005654 102412      BVS      99$           ;BRANCH IF ERROR ON REWIND
2288 005656 004737 005770      JSR      PC,WRITE      ;WRITE A RECORD
2289 005662 005215      INC      (R5)          ;SET 'GO' BIT
2290 005664 004737 005602      JSR      PC,WAITRDY      ;WAIT FOR READY
2291 005670 000404      BR      99$           ;TAKE ERROR EXIT
2292 005672 032765 040000 000012 3$:      BIT      #ERR,DS(R5)      ;CHECK ERROR EXIT
2293 005700 001401      BEQ      100$
2294 005702 000262      99$:      SEV
2295 005704 000207      100$:     RTS      PC
2296      ;SUBROUTINE TO REWIND A UNIT (DRIVE/SLAVE COMBINATION)
2297      ;CALL      MOVB      DRIVE #,@#DRVNUM
2298      ;          MOVB      SLAVE #,@#SLVNUM
2299      ;          JSR      PC,.REWIND
2300      ;SUBROUTINE RETURNS TO CALLER WITH SELECTED SLAVE AT 'BOT', & 'V' SET IF
2301      ;AN ERROR OCCURS.
2302
2303 005706 004737 005544      .REWIND:JSR      PC,RHINIT      ;INITIALIZE CONTROLLER
2304 005712 004337 006124      JSR      R3,TMCMD      ;GO TO TM COMMAND SUBROUTINE
2305 005716 000000      .WORD   0             ;BUS ADDRESS (NOT USED)
2306 005720 000000      .WORD   0             ;WORD COUNT (NOT USED)
2307 005722 000000      .WORD   0             ;FRAME COUNT (NOT USED)
2308 005724 000006      .WORD   RWD           ;REWIND COMMAND
2309 005726 005215      INC      (R5)          ;SET 'GO' BIT
2310 005730 032765 000002 000012 1$:      BIT      #BOT,DS(R5)      ;BRANCH IF 'BOT' SET
2311 005736 001005      BNE      2$
2312 005740 032765 040000 000012      BIT      #ERR,DS(R5)      ;CHECK ERROR BIT
2313 005746 001006      BNE      99$           ;BRANCH IF ERROR BIT SET
2314 005750 000767      BR      1$
2315
2316 005752 032765 020000 000012 2$:      BIT      #PIP,DS(R5)      ;WAIT FOR TAPE MOTION TO STOP
2317 005760 001374      BNE      2$
2318 005762 000401      BR      100$
2319 005764 000262      99$:      SEV
2320 005766 000207      100$:     RTS      PC
2321
2322      ;SUBROUTINE TO WRITE 256. WORD RECORD
2323      ;CALL:      JSR      PC,WRITE
2324
2325 005770 004337 006124      WRITE:      JSR      R3,TMCMD      ;GO TO TM COMMAND SUBROUTINE

```

```
2326 005774 016674          .WORD  WTBUF          :BUS ADDRESS
2327 005776 177600          .WORD  WRDCNT         :WORD COUNT
2328 006000 177400          .WORD  FRMCNT         :FRAME COUNT
2329 006002 000060          .WORD  WFW           :WRITE FORWARD COMMAND
2330 006004 000207          RTS      PC
2331
2332          ;SUBROUTINE TO READ A 256. WORD RECORD.
2333          ;CALL: JSR      PC,READ
2334
2335 006006 004337 006124      READ: JSR      R3,@TMCMD
2336 006012 016674          .WORD  RDBUF          :ADDRESS OF READ BUFFER
2337 006014 177600          .WORD  WRDCNT         :2'S COMPLEMENT OF WORD COUNT
2338 006016 177400          .WORD  FRMCNT         :2'S COMPLEMENT OF FRAME COUNT
2339 006020 000070          .WORD  RDFWD         :READ FORWARD COMMAND
2340 006022 000207          RTS      PC
2341
2342          ;SUBROUTINE TO INITIATE READ REVERSE COMMAND
2343          ;CALL: JSR      PC,REVRD
2344
2345 006024 004337 006124      REVRD: JSR      R3,TMCMD
2346 006030 017274          .WORD  RDBUF+256.    :ADDRESS OF READ REVERSE BUFFER
2347 006032 177600          .WORD  WRDCNT         :2'S COMPLEMENT OF WORD COUNT
2348 006034 177400          .WORD  FRMCNT         :2'S COMPLEMENT OF FRAME COUNT
2349 006036 000076          .WORD  RDREV         :READ REVERSE COMMAND
2350 006040 000207          RTS      PC
2351
2352          ;SUBROUTINE TO SPACE FORWARD 1 RECORD
2353 006042 012765 177777 000006      FWDSPC: MOV     #-1,FC(R5) :LOAD RECORD COUNT
2354 006050 012715 000031          MOV     #SPCFWD+1,(R5) :LOAD COMMAND
2355 006054 004737 005602          JSR     PC,WAITRDY    :WAIT FOR READY
2356 006060 000207          RTS      PC          :RETURN
2357
2358          ;SUBROUTINE TO WRITE A RECORD AND BACK SPACE OVER THE RECORD.
2359 006062 004737 005770      WRT.BK: JSR     PC,WRITE :WRITE THE RECORD
2360 006066 005215          INC     (R5)          :SET 'GO' BIT
2361 006070 004737 005602          JSR     PC,WAITRDY
2362 006074 102412          BVS    2$
2363 006076 012765 177777 000006      MOV     #-1,FC(R5)    :LOAD RECORD COUNT
2364 006104 012715 000033          MOV     #SPCREV+1,(R5) :LOAD COMMAND
2365 006110 004737 005602          JSR     PC,WAITRDY
2366 006114 102402          BVS    2$
2367 006116 004737 005310      1$: JSR     PC,DELAY    :WAIT FOR TAPE MOTION TO STOP
2368 006122 000207          2$: RTS      PC
2369
2370          ;SUBROUTINE TO LOAD A COMMAND
2371          ;CALL: JSR      R3,TMCMD
2372          :
2373          : .WORD  BUS ADDRESS
2374          : .WORD  WORD COUNT (2'S COMPLEMENT)
2375          : .WORD  FRAME COUNT (2'S COMPLEMENT)
2376          : .WORD  COMMAND
2377 006124 012365 000004      TMCMD: MOV     (R3)+,BA(R5) :LOAD BUS ADDRESS
2378 006130 012365 000002          MOV     (R3)+,WC(R5) :LOAD WORD COUNT
2379 006134 012365 000006          MOV     (R3)+,FC(R5) :LOAD FRAME COUNT
2380 006140 012315          MOV     (R3)+,(R5)   :LOAD COMMAND
2381 006142 000203          RTS      R3        :RETJRN
```

```
2382
2383 ;SUBROUTINE TO PRINT SERIAL NUMBER
2384 ;JSR PC,SNPT
2385
2386 006144 016503 000030 SNPT: MOV SN(R5),R3
2387 006150 012701 001152 MOV #ODIGITS,R1
2388 006154 000303 SWAB R3
2389 006156 006003 ROR R3
2390 006160 006003 ROR R3
2391 006162 006003 ROR R3
2392 006164 006003 ROR R3 ;GET FIRST DIGIT
2393 006166 042703 177760 BIC #177760,R3
2394 006172 052703 000260 BIS #260,R3
2395 006176 110321 MOVB R3,(R1)+ ;FILL FIRST DIGIT
2396 006200 016503 000030 MOV SN(R5),R3
2397 006204 000303 SWAB R3
2398 006206 042703 177760 BIC #177760,R3
2399 006212 052703 000260 BIS #260,R3
2400 006216 110321 MOVB R3,(R1)+ ;GET SECOND DIGIT
2401 006220 016503 000030 MOV SN(R5),R3
2402 006224 006003 ROR R3
2403 006226 006003 ROR R3
2404 006230 006003 ROR R3
2405 006232 006003 ROR R3
2406 006234 042703 177760 BIC #177760,R3
2407 006240 052703 000260 BIS #260,R3
2408 006244 110321 MOVB R3,(R1)+ ;GET THIRD DIGIT
2409 006246 016503 000030 MOV SN(R5),R3
2410 006252 042703 177760 BIC #177760,R3
2411 006256 052703 000260 BIS #260,R3
2412 006262 110321 MOVB R3,(R1)+ ;GET FOURTH DIGIT
2413 006264 105011 CLRB (R1)
2414 006266 000004 001152 TYPE,ODIGITS ;TYPE SERIAL NUMBER
2415 006272 000207 RTS PC ;RETURN
2416
```

```
2417 .SBTTL PROGRAM INITIALIZATION
2418 006274 012706 000600 INIT: MOV #STKPTR,SP ;SET STACK PTR
2419 006300 005037 001272 CLR @WINBUF
2420
2421 006304 013746 000006 MOV @#6,-(SP) ;SAVE VECTORS
2422 006310 013746 000004 MOV @#4,-(SP)
2423 006314 012737 006334 000004 MOV #61$,@#4 ;SET UP FOR TIMEOUT
2424 006322 022777 177777 172450 CMP #-1,@SWR ;REFERENCE HARDWARE SWITCH REGISTER
2425 006330 001402 BEQ 60$
2426 006332 000404 BR 62$
2427 006334 022626 61$: CMP (SP)+,(SP)+ ;ADJUST STACK
2428 006336 012737 000176 001000 60$: MOV #SWREG,SWR ;POINT TO SOFTWARE SWITCH REG
2429 006344 012637 000004 62$: MOV (SP)+,@#4 ;RESTORE VECTORS
2430 006350 012637 000006 MOV (SP)+,@#6
2431 006354 105037 001131 CLRB @#PRGFLG ;CLEAR PROGRAM FLAG
2432 006360 105037 001135 CLRB @#ASFLG ;CLEAR ASK FLAG
2433 006364 105037 001134 CLRB @#PSCNT ;SET PASS COUNT = 0
2434 006370 005027 CLR (PC)+ ;:CLEAR CHAIN INDICATOR
2435 006372 000000 CHNFLG: .WORD 0 ;:CHAIN MODE INDICATOR
2436 ;:1/0 = CHAIN/NOT CHAIN MODE
2437 006374 005737 000042 TST @#42 ;:BRANCH IF IN DUMP MODE
2438 006400 001407 BEQ 50$
2439 006402 012737 000176 001000 MOV #SWREG,SWR ;:INVOKE SOFTWARE SWR
2440 006410 005237 006372 INC CHNFLG ;:SET CHNFLG = CHAIN MODE
2441 006414 000137 006420 JMP 1$ ;:GO TO CHAIN ADDRESS
2442 006420 50$:
2443 006420 122737 000006 000041 1$: CMPB #6,@#41 ;BRANCH IF NOT LOADED VIA TMDP
2444 006426 001003 BNE 2$
2445 006430 000004 014344 TYPE,I.REM ;ADVISE USER TO REMOVE TMDP
2446 006434 000000 HALT
2447 006436 000004 014134 2$: TYPE,M.NAM ;TYPE TITLE
2448 006442 005737 006372 TST CHNFLG ;SEE IF CHAIN MODE
2449 006446 001025 BNE 5$ ;IF SO: BR
2450 006450 105037 014134 CLRB M.NAM ;DO NOT TYPE TITLE ON RESTART
2451 006454 000004 014417 TYPE,I.REG ;ASK USER TO TYPE CONT BASE ADRS
2452 006460 013702 001010 MOV @#TMBASE,R2 ;GET CURRENT CONT BASE ADDRESS
2453 006464 004737 003134 JSR PC,TYPECT ;AND TYPE IT
2454 006470 000004 001416 TYPE,SPACE
2455 006474 004737 004000 JSR PC,.INPUT ;GET USER INPUT
2456 006500 122737 000015 001272 CMPB #CR,@#INBUF ;DO NOT CHANGE CURRENT VALUE
2457 006506 001405 BEQ 5$ ;IF USER TYPES <CR>
2458 006510 004737 003564 4$: JSR PC,CNVTA0 ;CONVERT ASCII TO OCTAL
2459 006514 013737 001122 001010 MOV @#OCTALO,@#TMBASE ;SET NEW ADDRESS
2460 006522 013705 001010 5$: MOV @#TMBASE,R5
2461 ;ROUTINE TO CHECK IF CONTROLLER (RH11) IS AVAILAABIE
2462 SEC ;SET 'C' IN PSW
2463 006526 000261 TST (R5) ;BRANCH IF CONTROLLER AVAIL
2464 006530 005715 BCC 6$
2465 006532 103003 TYPE,E.NCON
2466 006534 000004 014712 BR INIT
2467 006540 000655 6$: MOV #ERRTRP,@#ERRVEC ;SET ERROR TRAP VECTOR
2468 006542 012737 004250 000004
```

```
2469 ;ROUTINE TO GET TM03 DRIVES USER DESIRES TO TEST
2470 006550 105037 001127 DRIVES: CLRB @#ERFLG ;CLEAR ERROR FLAG
2471 006554 012701 001162 MOV #DRVTBL,R1 ;MARK ALL DRIVES AS NOT TO
2472 006560 012700 000004 MOV #4,R0 ;BE TESTED. A '0' INDICATES
2473 006564 005021 1$: CLR (R1)+ ;THAT A DRIVE IS NOT TO BE
2474 006566 005300 DEC R0 ;TESTED
2475 006570 001375 BNE 1$
2476 006572 005737 006372 TST CHNFLG ;BRANCH IF IN CHAIN MODE
2477 006576 001014 BNE 2$
2478 006600 000004 014464 TYPE,I.DRVS
2479 006604 004737 004000 JSR PC,,INPUT ;GET USER INPUT
2480 006610 012700 001272 MOV #INBUF,R0
2481 006614 122710 000101 CMPB #'A,(R0) ;IF USER RESPONDS WITH 'A' OR
2482 006620 001403 BEQ 2$ ;<CR> THEN ALL AVAILABLE DRIVES
2483 006622 122710 000015 CMPB #CR,(R0) ;ARE TO BE TESTED
2484 006626 001013 BNE 4$
2485 006630 110637 001131 2$: MOVB SP,PRGFLG ;SET FLAG TO IND ALL DRIVES
2486 006634 012701 001162 MOV #DRVTBL,R1 ;MARK ALL DRIVES TO BE TESTED
2487 006640 012700 000004 MOV #4,R0 ;A '-1' INDICATES THAT A DRIVE
2488 006644 012721 177777 3$: MOV #-1,(R1)+ ;IS TO BE TESTED
2489 006650 005300 DEC R0
2490 006652 001374 BNE 3$
2491 006654 000417 BR CHKDRV ;GO CHECK DRIVE AVAILABILITY
2492
2493 ;GET USER SELECTED DRIVES AND MARK EACH DRIVE SELECTED TO BE TESTED
2494 006656 122710 000015 4$: CMPB #CR,(R0)
2495 006662 001414 BEQ CHKDRV
2496 006664 121027 000054 CMPB (R0),'#', ;CHECK IF 'COMMA'
2497 006670 001001 BNE 5$
2498 006672 105720 TSTB (R0)+ ;STEP PTR PAST 'COMMA'
2499 006674 112001 5$: MOVB (R0)+,R1
2500 006676 042701 177770 BIC #177770,R1
2501 006702 112761 177777 001162 MOVB #-1,DRVTBL(R1)
2502 006710 000240 NOP
2503 006712 000761 BR 4$
2504
2505 ;ASCERTAIN THAT DRIVES (TM03'S) SPECIFIED ARE AVAILABLE
2506 006714 005000 CHKDRV: CLR R0 ;A (0) IN DRVTBL(R0) INDICATES
2507 006716 105760 001162 1$: TSTB DRVTBL(R0) ;THE DRIVE IS NOT TO BE TESTED
2508 006722 001005 BNE 3$ ;A '1' INDICATES TO BE TESTED
2509 006724 005200 2$: INC R0
2510 006726 122700 000010 CMPB #8.,R0
2511 006732 001371 BNE 1$
2512 006734 000424 BR 5$
2513 006736 110037 001004 3$: MOVB R0,@#DRVNUM ;GET DRIVE #
2514 006742 004737 005466 JSR PC,@#DRVAVA ;AND CHECK IF AVAILABLE
2515 006746 102366 BVC 2$ ;'V' BIT SET INDICATES NOT AVAIL
2516 006750 105737 001131 TSTB @#PRGFLG ;DO NOT TYPE NOT AVAILABLE
2517 006754 001011 BNE 4$ ;MESSAGE IF ALL SELECTED
2518 006756 000004 014757 TYPE,E.NDRV
2519 006762 116037 001140 015011 MOVB DIGTAB(R0),@#E.NAVA ;SET DRIVE # IN MESSAGE
2520 006770 000004 015011 TYPE,E.NAVA
2521 006774 110637 001127 MOVB SP,@#ERFLG ;SET 'ERROR' FLAG
2522 007000 105060 001162 4$: CLRB DRVTBL(R0) ;MARK DRIVE UNAVAILABLE
2523 007004 000747 BR 2$ ;CHECK NEXT DRIVE
2524 007006 105737 001127 5$: TSTB @#ERFLG ;GO GET SLAVES IF NO ERROR
```

```
2525 007012 001256          BNE      DRIVES          ;ELSE ASK USER TO RETYPE DRIVES
2526
2527          ;ROUTINE TO GET SLAVES (TE16/TU77'S) USER DESIRES TO TEST
2528 007014 105037 001127      SLAVES: CLR      @MERFLG          ;CLEAR ERROR INDICATOR
2529 007020 012701 001172      MOV      #SLVTBL,R1          ;MARK ALL SLAVES (64.) AS NOT
2530 007024 012700 000040      MOV      #32.,R0            ;TO BE TESTED.A 0 INDICATES THAT
2531 007030 005021          1$:   CLR      (R1)+            ;A DRIVE'S SLAVE IS NOT TO BE
2532 007032 005300          DEC      R0                  ;TESTED
2533 007034 001375          BNE      1$
2534 007036 012701 001172      MOV      #SLVTBL,R1          ;R1 POINTS TO DRIVE'S SLAVE
2535 007042 105760 001162      2$:   TSTB     DRVTBL(R0)      ;BRANCH IF DRIVE IS TO BE TESTED
2536 007046 001007          BNE      4$                  ;& IS AVAILABLE
2537 007050 062701 000010      3$:   ADD      #8.,R1          ;STEP SLAVE PTR TO NEXT DRIVE'S
2538 007054 005200          INC      R0                  ;SLAVES AND INCREMENT DRIVE #
2539 007056 122700 000010      CMPB     #8.,R0            ;CHECK ALL DRIVES
2540 007062 001367          BNE      2$                  ;AND WHEN ALL DRIVES CHECKED
2541 007064 000457          BR       CHKSLV             ;GO CHECK SLAVE AVAILABILITY
2542
2543 007066 105737 001131      4$:   TSTB     @MPRGFLG        ;BRANCH IF USER SELECTED ALL
2544 007072 001021          BNE      5$                  ;DRIVES
2545 007074 110037 001004      MOVB     R0,DRVNUM          ;GET DRIVE #
2546 007100 116037 001140      MOVB     DIGTAB(R0),@#I.DRV ;PREPARE USER ACTION MESSAGE
2547 007106 000004 014534          TYPE,I.SLV
2548 007112 004737 004000      JSR      PC,,INPUT          ;GET USER INPUT
2549 007116 012703 001272      MOV      #INBUF,R3          ;SET PTR TO USER INPUT
2550 007122 122713 000101      CMPB     #'A',(R3)          ;AN 'A' OR <CR> AS FIRST CHAR
2551 007126 001403          BEQ      5$                  ;INDICATES TEST ALL SLAVES
2552 007130 122713 000015      CMPB     #CR,(R3)
2553 007134 001015          BNE      7$
2554 007136 110637 001131      5$:   MOVB     SP,@MPRGFLG        ;SET 'ALL' INDICATOR
2555 007142 012701 001172      MOV      #SLVTBL,R1          ;MARK ALL SLAVES FOR ALL
2556 007146 012700 000040      MOV      #32.,R0            ;DRIVES AS TO BE TESTED
2557 007152 012721 177777      6$:   MOV      #-1,(R1)+
2558 007156 005300          DEC      R0
2559 007160 001374          BNE      6$
2560 007162 105737 001131      TSTB     @MPRGFLG          ;BRANCH IF ALL WAS SELECTED
2561 007166 001016          BNE      CHKSLV
2562
2563 007170 122713 000015      7$:   CMPB     #CR,(R3)          ;GET USER SELECTED SLAVES FOR
2564 007174 001725          BEQ      3$                  ;DRIVE
2565 007176 121327 000054      CMPB     (R3),#',          ;STEP PTR PAST 'COMMA'
2566 007202 001001          BNE      8$
2567 007204 105723          TSTB     (R3)+
2568 007206 112304          8$:   MOVB     (R3)+,R4          ;AND MARK SELECED SLAVE
2569 007210 042704 177770      BIC      #177770,R4          ;AS TO BE TESTED
2570 007214 060104          ADD      R1,R4
2571 007216 112714 177777      MOVB     #-1,(R4)
2572 007222 000762          BR       7$
2573
2574          ;ASCERTAIN THAT SLAVES (TE16/TU77'S) SELECTED ARE AVAILABLE
2575 007224 005000      CHKSLV: CLR      R0          ;R0 WILL CONTAIN THE DRIVE #
2576 007226 005001          CLR      R1                  ;AND R1 THE SLAVE #
2577 007230 012702 001172      MOV      #SLVTBL,R2          ;SET PTR TO SLAVE TABLE
2578 007234 105760 001162      1$:   TSTB     DRVTBL(R0)      ;BRANCH IF DRIVE SELECTED
2579 007240 001020          BNE      3$                  ;& AVAILABLE FOR TEST
2580 007242 005200          2$:   INC      R0              ;INCREMENT DRIVE #
```

```
2581 007244 105760 001161      TSTB    <DRVTBL-1>(R0)      ;++C WAS PREVIOUS DRIVE SELECTED
2582 007250 001003              BNE     9$                   ;++C BRANCH IF AVAIL.
2583 007252 062702 000010      ADD     #8.,R2               ;++C ADJUST SLAVE POINTER
2584 007256 000405              BR      10$
2585 007260 105737 001131      9$:    TSTB    @#PRGFLG        ;++C WAS ALL SELECTED
2586 007264 001002              BNE     10$
2587 007266 062702 000010      ADD     #8.,R2               ;++C ADJUST SLAVE POINTER
2588 007272 022700 000010      10$:   CMP     #8.,R0           ;SLAVES. BRANCH TO 1$ IF NOT ALL
2589 007276 001356              BNE     1$                   ;DRIVES CHECKED OTHERWISE EXIT
2590 007300 000437              BR      8$
2591
2592 007302 005001      3$:    CLR     R1               ;SET SLAVE # 0
2593 007304 105712      4$:    TSTB    (R2)             ;BRANCH IF DRIVE'S SLAVE IS SEL-
2594 007306 001006              BNE     6$                   ;ECTED FOR TEST
2595 007310 005201      5$:    INC     R1               ;INCREMENT SLAVE #
2596 007312 005202              INC     R2                   ;STEP PTR TO NEXT SLAVE
2597 007314 022701 000010      CMP     #8.,R1             ;GO TO 4$ IF ALL SLAVES NOT
2598 007320 001371              BNE     4$                   ;CHECKED
2599 007322 000747              BR      2$                   ;OTHERWISE GO TO 2$ ABOVE
2600
2601 007324 110037 001004      6$:    MOVB   R0,@#DRVNUM      ;PASS DRIVE & SLAVE #
2602 007330 110137 001005      MOVB   R1,@#SLVNUM
2603 007334 004737 005514      JSR    PC,@#SLVAVA
2604 007340 102363              BVC     5$                   ;AND CHECK IF AVAILABLE
2605 007342 105737 001131      TSTB    @#PRGFLG          ;'V' SET INDICATES ERROR
2606 007346 001012              BNE     7$                   ;DO NOT TYPE ERROR MSG IF ALL
2607 007350 116037 001140 015001  MOVB   DIGTAB(R0),@#E.DRV   ;SLAVES SELECTED
2608 007356 116137 001140 015011  MOVB   DIGTAB(R1),@#E.NAVA  ;ICATES ERROR. PREPARE ERROR
2609 007364 000004 014773      TYPE,E.NSLV               ;MESSAGE
2610 007370 110637 001127      MOVB   SP,@#ERRFLG
2611 007374 105012      7$:    CLRB   (R2)             ;SET ERROR INDICATOR
2612 007376 000744              BR      5$                   ;CLEAR SLAVE TABLE ENTRY
2613
2614 007400 105737 001127      8$:    TSTB    @#ERRFLG        ;GET NEXT SLAVE
2615 007404 001203              BNE     SLAVES
2616 007406 012737 004250 000004  100$:  MOV     #ERRTRP,@#ERRVEC   ;BRANCH IF ERROR
2617
2618
2619
2620
2621 007414 012706 000600      ;SCAN DRIVE AND SLAVE TABLE FOR DRIVE/SLAVE COMBINATION TO TEST.
2622 007420 105037 001004      ;RESTART ADDRESS--PROGRAM STARTS HERE WHEN START ADDRESS = 210 AND
2623 007424 105037 001005      ;AFTER ALL SELECTED DRIVE/SLAVE COMBINATIONS HAVE BEEN TESTED.
2624 007430 012737 001172 001006  RSTRT: MOV     #600,SP       ;SET STACK PTR
2625 007436 105037 001132      CLRB   @#DRVNUM           ;SET DRIVE AND SLAVE # 0
2626
2627
2628 007442 113700 001004      CLRB   @#SLVNUM
2629 007446 113701 001005      CLRB   @#SLVNUM
2630 007452 013702 001006      MOV     #SLVTBL,@#SLVPTR   ;SET PTR TO SLAVE TABLE
2631 007456 122737 000006 000041  MOV     #6,@#41           ;CLEAR 'UNIT FOUND' IND.
2632 007464 001001              CLRB   @#UNTFND
2633 007466 105012              ;PROGRAM RESTARTS HERE AFTER A DRIVE/SLAVE HAS BEEN TESTED.
2634
2635 007470 105760 001162      BEGIN: MOVB   @#DRVNUM,R0   ;GET DRIVE #
2636 007474 001011              MOVB   @#SLVNUM,R1        ;AND SLAVE #
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
```

```
2637 007476 005001          CLR      R1          ;CLEAR SLAVE #
2638 007500 062702 000010    ADD      #8.,R2      ;AND STEP PTR TO NEXT DRIVE'S
2639 007504 005200          INC      R0          ;SLAVES AND INCREMENT DRIVE #
2640 007506 022700 000010    CMP      #8.,R0      ;EXIT TEST IF ALL DRIVES
2641 007512 001366          BNE     1$          ;CHECKED OTHERWISE CONTINUE
2642 007514 000137 013432    JMP      @END        ;SCAN FOR NEXT 'UNIT'
2643
2644 007520 105712          3$:     TSTB     (R2)    ;BRANCH IF SLAVE ON DRIVE IS
2645 007522 001007          BNE     4$          ;AVAILABLE THERWISE STEP
2646 007524 005202          INC      R2          ;PTR TO NEXT SLAVE
2647 007526 005201          INC      R1          ;INCREMENT SLAVE #
2648 007530 122701 000010    CMPB    #8.,R1      ;UNTIL ALL SLAVES CHECKED
2649 007534 001371          BNE     3$          ;WHEN ALL SLAVES CHECKED
2650 007536 005001          CLR      R1          ;SET SLAVE # 0
2651 007540 000761          BR      2$          ;AND CONTINUE SCAN
2652
2653 007542 110637 001132    4$:     MOV      SP,@UNTFND ;INDICATE THAT A 'UNIT' IS FOUND
2654 007546 110037 001004    MOV      R0,@DRVNUM  ;SET DRIVE #
2655 007552 110137 001005    MOV      R1,@SLVNUM  ;SET SLAVE #
2656 007556 010237 001006    MOV      R2,@SLVPTR  ;SAVE SLAVE PTR
2657
2658 007562 105737 001135    5$:     TSTB     @ASFLG    ;
2659 007566 001034          BNE     7$          ;
2660 007570 112737 000001 001135    MOV      #1,ASFLG    ;
2661 007576 005737 006372    TST     CHNFLG      ;BRANCH IF IN CHAIN MODE
2662 007602 001026          BNE     7$          ;
2663 007604 105037 001130    CLRB    @SKEWFLG    ;++B CLEAR SKEW (SPEED) TESTS SELECTED FLAG
2664 007610 000004 014620    TYPE,I.SPD         ;ASK USER IF HE WANTS TO RUN SPEED TESTS
2665 007614 004737 004000    JSR     PC,,INPUT   ;GET USER INPUT
2666 007620 012703 001272    MOV     #INBUF,R3   ;GET REPLY
2667 007624 122713 000015    CMPB    #CR,(R3)    ;DO NOT DO SKEW TESTS IF <CR> IS FIRST
2668 007630 001405          BEQ     6$          ;
2669 007632 132713 000001    BITB    #1,(R3)     ;BRANCH IF 'N'
2670 007636 001402          BEQ     6$          ;
2671 007640 111337 001130    MOV      (R3),@SKEWFLG ;SET INDICATOR
2672 007644 022737 000176 001000 6$:     CMP      #SWREG,SWR  ;BRANCH IF SOFTWARE SWR
2673 007652 001002          BNE     7$          ;NOT INVOKED
2674 007654 004737 002526    JSR     PC,GTSWR    ;GET SWITCH REGISTER
2675
2676          7$:
2677          ;ROUTINE TO ASCERTAIN SLAVE TYPE AND LOAD APPROPRIATE SPECIFICATION
2678          ;TABLES (TE16 OR TU77) INTO STIMTBL AND GTIMTBL.
2678 007660 013705 001010    MOV      @TMBASE,R5  ;GET BASE ADDRESS OF REGISTERS
2679 007664 004737 005544    JSR     PC,RHINIT   ;INIT DRIVE/SLAVE
2680 007670 012704 000240    MOV      #ENDTBL-STIBL,R4 ;GET TABLE LENGTH
2681 007674 012703 002124    MOV      #STIMTBL,R3 ;AND STARTING ADDRESS OF TABLE
2682 007700 012702 001424    MOV      #TE16TTBL,R2 ;GET ADDRESS OF TE16 TIME TABLE
2683 007704 105037 001136    CLRB    @TE16       ;SET FLAG = TE16
2684 007710 032765 000004 000026    BIT     #4,DT(R5)   ;BRANCH IF TU77
2685 007716 001012          BNE     9$          ;
2686 007720 012737 000070 001012    MOV      #56.,OSCTIM ;SET US/TICK = 56
2687 007726 012737 000022 001014    MOV      #18.,GAPDEL ;SET 18 TICKS/MS
2688 007734 112223          8$:     MOV      (R2)+,(R3)+ ;MOVE TE16 TIME AND GAP TABLES
2689 007736 005304          DEC     R4          ;INTO STIMTBL & GTIMTBL
2690 007740 001375          BNE     8$          ;
2691 007742 000416          BR      11$         ;EXIT ROUTINE
2692
```

CZTEECO TMO3-TE16/TU77 DFT
CZTEEC.P11 22-AUG-79 08:40

MACY11 30A(1052) 22-AUG-79 09:00 PAGE 60
PROGRAM INITIALIZATION

SEG 0060

```

2693 007744 012702 001664          9$:  MOV    #TU77TTBL,R2    ;GET ADDRESS OF TU77 TIME TABLE
2694 007750 112737 000001 001136  MOVB   #1,@#TE16      ;SET FLAG = TU77
2695 007756 012737 000120 001012  MOV    #80.,OSCTIM    ;SET US/TICK = 80
2696 007764 012737 000003 001014  MOV    #3,GAPDEL      ;SET 3 TICKS PER .25MS
2697 007772 112223          10$:  MOVB   (R2)+,(R3)+    ;MOVE TU77 TIME AND GAP TABLES
2698 007774 005304          DEC    R4              ;INTO STIMTBL & GTIMTBL
2699 007776 001375          BNE    10$
2700 010000          11$:
2701
2702          ;NOTE THIS IS NOT A TEST
2703          ;INITIALIZE PROGRAM FLAGS
2704 010000 105037 001126          TST00G: CLRB   @#TSTNUM    ;SET TEST # 0
2705 010004 013705 001010          MOV    @#TMBASE,R5    ;SET ADDRESS OF FIRST TMO3 REG
2706 010010 010500          MOV    R5,R0          ;R0 CONTAINS ADDRESS OF FC REG
2707 010012 062700 000006          ADD    #FC,R0
2708 010016 010501          MOV    R5,R1          ;R1 CONTAINS ADDRESS OF DS REG
2709 010020 062701 000012          ADD    #DS,R1
2710 010024 012703 005020          MOV    #TIMER,R3     ;SET JUMP ADDRESS TO TIMER
2711 010030 105037 001125          CLRB   @#ITCNT       ;CLEAR SUBTEST ITERATION COUNT
2712 010034 052737 000100 177560  BIS    #100,@#TKS     ;SET KEYBOARD IE BIT
2713
2714          ;GET USER RUN PROCEDURE
2715          ;IF SWR<05::00> IS NOT 0 THEN RUN TEST IN SWR<05::00>
2716          ;OTHERWISE RUN ALL TESTS
2717
2718 010042 004737 005706          JSR    PC,.REWIND     ;REWIND SLAVE
2719 010046 102504          BVS    99$           ;BRANCH IF ERROR ON REWIND
2720 010050 105737 001130          TSTB   @#SKEWFLG     ;++B BRANCH IF SWEW (SPEED) TEST SELECTED
2721 010054 C01006          BNE    10$
2722 010056 004737 005770          JSR    PC,WRITE      ;WRITE A RECORD
2723 010062 005215          INC    (R5)          ;SET 'GO' BIT
2724 010064 004737 005602          JSR    PC,WAITRDY    ;WAIT FOR READY
2725 010070 102473          BVS    99$
2726 010072 117702 170702          10$:  MOVB   @SWR,R2        ;GET SWITCHES
2727 010076 042702 177740          BIC    #177740,R2    ;CLEAR ALL BUT TEST #
2728 010102 001421          BEC    2$           ;& BRANCH IF TEST 0 WAS SELECTED
2729 010104 000004 015137          TYPE,E.HDR          ;TYPE TEST #
2730 010110 004737 003134          JSR    PC,TYPOCT
2731 010114 006302          ASL    R2            ;FORM INDEX VALUE
2732 010116 016237 002364 010126  MOV    NAMPTR(R2),1$ ;GET ADDRESS OF TEST'S NAME
2733 010124 000004          TYPE                                ;AND TYPE IT
2734 010126 000000          1$:  .WORD  0
2735 010130 000004 001402          TYPE,CRLF
2736 010134 016237 002444 001002  MOV    TSTTBL(R2),@#SCPADR ;SET SCOPE ADDRESS FOR TEST
2737 010142 000172 002444          JMP    @TSTTBL(R2)    ;GO TO TEST
2738 010146 032777 002000 170624  2$:  BIT    #SW10,@SWR    ;BRANCH IF TIMES NOT TO BE TYPED
2739 010154 001034          BNE    5$
2740 010156 000004 015346          TYPE,L.HDR1
2741 010162 113702 001004          MOVB   DRVNUM,R2     ;GET DRIVE #
2742 010166 113704 001005          MOVB   SLVNUM,R4     ;AND SLAVE #
2743 010172 116237 001140 015530  MOVB   DIGTAB(R2),@#L.DRV ;SET DRIVE AND SLAVE #'S
2744 010200 116437 001140 015542  MOVB   DIGTAB(R4),@#L.SLV ;INTO L.HDR2 MESSAGE
2745 010206 000004 015463          TYPE,L.HDR2
2746 010212 105737 001136          TSTB   @#TE16        ;BRANCH IF NOT TE16
2747 010216 001003          BNE    3$
2748 010220 000004 015546          TYPE,L.TE16         ;TYPE 'TE16'

```

```
2749 010224 000402          BR      4$
2750 010226 000004 015554   3$:    TYPE,L.TU77          ;TYPE 'TU77'
2751 010232 000004 015562   4$:    TYPE,L.SER          ;TYPE 'SERIAL #'
2752 010236 004737 006144   JSR    PC,SNPT          ;PRINT SLAVE SERIAL #
2753 010242 000004 015574   TYPE,L.HDR3
2754 010246 105737 001130   5$:    TSTB @#SKEWFLG      ;++B BRANCH IF SPEED TESTS NOT
2755 010252 001405          BEQ    100$             ;SELECTED
2756 010254 000137 013550   JMP    @#SKEWTST       ;GO DO SPEED TESTS
2757 010260 104400          99$:   HLT
2758 010262 000137 010000   JMP    TST000          ;LOOP TEST AS LONG AS ERROR PERSISTS
2759 010266 012737 010274 001002 100$:  MOV    #TST001,@#SCPADR ;SE' SCOPE LOOP ADDRESS
2760
```

```

2761          .SBTTL START OF TESTS
2762          ;TEST 001 - WRITE FROM BOT
2763          ;THIS TEST WILL MEASURE ACCELERATION DELAY REQUIRED TO
2764          ;MOVE THE TAPE APPROXIMATELY SEVEN (7) INCHES FORWARD
2765          ;FROM DEAD STOP BEFORE STARTING TO TRANSFER DATA.
2766
2767          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2768 010274 112737 000001 001126 TST001: MOVB #1,@TSTNUM ;SET TEST #
2769 010302 012702 010326          MOV #1$,R2 ;SET RETURN PC FROM TIMER
2770 010306 004737 005706          JSR PC,.REWIND ;REWIND SLAVE
2771 010312 102420          BVS 99$ ;BRANCH IF ERROR ON REWIND
2772 010314 004737 005770          JSR PC,WRITE ;GO SETUP WRITE COMMAND
2773 010320 004737 004730          JSR PC,TIMON ;TURN TIMER ON
2774 010324 005215          INC (R5) ;SET 'GO' BIT
2775
2776 010326 005765 000032 1$: TST TC(R5) ;BRANCH WHEN 'ACCL'=0
2777 010332 100002          BPL 2$
2778 010334 000163 005020          JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2779
2780 010340 004737 005602 2$: JSR PC,WAITRDY ;WAIT FOR COMMAND TO FINISH
2781 010344 102403          BVS 99$ ;BRANCH IF ERROR
2782 010346 004737 005056          JSR PC,TIMOK ;GO CHECK TIME
2783 010352 000401          BR 100$
2784 010354 104400 99$: HLT
2785 010356 104000 100$: SCOPE
2786
2787          ;TEST 002 - WRITE START
2788          ;THIS TST MEASURES TIME FROM 'GO'=1 TO 'ACCL'-0.
2789 010360 112737 000002 001126 TST002: MOVB #2,@TSTNUM ;SET TEST # 2
2790 010366 004737 005770          JSR PC,WRITE ;INITIATE WRITE COMMAND
2791 010372 012702 010404          MOV #1$,R2 ;SET RETURN PC FROM TIMER
2792 010376 004737 004730          JSR PC,TIMON
2793 010402 005215          INC (R5) ;SET 'GO' BIT
2794
2795 010404 005765 000032 1$: TST TC(R5) ;BRANCH WHEN 'ACCL'=0
2796 010410 100002          BPL 2$
2797 010412 000163 005020          JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2798
2799 010416 004737 005602 2$: JSR PC,WAITRDY ;WAIT FOR READY
2800 010422 102403          BVS 99$ ;BRANCH IF ERROR
2801 010424 004737 005056          JSR PC,TIMOK ;GO CHECK TIME RECORDED
2802 010430 000401          BR 100$ ;EXIT VIA SCOPE
2803
2804 010432 104400 99$: HLT ;REPORT ERROR
2805 010434 104000 100$: SCOPE
2806
2807          ;TEST 003- WRITE SHUTDOWN
2808          ;THIS TEST MEASURES TIME FROM 'FC REG'-0 TO 'SWDN'=1.
2809 010436 112737 000003 001126 TST003: MOVB #3,@TSTNUM ;SET TEST#3
2810 010444 004737 005770          JSR PC,WRITE ;INITIATE WRITE COMMAND
2811 010450 005215          INC (R5) ;SET 'GO' BIT
2812
2813 010452 005710 1$: TST (R0) ;BRANCH WHEN WRITING FINISHED
2814 010454 001404          BEQ 2$
2815 010456 032711 040000          BIT #ERR,(R1) ;MONITOR ERROR BIT
2816 010462 001017          BNE 99$

```

```
2817 010464 000772          BR      1$
2818
2819 010466          2$:
2820 010466 004737 004730      JSR    PC,TIMON      ;TURN TIMER ON
2821 010472 010702          MOV    PC,R2        ;LOAD RETURN PC FROM TIMER
2822 010474 032711 000020      3$:    BIT    #SDWN,(R1) ;BRANCH WHEN DS <SDWN> SETS
2823 010500 001002          BNE    4$
2824 010502 000163 005020      JMP    TIMER(R3)    ;GO TO TIMER & RETURN VIA R2
2825
2826 010506 004737 005602      4$:    JSR    PC,WAITRDY  ;WAIT FOR READY
2827 010512 102403          BVS    99$
2828 010514 004737 005056      JSR    PC,TIMOK     ;GO CHECK TIME RECORDED
2829 010520 000401          BR     100$
2830 010522 104400      99$:   HLT
2831 010524 104000      100$: SCOPE        ;REPORT ERROR
2832
2833          ;TEST 004 - WRITE SETTLEDOWN
2834          ;THIS TEST MEASURES TIME FROM 'SWDN'=1 TO 'SWDN'=0.
2835 010526 112737 000004 001126 TST004: MOVB    #4,@TSTNUM
2836 010534 004737 005770      JSR    PC,WRITE
2837 010540 005215          INC    (R5)        ;SET 'GO' BIT
2838
2839 010542 005710      1$:    TST    (R0)        ;BRANCH WHEN WRITING FINISHED
2840 010544 001404          BEQ    2$
2841 010546 032711 040000      BIT    #ERR,(R1)   ;CHECK ERROR BIT
2842 010552 001026          BNE    99$
2843 010554 000772          BR     1$
2844
2845 010556 032711 000020      2$:    BIT    #SDWN,(R1) ;WAIT FOR ASSERTION OF 'SDWN'
2846 010562 001004          BNE    3$
2847 010564 032711 040000      BIT    #ERR,(R1)   ;MONITOR ERROR BIT
2848 010570 001017          BNE    99$
2849 010572 000771          BR     2$
2850
2851 010574          3$:
2852 010574 004737 004730      JSR    PC,TIMON      ;TURN TIMER ON
2853 010600 010702          MOV    PC,R2        ;SET RETURN PC FROM TIMER
2854 010602 032711 000020      BIT    #SDWN,(R1)  ;BRANCH WHEN SWDN CLEARS
2855 010606 001402          BEQ    5$
2856 010610 000163 005020      JMP    TIMER(R3)    ;GO TO TIMER & RETURN VIA R2
2857
2858 010614 004737 005602      5$:    JSR    PC,WAITRDY  ;WAIT FOR READY
2859 010620 102403          BVS    99$
2860 010622 004737 005056      JSR    PC,TIMOK     ;GO CHECK TIME RECORDED
2861 010626 000401          BR     100$
2862
2863 010630 104400      99$:   HLT
2864 010632 104000      100$: SCOPE        ;REPORT ERROR
2865
2866          ;TEST 005 - READ FROM BOT
2867          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
2868 010634 112737 000005 001126 TST005: MOVB    #5,@TSTNUM
2869 010642 004737 005706      JSR    PC,.REWIND   ;SET TEST #5
2870 010646 102422          BVS    99$         ;REWIND SLAVE
2871 010650 004737 006006      JSR    PC,READ      ;BRANCH IF ERROR ON REWIND
2872 010654 012702 010666      MOV    #1$,R2      ;SET RETURN PC FROM TIMER
```

```

2873 010660 004737 004730          JSR    PC,TIMON          ;TURN TIMER ON
2874 010664 005215                   INC    (R5)              ;SET 'GO' BIT
2875                                     1$:   TST    TC(R5)        ;BRANCH WHEN 'ACCL' RESETS
2876 010666 005765 000032          BPL    2$                ;
2877 010672 100002                   JMP    TIMER(R3)        ;GO TO TIMER & RETURN VIA R2
2878 010674 000163 005020          2$:   JSR    PC,WAITRDY   ;WAIT FOR READY
2879                                     BVS    99$              ;BRANCH IF ERROR
2880 010700 004737 005602          JSR    PC,TIMOK         ;CHECK RECORDED TIME
2881 010704 102403                   BR     100$             ;
2882 010706 004737 005056          99$:  HLT
2883 010712 000401                   100$: SCOPE
2884
2885 010714 104400                   ;TEST 006 - READ START
2886 010716 104000                   ;THIS TEST MEASURES TIME FROM 'GO'-1 TO 'ACCL'=0.
2887                                     TST006: MOVB   #6, @TSTNUM ;SET TEST #6
2888                                     JSR    PC,WRT.BK        ;WRITE A RECORD & BACK SPACE
2889                                     BVS    99$
2890 010720 112737 000006 001126   JSR    PC,READ         ;
2891 010726 004737 006062          MOV    #1$,R2          ;SET RETURN PC FROM TIMER
2892 010732 102422                   JSR    PC,TIMON        ;TURN TIMER ON
2893 010734 004737 006006          INC    (R5)           ;SET 'GO' BIT
2894 010740 012702 010752          1$:   TST    TC(R5)        ;BRANCH WHEN 'ACCL' RESETS
2895 010744 004737 004730          BPL    2$                ;
2896 010750 005215                   JMP    TIMER(R3)        ;GO TO TIMER & RETURN VIA R2
2897                                     2$:   JSR    PC,WAITRDY   ;WAIT FOR READY
2898 010752 005765 000032          BVS    99$              ;BRANCH IF ERROR
2899 010756 100002                   JSR    PC,TIMOK         ;CHECK RECORDED TIME
2900 010760 000163 005020          BR     100$             ;
2901                                     99$:  HLT
2902 010764 004737 005602          100$: SCOPE
2903 010770 102403
2904 010772 004737 005056          ;TEST 007 - READ SHUTDOWN
2905 010776 000401                   ;THIS TEST MEASURES TIME FROM 'FC REG'=FRAME COUNT TO 'SWDN'=1.
2906                                     TST007: MOVB   #7, @TSTNUM ;SET TEST #7
2907 011000 104400                   JSR    PC,WRT.BK        ;WRITE A RECORD & BACK SPACE
2908 011002 104000                   BVS    99$              ;BRANCH IF ERROR
2909                                     JSR    PC,READ         ;
2910                                     INC    (R5)           ;SET 'GO' BIT
2911                                     1$:   CMP    #-FRMCNT, (R0) ;WAIT FOR FRAME COUNT TO
2912 011004 112737 000007 001126   BEQ    2$                ;- # OF FRAMES WRITTEN
2913 011012 004737 006062          BIT    #ERR, (R1)       ;MONITOR ERROR BIT
2914 011016 102430                   BNE    99$
2915 011020 004737 006006          BR     1$
2916 011024 005215                   2$:   JSR    PC,TIMON        ;TURN TIMER ON
2917                                     MOV    PC,R2          ;SET RETURN PC FROM TIMER
2918 011026 022710 000400          BIT    #SDWN, (R1)     ;BRANCH WHEN SDWN SETS
2919 011032 001404                   BNE    3$
2920 011034 032711 040000          3$:
2921 011040 001017
2922 011042 000771
2923
2924 011044
2925 011044 004737 004730          JSR    PC,TIMON        ;TURN TIMER ON
2926 011050 010702                   MOV    PC,R2          ;SET RETURN PC FROM TIMER
2927 011052 032711 000020          BIT    #SDWN, (R1)     ;BRANCH WHEN SDWN SETS
2928 011056 001002                   BNE    3$

```

```

2929 011060 000163 005020          JMP      TIMER(R3)          ;GO TO TIMER & RETURN VIA R2
2930
2931 011064 004737 005602          3$:     JSR      PC, WAITRDY
2932 011070 102403                    BVS     99$
2933 011072 004737 005056          JSR     PC, TIMOK
2934 011076 000401                    BR      100$
2935
2936 011100 104400          99$:    HLT
2937 011102 104000          100$:   SCOPE          ;REPORT ERROR
2938
2939
2940
2941 011104 112737 000010 001126  ;TEST 010 - HEAD SETTLEDOWN
2942 011112 012702 011170          ;THIS TEST MEASURES TIME FROM 'SWDN'=1 TO 'SWDN'=0.
2943 011116 004737 006062          TST010: MOVB    #10, R2STNUM ;SET TEST #10
2944 011122 102436                    MOV     #4$, R2          ;SET RETURN PC FROM TIMER
2945 011124 004737 006006          JSR     PC, WRT.BK      ;WRITE A RECORD & BACK SPACE
2946 011130 005215                    BVS     99$
2947
2948 011132 105711          1$:     JSR     PC, READ      ;SET 'GO' BIT
2949 011134 100404                    INC     (R5)
2950 011136 032711 040000          TSTB   (R1)          ;WAIT FOR READY
2951 011142 001026                    BMI     2$             ;BRANCH WHEN SET
2952 011144 000772                    BIT     #ERR, (R1)     ;CHECK ERROR BIT
2953
2954 011146 032711 000020          2$:     BNE     3$
2955 011152 001004                    BIT     #ERR, (R1)     ;MONITOR ERROR BIT
2956 011154 032711 040000          BNE     99$
2957 011160 001017                    BR      2$
2958 011162 000771
2959
2960 011164          3$:
2961 011164 004737 004730          JSR     PC, TIMON      ;TURN TIMER ON
2962 011170 032765 000020 000012 4$:     BIT     #SDWN, DS(R5)   ;WAIT FOR NEGATION OF SDWN
2963 011176 001402                    BEQ     5$
2964 011200 000163 005020          JMP     TIMER(R3)      ;GO TO TIMER & RETURN VIA R2
2965
2966 011204 004737 005602          5$:     JSR     PC, WAITRDY
2967 011210 102403                    BVS     99$
2968 011212 004737 005056          JSR     PC, TIMOK
2969 011216 000401                    BR      100$
2970
2971 011220 104400          99$:    HLT
2972 011222 104000          100$:   SCOPE
2973
2974
2975
2976
2977 011224 112737 000011 001126  ;TEST 011-READ REVERSE START
2978 011232 012702 011270          ;THIS TEST MEASURES TIME FROM 'GO' 1 TO 'ACCL'=0.
2979 011236 004737 005770          TST011: MOVB    #11, R2STNUM ;SET RETURN PC FROM TIMER
2980 011242 005215                    MOV     #1$, R2          ;WRITE A RECORD
2981 011244 004737 005602          JSR     PC, WRITE
2982 011250 102422                    INC     (R5)          ;SET 'GO' BIT
2983 011252 004737 005310          JSR     PC, WAITRDY
2984 011256 004737 006024          JSR     PC, DELAY      ;WAIT FOR TAPE MOTION TO STOP
          JSR     PC, REVDR
  
```

```
2985 011262 004737 004730 JSR PC,TIMON ;TURN TIMER ON
2986 011266 005215 INC (R5) ;SET 'GO' BIT
2987
2988 011270 005765 000032 1$: TST TC(R5) ;BRANCH WHEN 'ACCL' = 0
2989 011274 100002 BPL 2$
2990 011276 000163 005020 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
2991
2992 011302 004737 005602 2$: JSR PC,WAITRDY
2993 011306 102403 BVS 99$ ;BRANCH IF ERROR
2994 011310 004737 005056 JSR PC,TIMOK
2995 011314 000401 BR 100$
2996
2997 011316 104400 99$: HLT
2998 011320 104000 100$: SCOPE
2999
3000 ;TEST 012-READ REVERSE SHUTDOWN
3001 ;THIS TEST MEASURES TIME FROM 'FC REG' = FRAME COUNT TO 'SDWN' 1.
3002 011322 112737 000012 001126 TST012: MOVB #12,@TSTNUM
3003 011330 012702 011400 MOV #3$,R2 ;SET RETURN PC FROM TIMER
3004 011334 004737 005770 JSR PC,WRITE ;WRITE A RECORD
3005 011340 005215 INC (R5) ;SET 'GO' BIT
3006 011342 004737 005602 JSR PC,WAITRDY
3007 011346 102427 BVS 99$
3008 011350 004737 006024 JSR PC,REVRD
3009 011354 005215 INC (R5) ;SET 'GO' BIT
3010
3011 011356 022710 000400 1$: CMP #-FRMCNT,(R0) ;BRANCH WHEN FRAME COUNT
3012 011362 001404 BEQ 2$ ;= # OF RECORD WRITTEN
3013 011364 032711 040000 BIT #ERR,(R1) ;MONITOR ERROR BIT IN 'DS' REG
3014 011370 001016 BNE 99$
3015 011372 000771 BR 1$
3016
3017 011374 2$:
3018 011374 004737 004730 JSR PC,TIMON ;TURN TIMER ON
3019 011400 032711 000020 3$: BIT #SDWN,(R1) ;BRANCH WHEN SDWN SETS
3020 011404 001002 BNE 4$
3021 011406 000163 005020 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3022
3023 011412 004737 005602 4$: JSR PC,WAITRDY ;WAIT FOR READY
3024 011416 102403 BVS 99$
3025 011420 004737 005056 JSR PC,TIMOK
3026 011424 000401 BR 100$
3027
3028 011426 104400 99$: HLT
3029 011430 104000 100$: SCOPE
3030
3031 ;TEST 013-READ REVERSE SETTLEDOWN
3032 ;THIS TEST MEASURES TIME FROM 'SDWN'-1 TO 'SDWN'=0.
3033 011432 112737 000013 001126 TST013: MOVB #13,@TSTNUM
3034 011440 012702 011524 MOV #4$,R2 ;SET RETURN PC FROM TIMER
3035 011444 004737 005770 JSR PC,WRITE ;WRITE A RECORD
3036 011450 005215 INC (R5) ;SET 'GO' BIT
3037 011452 004737 005602 JSR PC,WAITRDY
3038 011456 102435 BVS 99$
3039 011460 004737 006024 JSR PC,REVRD
3040 011464 005215 INC (R5) ;SET 'GO' BIT
```

```

3041
3042 011466 105711          1$:   TSTB   (R1)           ;BRANCH WHEN
3043 011470 100404          BMI    2$           ;READY SETS
3044 011472 032711 040000  BIT    #ERR,(R1)
3045 011476 001005          BNE    99$
3046 011500 000772          BR     1$
3047
3048 011502 032711 000020  2$:   BIT    #SDWN,(R1)
3049 011506 001004          BNE    3$
3050 011510 032711 040000  BIT    #ERR,(R1)
3051 011514 001016          BNE    99$
3052 011516 000771          BR     2$
3053
3054 011520
3055 011520 004737 004730  3$:   JSR    PC,TIMON      ;TURN TIMER ON
3056 011524 032711 000020  4$:   BIT    #SDWN,(R1)   ;BRANCH WHEN SWDN 0
3057 011530 001402          BEQ    5$
3058 011532 000163 005020  JMP    TIMER(R3)      ;GO TO TIMER & RETURN VIA R2
3059
3060 011536 004737 005602  5$:   JSR    PC,WAITRDY    ;WAIT FOR READY
3061 011542 102403          BVS    99$
3062 011544 004737 005056  JSR    PC,TIMOK
3063 011550 000401          BR     100$
3064
3065 011552 104400          99$:   HLT
3066 011554 104000          100$:  SCOPE
3067
3068          ;REWIND DRIVE
3069 011556          A:
3070 011556 004737 005706  JSR    PC,.REWIND     ;REWIND SLAVE
3071 011562 102401          BVS    99$           ;BRANCH IF ERROR ON REWIND
3072 011564 102002          BVC    100$
3073 011566 104400          99$:   HLT
3074 011570 000772          BR     A
3075 011572          100$:
3076
3077          ;TEST 014-TURN AROUND DELAY (FORWARD-REVERSE)
3078          ;THIS TEST MEASURES TIME FROM 'GO'=1 (READ REVERSE) TO 'ACCL'=0
3079 011572 112737 000014 001126 TST014: MOVB   #14,R2STNUM
3080 011600 012702 011632  MOV    #2$,R2        ;SET RETURN PC FROM TIMER
3081 011604 004737 005770  JSR    PC,WRITE      ;WRITE A RECORD
3082 011610 005215          INC    (R5)         ;SET 'GO' BIT
3083 011612 004737 005602  JSR    PC,WAITRDY
3084 011616 102420          BVS    99$
3085
3086 011620 004737 006024  1$:   JSR    PC,REVRD     ;READ THE RECORD (REVERSE)
3087 011624 004737 004730  JSR    PC,TIMON      ;TURN TIMER ON
3088 011630 005215          INC    (R5)         ;SET 'GO' BIT
3089
3090 011632 005765 000032  2$:   TST    TC(R5)      ;WAIT FOR 'ACCL' - 0
3091 011636 100002          BPL    3$
3092 011640 000163 005020  JMP    TIMER(R3)     ;GO TO TIMER & RETURN VIA R2
3093
3094 011644 004737 005602  3$:   JSR    PC,WAITRDY
3095 011650 102403          BVS    99$
3096 011652 004737 005056  JSR    PC,TIMOK

```

```
3097 011656 000401 BR 100$
3098
3099 011660 104400 99$: HLT
3100 011662 104000 100$: SCOPE
3101
3102 ;TEST 015- TURN AROUND DELAY (REVERSE-FORWARD)
3103 ;THIS TEST MEASURES TIME FROM 'GO'=1 (READ) TO 'ACCL'=0.
3104 011664 112737 000015 001126 TST015: MOVB #15,@TSTNUM
3105 011672 012702 011740 MOV #2$,R2 ;SET RETURN PC FROM TIMER
3106 011676 004737 005770 JSR PC,WRITE ;WRITE A RECORD
3107 011702 005215 INC (R5) ;SET 'GO' BIT
3108 011704 004737 005602 JSR PC,WAITRDY ;WAIT FOR READY
3109 011710 102426 BVS 99$
3110 011712 004737 006024 JSR PC,REVRD ;READ A RECORD IN THE
3111 011716 005215 INC (R5) ;SET 'GO' BIT
3112
3113 011720 004737 005602 JSR PC,WAITRDY
3114 011724 102420 BVS 99$
3115
3116 011726 004737 006006 1$: JSR PC,READ ;READ RECORD FORWARD
3117 011732 004737 004730 JSR PC,TIMON ;TURN TIMER ON
3118 011736 005215 INC (R5) ;SET 'GO' BIT
3119
3120 011740 005765 000032 2$: TST TC(R5) ;WAIT FOR 'ACCL' - 0
3121 011744 100002 BPL 3$
3122 011746 000163 005020 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3123
3124 011752 004737 005602 3$: JSR PC,WAITRDY
3125 011756 102403 BVS 99$
3126 011760 004737 005056 JSR PC,TIMOK
3127 011764 000401 BR 100$
3128
3129 011766 104400 99$: HLT
3130 011770 104000 100$: SCOPE
3131
3132 ;TEST 016-GAP SIZE (STOP HALF)
3133 011772 112737 000016 001126 TST016: MOVB #16,@TSTNUM
3134 012000 012702 012036 MOV #1$,R2 ;SET RETURN PC FROM TIMER
3135 012004 004737 005770 JSR PC,WRITE ;WRITE A RECORD
3136 012010 005215 INC (R5) ;SET 'GO' BIT
3137 012012 004737 005602 JSR PC,WAITRDY
3138 012016 102421 BVS 99$
3139 012020 004737 005310 JSR PC,DELAY ;DELAY 350 MS
3140 012024 004737 006024 JSR PC,REVRD ;READ REVERSE RECORD
3141 012030 004737 004730 JSR PC,TIMON ;TURN TIMER ON
3142 012034 005215 INC (R5) ;SET 'GO' BIT
3143
3144 012036 005710 1$: TST (R0) ;WAIT FOR FRAME COUNT > 0
3145 012040 001002 BNE 2$
3146 012042 000163 005020 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3147
3148 012046 004737 005602 2$: JSR PC,WAITRDY ;WAIT FOR READY BIT TO SET
3149 012052 102403 BVS 99$
3150 012054 004737 005056 JSR PC,TIMOK ;CHECK TIME
3151 012060 000401 BR 100$
3152
```

```
3153 012062 104400          99$:  HLT
3154 012064 104000          100$: SCOPE
3155
3156          ;TEST 017-GAP SIZE (START HALF)
3157 012066 112737 000017 001126 TST017: MOVB #17,R2          ;SET RETURN PC FROM TIMER
3158 012074 012702 012146          MOV #1$,R2          ;WRITE A RECORD
3159 012100 004737 005770          JSR PC,WRITE          ;SET 'GO' BIT
3160 012104 005215          INC (R5)          ;WAIT FOR READY
3161 012106 004737 005602          JSR PC,WAITRDY
3162 012112 102427          BVS 99$
3163 012114 004737 006024          JSR PC,REVRD          ;READ REVERSE THE RECORD
3164 012120 005215          INC (R5)          ;SET 'GO' BIT
3165 012122 004737 005602          JSR PC,WAITRDY          ;WAIT FOR READY
3166 012126 102421          BVS 99$          ;BRANCH ON ERROR
3167 012130 004737 005310          JSR PC,DELAY          ;WAIT FOR TAPE MOTION TO STOP
3168 012134 004737 006006          JSR PC,READ          ;READ RECORD
3169 012140 004737 004730          JSR PC,TIMON          ;TURN TIMER ON
3170 012144 005215          INC (R5)          ;SET 'GO' BIT
3171
3172 012146 005710          1$:  TST (R0)          ;WAIT FOR FRAME COUNT > 0
3173 012150 001002          BNE 2$
3174 012152 000163 005020          JMP TIMER(R3)          ;GO TO TIMER & RETURN VIA R2
3175
3176 012156 004737 005602          2$:  JSR PC,WAITRDY          ;WAIT FOR READY
3177 012162 102403          BVS 99$
3178 012164 004737 005056          JSR PC,TIMOK          ;CHECK TIME
3179 012170 000401          BR 100$
3180
3181 012172 104400          99$:  HLT
3182 012174 104000          100$: SCOPE
3183
3184          ;TEST 020- GAP SIZE (INTERRECORD)
3185          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'FC REG' >0.
3186 012176 112737 000020 001126 TST020: MOVB #20,R2
3187 012204 012702 012266          MOV #1$,R2          ;SET RETURN PC FROM TIMER
3188 012210 004737 005770          JSR PC,WRITE          ;WRITE A RECORD
3189 012214 005215          INC (R5)          ;SET 'GO' BIT
3190 012216 004737 005602          JSR PC,WAITRDY          ;WAIT FOR READY
3191 012222 102433          BVS 99$
3192 012224 004737 005770          JSR PC,WRITE          ;WRITE SECOND RECORD
3193 012230 005215          INC (R5)          ;SET 'GO' BIT
3194 012232 004737 005602          JSR PC,WAITRDY          ;WAIT FOR READY
3195 012236 102425          BVS 99$
3196 012240 004737 006024          JSR PC,REVRD          ;READ REVERSE SECOND RECORD
3197 012244 005215          INC (R5)          ;SET 'GO' BIT
3198 012246 004737 005602          JSR PC,WAITRDY          ;WAIT FOR READY
3199 012252 102417          BVS 99$
3200 012254 004737 006024          JSR PC,REVRD          ;READ REVERSE FIRST RECORD
3201 012260 004737 004730          JSR PC,TIMON          ;TURN TIMER ON
3202 012264 005215          INC (R5)          ;SET 'GO' BIT
3203
3204 012266 005710          1$:  TST (R0)          ;WAIT FOR FRAME COUNT > 0
3205 012270 001002          BNE 2$
3206 012272 000163 005020          JMP TIMER(R3)          ;GO TO TIMER & RETURN VIA R2
3207
3208 012276 004737 005602          2$:  JSR PC,WAITRDY          ;WAIT FOR READY
```

3209 012302 102403
3210 012304 004737 005056
3211 012310 000401

BVS 99\$
JSR PC,TIMOK
BR 100\$

3213 012312 104400
3214 012314 104000

99\$: HLT
100\$: SCOPE

3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227

;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
;BETWEEN 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.

3228 012316 112737 000021 001126
3229 012324 012702 012462
3230 012330 004737 005706
3231 012334 102530
3232 012336 005037 001120
3233 012342 012700 000021
3234 012346 004737 005770
3235 012352 005215
3236 012354 004737 005602
3237 012360 102516
3238 012362 004737 005340
3239 012366 063737 001014 001120
3240 012374 005300
3241 012376 001363

TST021: MOV #21, @TSTNUM
MOV #4\$,R2 ;SET RETURN PC FROM TIMER
JSR PC,REWIND ;REWIND SLAVE
BVS 99\$;BRANCH IF ERROR ON REWIND
CLR DELTIM ;CLEAR VARIABLE DELAY TIME
MOV #17,RO ;SET # OF RECORDS TO WRITE
1\$: JSR PC,WRITE ;WRITE 17. RECORDS
INC (R5) ;SET 'GO' BIT
JSR PC,WAITRDY ;WAIT FOR READY
BVS 99\$
JSR PC,DELAYV ;DELAY BEFORE WRITING NEXT REC.
ADD GAPDEL,DELTIM ;ADD 1MS TO DELAY TIME
DEC RO ;DECREMENT RECORDS WRITTEN COUNT
BNE 1\$

3242
3243 012400 012700 000021
3244 012404 004737 006024
3245 012410 005215
3246 012412 004737 005602
3247 012416 102477
3248 012420 005300
3249 012422 001370

2\$: MOV #17,RO ;SET # OF RECS. TO REVERSE READ
JSR PC,REVRD ;REVERSE READ 17. RECORDS
INC (R5) ;SET 'GO' BIT
JSR PC,WAITRDY ;WAIT FOR READY
BVS 99\$
DEC RO ;DECREMENT RECORD COUNT
BNE 2\$

3250
3251 012424 012700 000020
3252 012430 012701 001060
3253 012434 004737 006006
3254 012440 005215

MOV #16,RO ;SET # OF RECORDS TO READ
MOV #GAPTBL,R1 ;SET PTR TO GAP TABLE FOR TEST
JSR PC,READ ;READ A RECORD
INC (R5) ;SET 'GO' BIT

3255
3256 012442 004737 005602
3257 012446 102463
3258 012450 004737 006006
3259 012454 004737 004730
3260 012460 005215

3\$: JSR PC,WAITRDY ;WAIT FOR READY
BVS 99\$
JSR PC,READ ;READ NEXT RECORD
JSR PC,TIMON ;TURN TIMER ON
INC (R5) ;SET 'GO' BIT

3261
3262 012462 005765 000006
3263 012466 001002
3264 012470 000163 005020

4\$: TST FC(R5) ;WAIT FOR FRAME COUNT > 0
BNE 5\$
JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2

```
3265
3266 012474 004737 005602      5$: JSR PC, WAITRDY      .WAIT FOR READY
3267 012500 102446              BVS 99$
3268 012502 010421              MOV R4, (R1)+          ;STORE TIME IN GAPTBL
3269 012504 005300              DEC R0                 ;DECREMENT # OF RECORDS READ
3270 012506 001355              BNE 3$
3271
3272 012510 105037 001124      CLR B @GAP            ;SET GAP # 0
3273 012514 012700 000020      MOV #16., R0
3274 012520 012701 001060      MOV #GAPTBL, R1
3275
3276 012524 012104              6$: MOV (R1)+, R4       ;GET GAP TICK COUNT
3277 012526 004737 005166      JSR PC, GAPOK         ;CHECK TIME
3278 012532 105237 001124      INCB @GAP             ;INCREMENT GAP #
3279 012536 122737 000020 001124 CMPB #16., @GAP       ;BRANCH IF ALL GAPS NOT CHECKED
3280 012544 001367              BNE 6$
3281
3282 012546 012700 000020      MOV #16., R0          ;SETUP TO AVERAGE GAP SIZES
3283 012552 012701 001060      MOV #GAPTBL, R1      ;SET PTR TO TABLE
3284 012556 005002              CLR R2                ;CLEAR 'SUM' REGISTERS
3285 012560 005003              CLR R3
3286 012562 062102              7$: ADD (R1)+, R2     ;ADD ALL GAP SIZES TOGETHER
3287 012564 005503              ADC R3
3288 012566 005300              DEC R0
3289 012570 001374              BNE 7$
3290 012572 012700 000004      MOV #4., R0           ;NOW DIVIDE BY 16.
3291 012576 006203              8$: ASR R3             ;BY SHIFTING 4 PLACES RIGHT
3292 012600 006002              ROR R2
3293 012602 005300              DEC R0
3294 012604 001374              BNE 8$
3295 012606 010204              MOV R2, R4            ;MOVE AVERAGED TIMES TO R4
3296 012610 004737 005056      JSR PC, TIMOK         ;CHECK AVERAGED TIMES
3297 012614 000401              BR 100$
3298
3299 012616 104400              99$: HLT
3300 012620 104000              100$: SCOPE
3301
3302
3303
3304
3305 ;TEST 022-DATA TIME (800BPI)
3306 012622 112737 000022 001126 ;THIS TEST MEASURES THE TIME FROM FC REG >-6400 TO 'RDY' - 1.
3307 012630 012702 012710      TST022: MOV B #022, @TSTNUM
3308 012634 004737 005706      MOV #3$, R2           ;SET RETURN PC FROM TIMER
3309 012640 102442              JSR PC, REWIND        ;REWIND SLAVE
3310 012642 052765 001700 000032 BVS 99$               ;BRANCH IF ERROR ON REWIND
3311 012650 004337 006124      BIS #NORM11, TC(R5)  ;SET 800 BPI
3312 012654 016674              JSR R3, TMCMD         ;WRITE 3200. WORD RECORD
3313 012656 171600              .WORD WTBUF
3314 012660 163400              .WORD -3200.
3315 012662 000060              .WORD -6400.
3316 012664 005215              .WORD WFW
3317 INC (R5)                ;SET 'GO' BIT
3318 012666 022710 163400      1$: CMP #-6400., (R0) ;WAIT FOR WRITING TO START
3319 012672 001004              BNE 2$
3320 012674 032711 040000      BIT #ERR, (R1)       ;MONITOR ERROR BIT
```

```

3321 012700 001022          BNE 99$
3322 012702 000771          BR 1$
3323
3324 012704          2$:
3325 012704 004737 004730      JSR PC,TIMON          ;TURN TIMER ON
3326 012710 105711          TSTB (R1)           ;BRANCH WHEN READY SETS
3327 012712 100402          BMI 4$
3328 012714 000163 005020      JMP TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
3329
3330 012720 012700 000003      4$: MOV #3,R0          ;SET SHIFT COUNT
3331 012724 006204          5$: ASR R4
3332 012726 005300          DEC R0
3333 012730 001375          BNE 5$
3334 012732 004737 005602      JSR PC,WAITRDY
3335 012736 102403          BVS 99$
3336 012740 004737 005056      JSR PC,TIMOK        ;CHECK TIME
3337 012744 000401          BR 100$
3338
3339 012746 104400          99$: HLT
3340 012750 104000          100$: SCOPE
3341
3342          ;TEST 023-DATA TIME (1600BPI)
3343          ;THIS TEST MEASURES THE TIME FROM FC REG >-6400 TO 'RDY' - 1.
3344 012752 112737 000023 001126  TST023: MOVB #023,#TSTNUM
3345 012760 012702 013046          MOV #3$,R2          ;SET RETURN PC FROM TIMFR
3346 012764 004737 005706          JSR PC,.REWIND      ;REWIND SLAVE
3347 012770 102442          BVS 99$             ;BRANCH IF ERROR ON REWIND
3348 012772 042765 003700 000032  BIC #3700,TC(R5)    ;CLEAR CURRENT DENSITY
3349 013000 052765 002300 000032  BIS #PE1600,TC(R5)  ;SET 1600 BPI
3350 013006 004337 006124          JSR R3,TMCMDB       ;WRITE 3200. WORD RECORD
3351 013012 016674          .WORD WTBUFF
3352 013014 171600          .WORD -3200.
3353 013016 163400          .WORD -6400.
3354 013020 000060          .WORD WFWD
3355 013022 005215          INC (R5)           ;SET 'GO' BIT
3356
3357 013024 022710 163400          1$: (MP #-6400.,(R0) ;BRANCH WHEN WRITING STARTS
3358 013030 001004          BNE 2$
3359 013032 032711 040000          BIT #ERR,(R1)      ;MONITOR ERROR BIT
3360 013036 001017          BNE 99$
3361 013040 000771          BR 1$
3362
3363 013042          2$:
3364 013042 004737 004730      JSR PC,TIMON          ;TURN TIMER ON
3365 013046 105711          3$: TSTB (R1)           ;BRANCH WHEN READY SETS
3366 013050 100402          BMI 4$
3367 013052 000163 005020      JMP TIMER(R3)       ;GO TO TIMER & RETURN VIA R2
3368
3369 013056 006204          4$: ASR R4          ;DIVIDE TIME BY 4
3370 013060 006204          ASR R4
3371 013062 004737 005602      JSR PC,WAITRDY
3372 013066 102403          BVS 99$
3373 013070 004737 005056      JSR PC,TIMOK        ;CHECK TIME
3374 013074 000401          BR 100$
3375
3376 013076 104400          99$: HLT
  
```

```

3377 013100 104000          100$: SCOPE
3378
3379                          ;TEST 024-ERASE
3380                          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'RDY'=1.
3381 013102 112737 000024 001126 TST024: MOVB #24,@TSTNUM
3382 013110 012702 013172          MOV #2$,R2 ;SET RETURN PC FROM TIMER
3383 013114 004737 005706          JSR PC,REWIND ;REWIND SLAVE
3384 013120 102436          BVS 99$ ;BRANCH IF ERROR ON REWIND
3385 013122 004737 005544          JSR PC,RHINIT ;SET NRZ
3386 013126 004737 005770          JSR PC,WRITE ;WRITE A RECORD
3387 013132 005215          INC (R5) ;SET 'GO' BIT
3388 013134 004737 005602          JSR PC,WAITRDY
3389 013140 102426          BVS 99$
3390 013142 012737 013150 001002 MOV #1$,@WSCPADR
3391 013150 004337 006124 1$: JSR R3,@TMCMD
3392 013154 000000          .WORD 0
3393 013156 000000          .WORD 0
3394 013160 000000          .WORD 0
3395 013162 000024          .WORD ERASE
3396 013164 004737 004730          JSR PC,TIMON ;TURN TIMER ON
3397 013170 005215          INC (R5) ;SET 'GO' BIT
3398
3399          2$: TSTB (R1) ;BRANCH WHEN READY SETS
3400 013174 100402          BMI 3$
3401 013176 000163 005020          JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3402
3403          3$: JSR PC,WAITRDY
3404 013206 102403          BVS 99$
3405 013210 004737 005056          JSR PC,TIMOK
3406 013214 000401          BR 100$
3407
3408          99$: HLT
3409 013220 104000          100$: SCOPE
3410
3411                          ;TEST 025 TAPE MARK
3412                          ;THIS TEST MEASURES TIME FROM 'GO' 1 TO 'RDY'-1.
3413 013222 112737 000025 001126 TST025: MOVB #25,@TSTNUM
3414 013230 012702 013272          MOV #1$,R2 ;SET RETURN PC FROM TIMER
3415 013234 004737 005770          JSR PC,WRITE ;WRITE A RECORD
3416 013240 005215          INC (R5) ;SET 'GO' BIT
3417 013242 004737 005602          JSR PC,WAITRDY
3418 013246 102423          BVS 99$
3419 013250 004337 006124          JSR R3,@TMCMD
3420 013254 000000          .WORD 0
3421 013256 000000          .WORD 0
3422 013260 000000          .WORD 0
3423 013262 000026          .WORD WFMK
3424 013264 004737 004730          JSR PC,TIMON ;TURN TIMER ON
3425 013270 005215          INC (R5) ;SET 'GO' BIT
3426
3427          1$: TSTB (R1) ;BRANCH WHEN READY SETS
3428 013274 100402          BMI 2$
3429 013276 000163 005020          JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3430
3431          2$: JSR PC,WAITRDY
3432 013306 102403          BVS 99$
  
```

CZTEECO TMO3-TE16/TU77 DFT
CZTEEC.P11 22-AUG-79 08.40

MACY11 30A(1052) 22-AUG-79 09:00 J 6 PAGE 74
START OF TESTS

SEQ 0074

3433 013310 004757 005056
3434 013314 000401
3435
3436 013316 104400
3437 013320
3438 013320 004737 005706
3439 013324 102774
3440 013326 104000
3441

JSR PC.TIMOK
BR 100\$

99\$: HLT
100\$:

JSR PC..REWIND
BVS 99\$
SCOPE

:REWIND SLAVE
:BRANCH IF ERROR ON REWIND

```

3442 013330 032777 002000 165442 FINISH: BIT #SW10,@SWR ;DO NOT SPACF PAPER
3443 013336 001011 BNE 2$ ;IF USER SELECTED NO OUTPUT
3444 013340 005737 006372 TST CHNFLG ;OR IF IN CH-IN MODE
3445 013344 001006 BNE 2$
3446 013346 012700 000012 MOV #10.,R0 ;SET LINE FEED COUNT
3447 013352 000004 001402 1$: TYPE,CRLF
3448 013356 005300 DEC R0
3449 013360 001374 BNE 1$
3450
3451
3452 013362 105237 001005 2$: INCB @#SLVNUM ;SET NEXT SLAVE #
3453 013366 005237 001006 INC @#SLVPTR ;AND ITS POINTER
3454 013372 122737 000010 001005 CMPB #8.,@#SLVNUM ;BRANCH IF LAST SLAVE (7)
3455 013400 001402 BEQ 3$
3456 013402 000137 007442 JMP @#BEGIN ;BEGIN TEST ON NEXT SLAVE
3457 013406 105037 001005 3$: CLRB @#SLVNUM ;SET SLAVE #0
3458 013412 105237 001004 INCB @#DRVNUM ;AND INCREMENT DRIVE #
3459 013416 122737 000010 001004 CMPB #8.,@#DRVNUM ;AND CHECK IF LAST DRIVE
3460 013424 001402 BEQ END
3461 013426 000137 007442 JMP @#BEGIN
3462
3463 013432 105737 001132 END: TSTB @#UNTFND ;BRANCH IF A UNIT WAS FOUND
3464 013436 001004 BNE 1$
3465 013440 000004 015044 TYPE,E.UNIT
3466 013444 000137 006274 JMP @#INIT
3467 013450 105237 001134 1$: INCB @#PSCNT ;INCREMENT PASS COUNT
3468 013454 000004 014275 TYPE,M.EOP
3469 013460 113702 001134 MOVB @#PSCNT,R2 ;GET PASSCOUNT
3470 013464 004737 003134 JSR PC,TYPOCT ;AND TYPE IT
3471 013470 000004 001402 TYPE,CRLF
3472 013474 013700 000042 MOV @#42,R0 ;GET ACT11 RETURN ADDRESS
3473 013500 001405 BEQ HERE ;BRANCH IF NOT ACT11
3474 013502 000005 RESET
3475 013504 004710 $ENDAD: JSR PC,(R0)
3476 013506 000240 NOP
3477 013510 000240 NOP
3478 013512 000240 NOP
3479 013514 000240 HERE: NOP
3480 013516 005737 006372 TST CHNFLG ;BRANCH IF CHAIN MODE
3481 013522 001004 BNE 1$
3482 013524 032777 000100 165246 BIT #SW06,@SWR ;BRANCH IF NOT CONTINOUS LOOP
3483 013532 001402 BEQ 2$
3484 013534 000137 007414 1$: JMP @#RSTRT ;RESTART
3485 013540 000000 2$: HALT
3486 013542 000005 RESET
3487 013544 000137 006274 JMP @#INIT ;RESTART

```

```

3488      :SKEW TAPE TIMING TESTS
3489      :THE FOLLOWING TESTS REQUIRE A SPECIALLY WRITTEN 800 BPI SKEW TAPE
3490 013550 012737 013556 001002 SKEWTST:MOV #TST026,@#SCPADR ;SET SCOPE POINTER
3491
3492      :TEST 026- SKEW TAPE SPEED TEST-FORWARD
3493      :THIS TEST READS 32" OF TAPE (26400.-800. = 25600. FRAMES), THEN
3494      :DIVIDES TIME BY 32. TO GET TIME TO READ 1" (800. FRAMES) OF TAPE.
3495 013556 112737 000026 001126 TST026: MOV #26,@#TSTNUM
3496 013564 012702 013652      MOV #2$,R2 ;SET RETURN PC FROM TIMER
3497 013570 004737 005706      JSR PC,.REWIND ;REWIND SLAVE
3498 013574 102445      BVS 99$ ;BRANCH IF ERROR ON REWIND
3499 013576 052765 001700 000032      BIS #NORM11,TC(R5) ;SET 800 BPI
3500 013604 052765 000010 000010      BIS #BAI,CS2(R5) ;INHIBIT BUS ADDRESS INCREMENT
3501 013612 004337 006124      JSR R3,@#TMCMD ;READ 32" OF TAPE-FORWARD
3502 013616 016674      .WORD RDBUF
3503 013620 177777      .WORD -1.
3504 013622 063440      10$: .WORD 26400. ;FRAME COUNT
3505 013624 000070      .WORD RDFWD
3506 013626 005215      INC (R5) ;SET 'GO' BIT
3507
3508 013630 032765 075027 000014 1$: BIT #HRDERR,ER(R5) ;BRANCH IF ANY HARD ERROR BIT SETS
3509 013636 001024      BNE 99$
3510 013640 022710 001440      CMP #800.,(R0) ;WAIT FOR FIRST 800 FRAMES
3511 013644 101371      BHI 1$ ;TO BE READ
3512
3513 013646 004737 004730      JSR PC,TIMON ;TURN TIMER ON
3514 013652 023710 013622      2$: CMP @#10$, (R0) ;WAIT FOR READING TO FINISH
3515 013656 103402      BLO 3$
3516 013660 000163 005020      JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3517
3518 013664 012700 000005      3$: MOV #5,R0 ;DIVIDE TIME BY 32.
3519 013670 006204      4$: ASR R4
3520 013672 005300      DEC R0
3521 013674 001375      BNE 4$
3522 013676 004737 005544      JSR PC,RHINIT ;INIT DRIVE
3523 013702 004737 005056      JSR PC,TIMOK ;CHECK TIME
3524 013706 000401      BR 100$
3525
3526 013710 104400      99$: HLT
3527 013712 104000      100$: SCOPE
3528
3529      :TEST 027-SKEW TAPE SPEED TEST-REVERSE
3530      :THIS TEST READS FORWARD 40" (32000. FRAMES) OF TAPE, THEN READS REVERSE
3531      :32" (26400.-800. = 25600. FRAMES) OF TAPE. THE TIME IS THEN DIVIDED BY
3532      :32. TO GET TIME TO READ 1" (800. FRAMES) OF TAPE.
3533 013714 112737 000027 001126 TST027: MOV #27,@#TSTNUM
3534 013722 012702 014060      MOV #3$,R2 ;SET RETURN PC FROM TIMER
3535 013726 004737 005706      JSR PC,.REWIND ;REWIND SLAVE
3536 013732 102471      BVS 99$ ;BRANCH IF ERROR ON REWIND
3537 013734 052765 001700 000032      BIS #NORM11,TC(R5)
3538 013742 052765 000010 000010      BIS #BAI,CS2(R5)
3539 013750 004337 006124      JSR R3,@#TMCMD ;READ FORWARD 32000. FRAMES
3540 013754 016674      .WORD RDBUF
3541 013756 177777      .WORD -1. ;WORD COUNT
3542 013760 076400      10$: .WORD 32000. ;FRAME COUNT
3543 013762 000070      .WORD RDFWD ;READ FORWARD
  
```

```

3544 013764 005215          INC      (R5)          ;SET 'GO' BIT
3545
3546 013766 032711 040000    1$:    BIT      #ERR,(R1)      ;BRANCH IF ERROR BITS SETS
3547 013772 001051          BNE     99$
3548 013774 023710 013760    CMP     @#10$, (R0)
3549 014000 101372          BHI     1$
3550
3551 014002 004737 005544    JSR     PC,RHINIT      ;INIT DRIVE
3552 014006 004737 005310    JSR     PC,DELAY      ;WAIT FOR TAPE MOTION TO STOP
3553 014012 052765 000010 000010  BIS     #BAI,CS2(R5)   ;INHIBIT BUS ADDRESS INCREMENT
3554 014020 004337 006124    JSR     R3,@#TMCMD    ;READ REVERSE 32" OF TAPE
3555 014024 016674          .WORD  RDBUF          ;READ BUFFER
3556 014026 177777          .WORD  -1             ;WORD COUNT
3557 014030 063440          11$:   .WORD  26400      ;FRAME COUNT
3558 014032 000076          .WORD  RDREV          ;READ REVERSE
3559 014034 005215          INC     (R5)          ;SET 'GO' BIT
3560
3561 014036 032765 075027 000014  2$:    BIT      #HRDERR,ER(R5) ;EXIT TEST IF ERROR BIT SETS
3562 014044 001024          BNE     99$
3563 014046 022710 001440    CMP     #800., (R0)   ;WAIT FOR FIRST 800 FRAMES
3564 014052 101371          BHI     2$           ;TO BE READ
3565
3566 014054 004737 004730    JSR     PC,TIMON      ;TURN TIMER ON
3567 014060 023710 014030    3$:    CMP     @#11$, (R0)  ;WAIT FOR ALL FRAMES TO BE READ
3568 014064 103402          BLO     4$
3569 014066 000163 005020    JMP     TIMER(R3)    ;GO TO TIMER & RETURN VIA R2
3570
3571 014072 012700 000005    4$:    MOV     #5,R0      ;DIVIDE TIME BY 32.
3572 014076 006204          5$:    ASR     R4
3573 014100 005300          DEC     R0
3574 014102 001375          BNE     5$
3575 014104 004737 005544    JSR     PC,RHINIT
3576 014110 004737 005056    JSR     PC,TIMOK
3577 014114 000401          BR     100$
3578
3579 014116 104400          99$:   HLT
3580 014120          100$:
3581 014120 004737 005706    JSR     PC,.REWIND   ;REWIND SLAVE
3582 014124 102774          BVS     99$         ;BRANCH IF ERROR ON REWIND
3583 014126 104000          SCOPE
3584
3585 014130 000137 013330    JMP     @#FINISH
3586
3587
3588

```

3589				
3590				
3591	014134	005015	046524	031460
3592	014142	052055	030505	027466
3593	014150	052524	033467	042040
3594	014156	044522	042526	043040
3595	014164	047125	052103	047511
3596	014172	020116	044524	042515
3597	014200	020122	041450	052132
3598	014206	042505	030103	051
3599	014213	015	052012	050131
3600	014220	020105	041474	037122
3601	014226	052040	020117	042524
3602	014234	046522	047111	052101
3603	014242	020105	042522	050123
3604	014250	047117	042523	023040
3605	014256	057040	020103	047524
3606	014264	051040	051505	040524
3607	014272	052122	000	
3608	014275	015	042412	042116
3609	014302	047440	020106	040520
3610	014310	051523	000040	
3611	014314	005015	040510	042122
3612	014322	040527	042522	051440
3613	014330	051127	044440	020116
3614	014336	051525	006505	000012
3615	014344	005015	042522	047515
3616	014352	042526	052040	042115
3617	014360	020120	051106	046517
3618	014366	051440	040514	042526
3619	014374	030040	020056	046103
3620	014402	040505	020122	047514
3621	014410	027103	032040	027060
3622	014416	000		
3623	014417	015	052012	050131
3624	014424	020105	044506	051522
3625	014432	020124	042101	051104
3626	014440	051505	020123	043117
3627	014446	041440	047117	051124
3628	014454	046117	042514	020122
3629	014462	000040		
3630	014464	054524	042520	052040
3631	014472	030115	020063	051104
3632	014500	053111	020105	023443
3633	014506	020123	047524	041040
3634	014514	020105	042524	052123
3635	014522	042105	020072	040440
3636	014530	046114	000040	
3637	014534	047506	020122	046524
3638	014542	031460	042040	044522
3639	014550	042526	040	
3640	014553	060	020055	054524
3641	014560	042520	051440	040514
3642	014566	042526	021440	051447
3643	014574	052040	020117	042502
3644	014602	052040	051505	042524

```

.SBTTL PROGRAM MESSAGES
:OPERATOR INSTRUCTIONS
M.NAM: .ASCII <CR><LF>'TM03-TE16/TU77 DRIVE FUNCTION TIMER (CZTEECO)';++B

.ASCIZ <CR><LF>'TYPE <CR> TO TERMINATE RESPONSE & ^C TO RESTART'

M.EOP: .ASCIZ <CR><LF>'END OF PASS '

M.HSWR: .ASCIZ <CR><LF>'HARDWARE SWR IN USE'<CR><LF>

I.REM: .ASCIZ <CR><LF>'REMOVE TMDP FROM SLAVE 0. CLEAR LOC. 40.'

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

I.DRVS: .ASCIZ %T'PE TM03 DRIVE #'S TO BE TESTED: ALL %

I.SLVS: .ASCII 'FOR TM03 DRIVE '

I.DRV: .ASCIZ %0- TYPE SLAVE #'S TO BE TESTED: ALL %

```

3645	014610	035104	040440	046114	
3646	014616	000040			
3647	014620	050123	042505	020104	I.SPD: .ASCIZ 'SPEED TESTS? (YES/NO): NO '
3648	014626	042524	052123	037523	
3649	014634	024040	042531	027523	
3650	014642	047516	035051	047040	
3651	014650	020117	000		
3652	014653	015	042412	042116	M.EOT: .ASCIZ <CR><LF>'END OF TAPE'<CR><LF>
3653	014660	047440	020106	040524	
3654	014666	042520	005015	000	
3655					
3656					:ERROR MESSAGES
3657	014673	015	052012	040522	E.TRP4: .ASCIZ <CR><LF>'TRAPPED TO 4'
3658	014700	050120	042105	052040	
3659	014706	020117	000064		
3660	014712	047516	041440	047117	E.NCON: .ASCIZ 'NO CONTROLLER AT ADDRESS SPECIFIED'<CR><LF>
3661	014720	051124	046117	042514	
3662	014726	020122	052101	040440	
3663	014734	042104	042522	051523	
3664	014742	051440	042520	044503	
3665	014750	044506	042105	005015	
3666	014756	000			
3667	014757	124	030115	020063	E.NDRV: .ASCIZ 'TM03 DRIVE '
3668	014764	051104	053111	020105	
3669	014772	000			
3670	014773	104	044522	042526	E.NSLV: .ASCII 'DRIVE '
3671	015000	040			
3672	015001	060	051440	040514	E.DRV: .ASCII '0 SLAVE '
3673	015006	042526	040		
3674	015011	060	047040	052117	E.NAVA: .ASCIZ '0 NOT AVAILABLE FOR TEST'<CR><LF>
3675	015016	040440	040526	046111	
3676	015024	041101	042514	043040	
3677	015032	051117	052040	051505	
3678	015040	006524	000012		
3679	015044	047516	052040	030115	E.UNIT: .ASCIZ 'NO TM03-TE16/TU77 UNIT FOUND TO TEST'<CR><LF>
3680	015052	026463	042524	033061	
3681	015060	052057	033525	020067	
3682	015066	047125	052111	043040	
3683	015074	052517	042116	052040	
3684	015102	020117	042524	052123	
3685	015110	005015	000		
3686	015113	123	043117	020124	E.SFT: .ASCIZ 'SOFT ERROR (DATA)'<CR><LF>
3687	015120	051105	047522	020122	
3688	015126	042050	052101	024501	
3689	015134	005015	000		
3690	015137	124	051505	020124	E.HDR: .ASCIZ 'TEST # '
3691	015144	020043	000		
3692	015147	040	042504	044526	E.HDR1: .ASCII ' DEVICE ERROR'<CR><LF>
3693	015154	042503	042440	051122	
3694	015162	051117	005015		
3695	015166	051503	004461	041527	.ASCIZ 'CS1'<HT>'WC'<HT>'BA'<HT>'FC'<HT>'CS2'<HT>'DS'<HT>'ER'<HT>'TC'<CR><LF>
3696	015174	041011	004501	041506	
3697	015202	041411	031123	042011	
3698	015210	004523	051105	052011	
3699	015216	006503	000012		
3700	015222	047440	052125	047440	E.HDR2: .ASCIZ ' OUT OF RANGE ERROR'<CR><LF>

3701	015230	020106	040522	043516	
3702	015236	020105	051105	047522	
3703	015244	006522	000012		
3704	015250	005015	044524	042515	E.TIMOV: .ASCIZ <CR><LF>'TIMER OVERFLOWED'<CR><LF>
3705	015256	020122	053117	051105	
3706	015264	046106	053517	042105	
3707	015272	005015	000		
3708	015275	015	052012	046511	E.TIMEX: .ASCIZ <CR><LF>'TIME EXPIRED WAITING FOR RDY'<CR><LF>
3709	015302	020105	054105	044520	
3710	015310	042522	020104	040527	
3711	015316	052111	047111	020107	
3712	015324	047506	020122	042122	
3713	015332	006531	000012		
3714	015336	043440	050101	021440	E.GAP: .ASCIZ ' GAP # '
3715	015344	000040			
3716					
3717					:TIME DOCUMENT LINES
3718	015346	005015	025052	025052	L.HDR1: .ASCIZ <CR><LF>'*****'
3719	015354	025052	025052	025052	
3720	015362	025052	025052	025052	
3721	015370	025052	025052	025052	
3722	015376	025052	025052	025052	
3723	015404	025052	025052	025052	
3724	015412	025052	025052	025052	
3725	015420	025052	025052	025052	
3726	015426	025052	025052	025052	
3727	015434	025052	025052	025052	
3728	015442	025052	025052	025052	
3729	015450	025052	025052	025052	
3730	015456	025052	005015	000	
3731	015463	052	052040	030115	L.HDR2: .ASCII '* TMO3 DRIVE FUNCTION TIMES- DRIVE # '
3732	015470	020063	051104	053111	
3733	015476	020105	052506	041516	
3734	015504	044524	047117	052040	
3735	015512	046511	051505	020055	
3736	015520	051104	053111	020105	
3737	015526	020043			
3738	015530	020060	046123	053101	L.DRV: .ASCII '0 SLAVE # '
3739	015536	020105	020043		
3740	015542	020060	000040		L.SLV: .ASCIZ '0 '
3741	015546	042524	033061	000040	L.TE16: .ASCIZ 'TE16 '
3742	015554	052524	033467	000040	L.TU77: .ASCIZ 'TU77 '
3743	015562	042523	044522	046101	L.SER: .ASCIZ 'SERIAL # '
3744	015570	021440	000040		
3745	015574	006440	025012	005015	L.HDR3: .ASCII ' '<CR><LF>'* '<CR><LF>
3746	015602	020052	052506	041516	.ASCIZ '* FUNCTION'<HT><HT>'TIME (SPECIFICATION)'<HT>'TIME (ACTUAL)'<CR><LF>
3747	015610	044524	047117	004411	
3748	015616	044524	042515	051450	
3749	015624	042520	044503	044506	
3750	015632	040503	044524	047117	
3751	015640	004451	044524	042515	
3752	015646	040450	052103	040525	
3753	015654	024514	005015	000	
3754					
3755	015661	122	047101	042507	L.RNG: .ASCIZ 'RANGE-<'
3756	015666	036075	000		

```

3757 015671 101 052103 040525 L.ACT: .ASCIZ 'ACTUAL-'
3758 015676 036514 000
3759
3760 ;TEST DESCRIPTOR HEADERS
3761 015701 052 020052 047111 A.T000: .ASCIZ '*** INITIALIZATION ERROR'
3762 015706 052111 040511 044514
3763 015714 040532 044524 047117
3764 015722 042440 051122 051117
3765 015730 000
3766 015731 052 053440 044522 A.T001: .ASCIZ '* WRITE FROM BOT'<HT>
3767 015736 042524 043040 047522
3768 015744 020115 047502 004524
3769 015752 000
3770 015753 052 053440 044522 A.T002: .ASCIZ '* WRITE START'<HT><HT>
3771 015760 042524 051440 040524
3772 015766 052122 004411 000
3773 015773 052 053440 044522 A.T003: .ASCIZ '* WRITE SHUTDOWN'<HT>
3774 016000 042524 051440 052510
3775 016006 042124 053517 004516
3776 016014 000
3777 016015 052 053440 044522 A.T004: .ASCIZ '* WRITE SETTLEDOWN'<HT>
3778 016022 042524 051440 052105
3779 016030 046124 042105 053517
3780 016036 004516 000
3781 016041 052 051040 040505 A.T005: .ASCIZ '* READ FROM BOT'<HT><HT>
3782 016046 020104 051106 046517
3783 016054 041040 052117 004411
3784 016062 000
3785 016063 052 051040 040505 A.T006: .ASCIZ '* READ START'<HT><HT>
3786 016070 020104 052123 051101
3787 016076 004524 000011
3788 016102 020052 042522 042101 A.T007: .ASCIZ '* READ SHUTDOWN'<HT><HT>
3789 016110 051440 052510 042124
3790 016116 053517 004516 000011
3791 016124 020052 042522 042101 A.T010: .ASCIZ '* READ SETTLEDOWN'<HT>
3792 016132 051440 052105 046124
3793 016140 042105 053517 004516
3794 016146 000
3795 016147 052 051040 040505 A.T011: .ASCIZ '* READ REV START'<HT>
3796 016154 020104 042522 020126
3797 016162 052123 051101 004524
3798 016170 000
3799 016171 052 051040 040505 A.T012: .ASCIZ '* READ REV SHUTDOWN'<HT>
3800 016176 020104 042522 020126
3801 016204 044123 052125 047504
3802 016212 047127 000011
3803 016216 020052 042522 042101 A.T013: .ASCIZ '* READ REV SETTLEDOWN'<HT>
3804 016224 051040 053105 051440
3805 016232 052105 046124 042105
3806 016240 053517 004516 000
3807 016245 052 052040 051125 A.T014: .ASCIZ '* TURN AROUND DELAY F-R'<HT>
3808 016252 020116 051101 052517
3809 016260 042116 042040 046105
3810 016266 054501 043040 051055
3811 016274 000011
3812 016276 020052 052524 047122 A.T015: .ASCIZ '* TURN AROUND DELAY R-F'<HT>
    
```

```
3813 016304 040440 047522 047125
3814 016312 020104 042504 040514
3815 016320 020131 026522 004506
3816 016326 000
3817 016327 052 043440 050101 A.T016: .ASCIZ '* GAP SIZE-STOP HALF'<HT>
3818 016334 051440 055111 026505
3819 016342 052123 050117 044040
3820 016350 046101 004506 000
3821 016355 052 043440 050101 A.T017: .ASCIZ '* GAP SIZE-START HALF'<HT>
3822 016362 051440 055111 026505
3823 016370 052123 051101 020124
3824 016376 040510 043114 000011
3825 016404 020052 040507 020120 A.T020: .ASCIZ '* GAP SIZE-INTERRECORD'<HT>
3826 016412 044523 042532 044455
3827 016420 052116 051105 042522
3828 016426 047503 042122 000011
3829 016434 020052 040507 020120 A.T021: .ASCIZ '* GAP CONSISTANCY'<HT>
3830 016442 047503 051516 051511
3831 016450 040524 041516 004531
3832 016456 000
3833 016457 052 042040 052101 A.T022: .ASCIZ '* DATA TIME-800BPI'<HT>
3834 016464 020101 044524 042515
3835 016472 034055 030060 050102
3836 016500 004511 000
3837 016503 052 042040 052101 A.T023: .ASCIZ '* DATA TIME-1600BPI'<HT>
3838 016510 020101 044524 042515
3839 016516 030455 030066 041060
3840 016524 044520 000011
3841 016530 020052 051105 051501 A.T024: .ASCIZ '* ERASE GAP TIME'<HT>
3842 016536 020105 040507 020120
3843 016544 044524 042515 000011
3844 016552 020052 051127 052111 A.T025: .ASCIZ '* WRITE FILE MARK'<HT>
3845 016560 020105 044506 042514
3846 016566 046440 051101 004513
3847 016574 000
3848 016575 052 052040 050101 A.T026: .ASCIZ '* TAPE SPEED-FWD'<HT>
3849 016602 020105 050123 042505
3850 016610 026504 053506 004504
3851 016616 000
3852 016617 052 052040 050101 A.T027: .ASCIZ '* TAPE SPEED-REV'<HT>
3853 016624 020105 050123 042505
3854 016632 026504 042522 004526
3855 016640 000
3856
3857 016641 015 057012 000107 L.CNTG: .ASCIZ <CR><LF>'^G'
3858 016646 005015 053523 036522 L.SWR: .ASCIZ <CR><LF>'SWR-'
3859 016654 000
3860 016655 040 047040 053505 L.NEW: .ASCIZ ' NEW= '
3861 016662 020075 000
3862 016665 015 037412 005015 L.QUEST: .ASCIZ <CR><LF>'?'<CR><LF>
3863 016672 000
3864 016674 .EVEN
3865 016674 RDBUF-.
3866 016674 WTBUF-.
3867 016674 000200 .BLKW 128.
3868 000001 .END
```


CZTEECO TMO3-TE16/TU77 DFT
CZTEEC.P11 22-AUG-79 08:40

MACY11 30A(1052) 22-AUG-79 09:00 PAGE 92
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0090

INPUT	1037#	1550	2455	2479	2548	2665										
RESTOR	1034#	1691	1726	1819	1862	1985	2130	2172								
REWIND	1040#	2286	2718	2770	2869	3069	3230	3308	3346	3383	3437	3497	3535	3580		
SAVE	1031#	1669	1708	1802	1829	2105	2143									
SETGO	1050#	2289	2309	2360	2723	2896	2916	2946	2980	2986	3005	3009	3036	3040	3082	
	3088	3107	3111	3118	3136	3142	3160	3164	3170	3189	3193	3197	3202	3235	3245	
	3254	3260	3316	3355	3387	3397	3416	3425	3506	3544	3559					
TIMCHK	1047#	2181	2196	2778	2797	2824	2856	2878	2900	2929	2964	2990	3021	3058	3092	
	3122	3146	3174	3206	3264	3328	3367	3401	3429	3516	3569					
TIMEON	1044#	2193	2773	2819	2851	2873	2895	2924	2960	2985	3017	3054	3087	3117	3141	
	3169	3201	3259	3324	3363	3396	3424	3513	3566							
SCATCH	987#	1234														
SCCHAIN	987#	2434														
SCPREG	987#	1055														
SCPVEC	987#	1082														
STYPE	987#	1561														
.SACT:	987#	1246														
.SEOP	987#	3472														

. ABS. 017274 000

ERRORS DETECTED: 0

DSKZ:CZTEEC.BIN,DSKZ:CZTEEC.SEQ/CRF/SOL=DSKZ:CZTEAC.SML/ML,DSKM:CZTEEC.P11
RUN-TIME: 14 23 2 SECONDS
RUN-TIME RATIO: 236/40-5.8
CORE USED: 8K (15 PAGES)