

TM02/TU16

DRIVE FUNCTION TIMER
MD-11-DZTUD-D

EP-DZTUD-D-DL
COPYRIGHT ©74-77
FICHE 1 OF 1

JAN 1978
digital
MADE IN USA

The main body of the document consists of a large grid of 16 columns and 16 rows of small, illegible text blocks. Each block appears to contain technical data or a small table, but the text is too small to be read. The grid is organized in a regular pattern across the page.

Product Code: MAINDEC-11-DZTUD-D-D
Product Name: TMO2 DRIVE FUNCTION TIMER
Date Created: OCTOBER, 1977
Maintainer: Diagnostic Group
Author: John Adams/R. BARNES/S. CARPENTER

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) 1974, 1977 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

TABLE OF CONTENTS

ABSTRACT

Chapter 1 REQUIREMENTS

- 1.1 EQUIPMENT
- 1.2 MEMORY STORAGE
- 1.3 PRELIMINARY PROGRAMS

Chapter 2 LOADING AND STARTING PROCEDURE

- 2.1 ACT11 OPERATION

Chapter 3 SWITCH SETTINGS

Chapter 4 ERRORS

- 4.1 ERROR TYPEOUT FORMAT (HARDWARE)
- 4.2 ERROR TYPEOUT FORMAT (FUNCTION OUT OF RANGE)

Chapter 5 SUBROUTINE ABSTRACTS

Chapter 6 MISCELLANEOUS

- 6.1 STACK POINTER
- 6.2 EXECUTION TIME

Chapter 7 PROGRAM DESCRIPTION

- 7.1 FUNCTION TIME DOCUMENT
- 7.2 TEST SEQUENCE / RELATED ADJUSTMENTS / ASSOCIATED HARDWARE
- 7.3 TEST DESCRIPTIONS

ABSTRACT

Program DZTUD measures the time required and GAP sizes produced by the TMO2/TU16 Magtape Drive/Slave.

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

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

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

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

CHAPTER 1
REQUIREMENTS

PDP-11 Family Central Processor with 4K memory with up to 64 TM11/TM02 controller/magtape stations

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

1.1 OPTIONAL EQUIPMENT USED

1. none

1.2 STORAGE

Program loads and runs in the first 4K of memory.

1.3 PRELIMINARY PROGRAMS (TO ASSURE HARDWARE OPERATION)

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

CHAPTER 2
LOADING AND STARTING PROCEDURE

The procedure is as follows:

Load program using the Absolute Loader
Load address = 200
Set operating switches
Press start

***IF THE SOFTWARE SWITCH REGISTER IS USED THEN THE PROGRAM WILL TYPE SWR=XXXXXX NEW=
THIS WILL ALLOW LOC. 176 TO BE CHANGED BEFORE THE START OF THE TESTING. (REFER TO CHAPTER 3 FOR OPTIONS)

Program will request DRIVE (TMO2) and SLAVE (TU16) numbers to be tested. Type DRIVE/SLAVE numbers with a comma (,) between each DRIVE/SLAVE to be tested.

REQUESTS FOR TAPE SPEED TESTS AND NRZ ONLY MODE WILL BE MADE.
RESPONSE TO TAPE SPEED ONLY REQUEST WITH A ONE (1) WILL CAUSE
THE PROGRAM TO EXECUTE TEST 31 AND 32 ONLY. THIS IS THE ONLY WAY
TO TEST TAPE SPEED.
NRZ ONLY MODE WILL CAUSE THE PROGRAM TO SKIP THE 1600 BPI DATA TIME TEST.
Type Control U (U) to delete line typed or rubout to delete last character(s).

Program will publish times required and report errors.

2.1 ACT11 OPERATION

WHEN RUN IN ACT11 QV OR AA MODE, THE PROGRAM WILL:

- A) HALT ON ERROR
- B) RUN QV FOR THE FIRST PASS
- C) NOT PRINT A TITLE
- D) SELECT THE DEFAULT CONTROLLER ADDRESS
- E) TEST ALL AVAILABLE DRIVES
- F) NOT RUN SKEW TESTS (31 & 32)
- G) ALLOW NRZ & PE MODE TESTING

CHAPTER 3
SWITCH SETTINGS

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G < G>; THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: SWR=XXXXXXNEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE ''NEW='' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
 - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED)
IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
 - B) IF A CONTROL U < U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

SW15 (100000)	HALT ON ERROR	This switch when set will halt the processor when an error is detected. The PC+2 and PSW at the time of the error is stored on the stack. Pressing continue will cause the error to be typed (if selected) and further testing resumed.
SW14 (040000)	LOOP SUBTEST	This switch when set loops the current subtest regardless of error condition.
SW13 (020000)	INHIBIT ERROR TYPEOUT	This switch when set inhibits error typeout.
SW11 (004000)	INHIBIT SUB- TEST ITERATION	This switch when set causes each subtest to be executed only once. (Initial Startup Only).
SW10 (002000)	INHIBIT FUNCTION TIME PUBLICATION	This switch when set will inhibit the printing of the function times. (See Chapter 7.1)
SW09 (001000)	RING BELL ON ERROR	This switch when set will ring the bell on the TTY when an error is detected.
SW07 (000200)	HALT AFTER SELECTED TEST	This switch when set will cause the program to HALT after the test selected in SW05-SW00 is executed.
SW06 (000100)	CONTINUOUS CYCLE	THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO RUN CONTINUOUSLY UNTIL STOPPED BY THE OPERATOR.
SW5-0	TEST SELECT	THE PROGRAM WILL HALT AFTER EXECUTION OF THE TEST SELECTED WHEN SW07 IS SET.

CHAPTER 4
ERRORS

Two types of errors are detected by this program, hardware errors and incorrect function times.

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

TEST # XXXXXX DEVICE ERROR

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

where:

XXXXXX = Test Number

AAAAAA-111111 = Contents of Tape Register 172440-172454

4.2 ERROR TYPEOUT FORMAT (FUNCTION TIME OUT OF RANGE)

TEST # XXXXXX OUT OF RANGE ERROR

RANGE = <AAAAAA-BBBBBB> ACTUAL = CCCCCC

CHAPTER 5
SUBROUTINE ABSTRACTS

5.1 SCOPE

The SCOPE Routine is called by the scope (EMT) instruction at the start of each subtest. The Scope routine performs the following functions:

1. Loads R5 with base address
2. Types Time Line <SW08>
3. Provides continuous loop <SW14>
4. Moves function time into table
5. Outputs Line Item if selected
6. Provides HALT on test <SW07>
7. Delays 350MS before starting test
8. Init's Drive/Slave
9. Clears the error flag (ERFLG)

The routine monitors SW14, SW11, SW10, SW03, and SW07.

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

5.2 PUBLISH

The Publish Routine is called from the Scope Routine if SW10 is equal to 0 (Publish Time Document). The routine will print a "SINGLE LINE ITEM" each time it is called.

5.3 .HLT

The HLT Routine is called by the HLT (Trap) instruction when an error is detected. A HLT (TRAP) instruction formats the error information as shown in Sec 4.1. A HLT+1 (TRAP+1) formats the ERROR AS SHOWN IN SEC 4.2.

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

CHAPTER 6
MISCELLANEOUS

6.1 STACK POINTER

The Stack Pointer is initially set to 500 and is reset to 500 by the SCOPEA Routine.

6.2 EXECUTION TIME

When SW11=1 (Inhibit Iterations) the time required is 2 min.

When SW11=0 (Iterate Subtests) the time required is 9 min.

CHAPTER 7
PROGRAM DESCRIPTION

7.1 SAMPLE TIME DOCUMENT

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

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

* FUNCTION	TIME(SPECIFICATION)	TIME(ACTUAL)
* WRITE FROM BOT	RANGE=<188000-184000>	ACTUAL=184740
* WRITE START	RANGE=<009500-008700>	ACTUAL=009120
* WRITE SHUTDOWN	RANGE=<008900-008500>	ACTUAL=008840
* WRITE SETTLEDOWN	RANGE=<013500-008100>	ACTUAL=010970
* READ FROM BOT	RANGE=<152000-149000>	ACTUAL=150580
* READ START	RANGE=<003200-002600>	ACTUAL=002740
* READ SHUTDOWN	RANGE=<004650-004250>	ACTUAL=004360
* READ SETTLEDOWN	RANGE=<013500-008100>	ACTUAL=010970
* READ REV START	RANGE=<003200-002600>	ACTUAL=002740
* READ REV SHUTDOWN	RANGE=<003700-003300>	ACTUAL=003520
* READ REV SETTLEDOWN	RANGE=<013500-008100>	ACTUAL=010970
* TURN AROUND DELAY F-R	RANGE=<016700-010700>	ACTUAL=013600
* TURN AROUND DELAY R-F	RANGE=<016700-010700>	ACTUAL=013660
* GAP SIZE-STOP HALF	RANGE=<012900-009500>	ACTUAL=012200
* GAP SIZE-START HALF	RANGE=<011800-008500>	ACTUAL=010520
* GAP SIZE-INTERRECORD	RANGE=<014800-013700>	ACTUAL=014500
* GAP CONSISANCY	RANGE=<014000-012400>	ACTUAL=013040
* DATA TIME-200 BPI	RANGE=<024100-023100>	ACTUAL=023460
* DATA TIME-556 BPI	RANGE=<024000-023000>	ACTUAL=023350
* DATA TIME-800BPI	RANGE=<024000-023000>	ACTUAL=023400
* DATA TIME-1600BPI	RANGE=<025100-024100>	ACTUAL=024470
* ERASE GAP TIME	RANGE=<101000-099000>	ACTUAL=099510
* WRITE FILE MARK	RANGE=<105000-103000>	ACTUAL=103990

7.1.1 SAMPLE TIME DOCUMENT FOR TAPE SPEED TESTS

TYPE FIRST ADDRESS OF CONTROLLER 172440

TYPE TMO2 DRIVE #'S TO BE TESTED 0

FOR TMO2 DRIVE 0- TYPE SLAVE #'S TO BE TESTED 7

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

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

*TM

*FUNCTION

*TAPE SPEED FWD

*TAPE SPEED REV

TIME(SPECIFICATION)

RANGE=<022700-021700>

RANGE=<022700-021700>

TIME(ACTUAL)

ACTUAL=022500

ACTUAL=022500

7.2 TEST SEQUENCE WITH RELATED ADJUSTMENTS AND ASSOCIATED HARDWARE

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

21. GAP CONSISTENCY	*SAME AS IN TEST 16	*WRITE CLOCK
*TEST NUMBER 22 IS RESERVED FOR FUTURE USE		
23. DATA TIME 200 BPI	*NONE	* " "
24. DATA TIME 556 BPI	* "	* " "
25. DATA TIME 800 BPI	* "	* " "
26. DATA TIME 1600 BPI	* "	* " "
27. ERASE GAP TIME	* "	*M8911 ROM*M8903 ACCL CNTR
30. WRITE FILE MARK	* "	* " " * " " "
31. TAPE SPEED-FORWARD	*FWD SPEED	*CAPSTAN SERVO LOOP
32. TAPE SPEED-REVERSE	*REVERSE SPEED	*CAPSTAN SERVO LOOP

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

7.3 TEST DESCRIPTIONS:

THE FIRST THIRTEEN (13) TESTS (T1 - T15) ARE CHECKS OF THE ROM CIRCUITS IN THE TU16 (M9811), THE ACCL COUNTER IN THE TMO2 (M8903), AND THE SETTLEDOWN ONE SHOT (M8910).

T1. WRITE FROM BOT:

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

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

T2. WRITE START:

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

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

T3. WRITE SHUTDOWN:

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

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

T15. TURN AROUND DELAY-REVERSE TO FORWARD

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

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

GAP MEASUREMENTS:

THE PREVIOUS THIRTEEN (13) TESTS WERE MEASUREMENTS OF LOGIC DELAYS PERFORMED BY THE TMO2 OR TU16 IN ORDER TO ALLOW FOR PROPER ACCELERATION AND DECELERATION OF TAPE ACCORDING TO THE DESIRED INTERCORD GAP (.6 INCHES). THIS TEST, HOWEVER, WILL MEASURE THE PHYSICAL SIZE OF THE INTERCORD GAP THAT EXISTS ON TAPE AS A RESULT OF THE START/STOP TIMES OF THE CAPSTAN ITSELF. BECAUSE THE INTERCORD GAP IS CREATED BY TWO ACTIONS, THE START OF MOTION AND THE STOP OF MOTION IT IS NECESSARY TO MAKE TWO SEPERATE MEASUREMENTS. A THIRD MEASUREMENT, MADE ON THE FLY, OF THE ENTIRE LENGTH OF THE GAP WILL ALSO BE MADE.

T16. GAP SIZE (STOP HALF)

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

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

T17. GAP SIZE (START HALF)

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

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

T20. GAP SIZE (INTERRECORD)

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

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

T21. GAP CONSISTENCY:

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

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

(SEE GTIMTBL IN DZTUD LISTING FOR GAP TIMES)

T22. RESERVED FOR FUTURE USE*****

T23. DATA TIME AT 200 BPI:

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

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

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

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

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

T27. ERASE:

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

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

T30. TAPE MARK:

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

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

T31. TAPE SPEED FORWARD:

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

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

T32. TAPE SPEED REVERSE:

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

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

1939	STARTING INSTRUCTIONS
1965	MACRO DEFINITIONS
1989	REGISTER ASSIGNMENTS
2018	TMO2/TU16 REGISTER BITS
2210	TIME SPECIFICATION TABLE
2247	GAP TIME SPECIFICATION TABLE
2271	TEST HEADER POINTERS
2371	PROGRAM SUBROUTINES
2372	TYPE SUBROUTINE
2404	OCTAL TO ASCII & TYPE ROUTINE
2441	OCTAL TO DECIMAL & TYPE ROUTINE
2474	TYPE SPECIFIED TIMES ROUTINE
2505	TYPE GAP TIMES SUBROUTINE
2536	ASCII TO OCTAL CONVERT SUBROUTINE
2558	PUBLISH SUBROUTINE
2599	INPUT SUBROUTINE
2645	ERROR SERVICE ROUTINES
2714	SCOPE SUBROUTINE
2776	TIMER SUBROUTINES
2904	DELAY SUBROUTINES
2929	DIVIDE SUBROUTINE
2965	DRIVE SUBROUTINES
3143	PROGRAM INITIALIZATION
3414	START OF TESTS
4275	PROGRAM MESSAGES

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

```
.MACRO $LCTRL
;; LISTING CONTROL
.NLIST SEQ,MD,MC,CND
.LIST ME
.ENABL ABS
.ENDM $LCTRL
.MACRO $STINST
;; LOADING AND STARTING PROCEEDURE
;; LOAD PROGRAM USING THE ABS LOADER
;; LOAD ADDRESS 200
;; SET SWITCH REGISTER OPTIONS
;; START
;; STACK POINTER IS SET AT 1100
.ENDM $STINST
.MACRO $CPREG
;; DEFINITIONS AND REGISTER ASSIGNMENTS
;; GENERAL REGISTER ASSIGNMENTS
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
R10=%0
R11=%1
R12=%2
R13=%3
R14=%4
R15=%5
;; REGISTER ADDRESSES
PSW= 177776
SLR= 177774
PIRQ= 177772
UBREAK= 177770
TKS= 177560
TKB= 177562
TPS= 177564
TPB= 177566
;; PROCESSOR STATUS WORD
;; STACK LIMIT REGISTER (11/40,11/45)
;; PROGRAM INTERRUPT REQ. (11/45)
;; MICRO-BREAK REGISTER (11/45)
;; KEYBOARD CSR
;; KEYBOARD DATA BUFFER REGISTER
;; TELEPRINTER CSR
;; TELEPRINTER DATA BUFFER REGISTER
.ENDM $CPREG
.MACRO $FPREGS
;; FLOATING POINT REGISTERS
AC0=%0
AC1=%1
AC2=%2
AC3=%3
AC4=%4
AC5=%5
.ENDM $FPREGS
.MACRO $SWOPT
;; SWITCH REGISTER SWITCH ASSIGNMENTS
```

```

57 SW15= 100000 ;;HALT ON ERROR SWITCH
58 SW14= 040000 ;;LOOP SUBTEST SWITCH
59 SW13= 020000 ;;INHIBIT ITERATIONS SWITCH
60 SW12= 010000 ;;INHIBIT PASS PARAMETERS SWITCH
61 SW11= 004000 ;;INHIBIT ERROR TYPEOUT SWITCH
62 SW10= 002000 ;;RING BELL ON ERROR SWITCH
63 SW9= 001000 ;;LOOP ON ERROR SWITCH
64
65 .ENDM $SWOPT
66 .MACRO REGBOX NAME, ADDRESS, B15, B14, B13, B12, B11, B10, B09, B08, B07, B06, B05, B04, B03
67 ;;
68 ;;
69 ;;
70 ;;
71 ;;
72 ;;
73 ;;
74 ;;
75
76
77
78
79
80 .ENDM REGBOX
81 .MACRO $PSWBITS
82 ;;STATUS REGISTER (PSW) BIT ASSIGNMENTS
83 C=1 ;;C BIT
84 V=2 ;;V BIT
85 Z=4 ;;Z BIT
86 N=10 ;;N BIT
87 T=20 ;;'T' BIT
88 PRTY7=340 ;;PRIORITY LEVEL 7
89 PRTY6=300 ;;PRIORITY LEVEL 6
90 PRTY5=240 ;;PRIORITY LEVEL 5
91 PRTY4=200 ;;PRIORITY LEVEL 4
92 KM=000000 ;;KERNEL MODE
93 SM=040000 ;;SUPERVISORY MODE
94 UM=140000 ;;USER MODE
95 PKM=000000 ;;PREVIOUS KERNEL MODE
96 PSM=010000 ;;PREVIOUS SUPERVISORY MODE
97 PUM=030000 ;;PREVIOUS USER MODE
98 REG=004000 ;;SELECT R10-R15
99
100 .ENDM $PSWBITS
101 .MACRO $CPVEC
102 ;;VECTOR ADDRESSES
103 ERRVEC=4 ;;ADDRESS OF ERROR VECTOR
104 RESVEC=10 ;;ADDRESS OF RESERVED INST. TRAP VECTOR
105 TBITVEC=14 ;;ADDRESS OF 'T' BIT TRAP VECTOR
106 TRTVEC=14 ;;ADDRESS OF 'TRACE' TRAP VECTOR
107 BPTVEC=14 ;;ADDRESS OF 'BREAKPOINT' TRAP VECTOR
108 IOTVEC=20 ;;ADDRESS OF IOT TRAP VECTOR
109 PFVEC=24 ;;ADDRESS OF POWER FAIL TRAP VECTOR
110 EMTVEC=30 ;;ADDRESS OF EMT VECTOR
111 TRAPVEC=34 ;;ADDRESS OF TRAP VECTOR
112 TKVEC= 60 ;;ADDRESS OF TTY KEYBOARD INT. VECTOR
  
```

```
113 TPVEC=64 ;: ADDRESS OF TTY PRINTER INTERRUPT VECTOR
114 PARVEC= 114 ;: ADDRESS OF MA/MF PARITY ERROR VECTOR
115 PIRVEC=240 ;: ADDRESS OF PIRQ VECTOR
116 FPEVEC=244 ;: ADDRESS OF FLOATING POINT INT. VECTOR
117 MMVEC=250 ;: ADDRESS OF MEM MGMT ERROR TRAP VECTOR
118
119 . ENDM $CPVEC
120 . MACRO $LPREGS
121 LPS= 177514 ;: ADDRESS OF LINE PRINTER STATUS REG
122 LPB= 177516 ;: AND PRINTER BUFFER REGISTER
123 LPVEC= 200 ;: INTERRUPT VECTOR
124 LPLVL= 200 ;: PRIORITY LEVEL 4
125 . ENDM $LPREGS
126 . MACRO $STCREGS
127 TCST= 177340 ;: ADDRESS OF DECTAPE STATUS REG
128 TCCM= 177342 ;: ADDRESS OF DECTAPE COMMAND REG
129 TCWC= 177344 ;: ADDRESS OF DECTAPE WORD COUNT REG
130 TCBA= 177346 ;: ADDRESS OF DECTAPE BUS ADDRESS REG
131 TCDT= 177350 ;: ADDRESS OF DECTAPE DATA REGISTER
132 TCVEC= 214 ;: ADDRESS OF DECTAPE INTERRUPT VECTOR
133 TCLVL= 240 ;: AND INTERRUPT LEVEL (5)
134
135 . ENDM $STCREGS
136 . MACRO $STKPTR A
137 ;: INITIAL STACK POINTER SETTING
138 STKPTR= A ;: INITIAL STACK POINTER
139 . ENDM $STKPTR
140 . MACRO $RFREGS
141 RFDCS= 177460 ;: ADDRESS OF DCS REGISTER
142 RFWC= 177462 ;: ADDRESS OF WC REGISTER
143 RFCMA= 177464 ;: ADDRESS OF CMA REGISTER
144 RFDAR= 177466 ;: ADDRESS OF DAR REGISTER
145 RFDAE= 177470 ;: ADDRESS OF DAE REGISTER
146 RFDOB= 177472 ;: ADDRESS OF DATA BUFFER REGISTER
147 RFMA= 177474 ;: ADDRESS OF MA REGISTER
148 RFADS= 177476 ;: ADDRESS OF ADS REGISTER
149 RFVEC= 204 ;: INTERRUPT VECTOR ADDRESS
150 RFLVL= 240 ;: AND PRIORITY LEVEL
151 . ENDM $RFREGS
152 . MACRO $RP4REG
153 RPCS1= 172000 ;: ADDRESS OF CS1 REGISTER
154 RPWC= 172002 ;: ADDRESS OF WORD COUNT REGISTER
155 RPBA= 172004 ;: ADDRESS OF BUS ADDRESS REGISTER
156 RPDST= 172006 ;: ADDRESS OF DESIRED SECTOR/TRACK REG.
157 RPCS2= 172010 ;: ADDRESS OF CS2 REGISTER
158 RPDS1= 172012 ;: ADDRESS OF DS1 REGISTER
159 RPER1= 172014 ;: ADDRESS OF ER1 REGISTER
160 RPAS= 172016 ;: ADDRESS OF ATTENTION SUMMARY REG.
161 RPLA= 172020 ;: ADDRESS OF LOOK AHEAD REGISTER
162 RPDB= 172022 ;: ADDRESS OF DATA BUFFER REGISTER
163 RPMR= 172024 ;: ADDRESS OF MAIN. REGISTER
164 RPDT= 172026 ;: ADDRESS OF DRIVE TYPE REGISTER
165 RPSN= 172030 ;: ADDRESS OF SERIAL # REGISTER
166 RHOF= 172032 ;: ADDRESS OF FORMAT REGISTER
167 RPCA= 172034 ;: ADDRESS OF DESIRED CYL. ADDRESS REG.
168 RPCC= 172036 ;: ADDRESS OF CURRENT CYLINDER ADDRESS REG
```

```
169 RPER2= 172040 ;; ADDRESS OF ER2 REGISTER
170 RPER3= 172042 ;; ADDRESS OF ER3 REGISTER
171 RPEC1= 172044 ;; ADDRESS OF ECC POSITION REGISTER
172 RPEC2= 172046 ;; ADDRESS OF ECC PATTERN REGISTER
173 . ENDM $RP4REG
174 . MACRO $RSREGS
175 RSCS1= 172040 ;; ADDRESS OF CONTROL REGISTER
176 RSWC= 172042 ;; ADDRESS OF WORD COUNT REGISTER
177 RSBA= 172044 ;; ADDRESS OF BUS ADDRESS REGISTER
178 RSDA= 172046 ;; ADDRESS OF DISK ADDRESS REGISTER
179 DSCS2= 172050 ;; ADDRESS OF CS2 REGISTER
180 RSDS= 172052 ;; ADDRESS OF DRIVE STATUS REGISTER
181 RSER= 172054 ;; ADDRESS OF ERROR REGISTER
182 RSAS= 172056 ;; ADDRESS OF ATTENTION SUMMARY REGISTER
183 RSLA= 172060 ;; ADDRESS OF LOOK AHEAD REGISTER
184 RSDB= 172062 ;; ADDRESS OF DATA BUFFER REGISTER
185 RSMR= 172064 ;; ADDRESS OF MAINTENANCE REGISTER
186 RSDT= 172066 ;; ADDRESS OF DRIVE TYPE REGISTER
187 RSVEC= 204 ;; INTERRUPT VECTOR ADDRESS
188 RSLVL= 240 ;; PRIORITY LEVEL ON INTERRUPT
189 . ENDM $RSREGS
190 . MACRO $KWPREGS
191
192 KWPCSR= 172540
193 KWPCSB= 172542
194 KWPCR= 172544
195 KWVEC= 0
196 KWPLVL= 300
197 . ENDM $KWPREGS
198
199 . MACRO $RKREGS
200 RKDS= 177400 ;; DRIVE STATUS REGISTER
201 RKER= 177402 ;; ERROR REGISTER
202 RKCS= 177404 ;; CONTROL STATUS REGISTER
203 RKWC= 177406 ;; WORD COUNT REGISTER
204 RKBA= 177410 ;; CURRENT BUS ADDRESS REGISTER
205 RKDA= 177412 ;; DISK REGISTER REGISTER
206 RKMR= 177414 ;; MAINTENANCE REGISTER
207 RKDB= 177416 ;; DATA BUFFER REGISTER
208 RKVEC= 000220 ;; ADDRESS OF INTERRUPT VECTOR
209 RKLVL= 000240 ;; PRIORITY LEVEL (5)
210
211 . ENDM $RKREGS
212 . MACRO $RPREGS
213 RPDS= 176710 ;; ADDRESS OF DRIVE STATUS REGISTER
214 RPER= 176712 ;; ERROR REGISTER
215 RPCS= 176714 ;; CONTROL STATUS REGISTER
216 RPWC= 176716 ;; WORD COUNT REGISTER
217 RPBA= 176720 ;; BUS ADDRESS REGISTER
218 RPCA= 176722 ;; CYLINDER ADDRESS REGISTER
219 RPDA= 176724 ;; DISK ADDRESS REGISTER
220 RPVEC= 254 ;; VECTOR ADDRESS
221 RPLVL= 240 ;; PRIORITY LEVEL (6)
222
223 . ENDM $RPREGS
224 . MACRO $RHREGS
```

```
225 ;;RH11 REGISTERS
226 RHDCS=172040 ;; ADDRESS OF DISK CONTROL & STATUS REG
227 RHCMA=172042 ;; ADDRESS OF WORD COUNT REGISTER
228 RHDAR=172046 ;; ADDRESS OF CURRENT MEMORY ADDR REG
229 RHDAR=172046 ;; ADDRESS OF DISK ADDRESS REGISTER
230 RHDAR=172046 ;; ADDRESS OF DISK ADDRESS REGISTER
231 RHDAR=172046 ;; ADDRESS OF DISK ADDRESS EXTENSION
232 RHDAR=172046 ;; AND ERROR REGISTER.
233 RHDAR=172046 ;; ADDRESS OF DISK DATA BUFFER REGISTER
234 RHDAR=172046 ;; ADDRESS OF MAINTENANCE REGISTER
235 RHDAR=172046 ;; ADDRESS OF DISK SECTOR REGISTER
236 RHDAR=172046 ;; ADDRESS OF RH11 INTERRUPT VECTOR
237 RHDAR=172046 ;; AND PRIORITY LEVEL (6)
238
239 . ENDM $RHREGS
240 . MACRO $RHBITS
241 ;;RH11 DISK CONTROL & STATUS REGISTER (RHDCS)
242 GO=1 ;; GO
243 NOOP=0 ;; NO-OPERATION
244 WRITE=2 ;; WRITE
245 READ=4 ;; READ
246 WCHK=6 ;; WRITE CHECK
247 XXXX=10
248 A16=20 ;; EXTENDED ADDRESS BIT 16
249 A17=40 ;; EXTENDED ADDRESS BIT 17
250 IE=100 ;; INTERRUPT ENABLE
251 RDY=200 ;; READY
252 RHCLR=400 ;; CLEAR
253 MA=1000 ;; MAINTENANCE MODE
254 WLO=2000 ;; WRITE LOCKOUT ERROR
255 DOF=4000 ;; DISK OVERFLOW
256 MXF=10000 ;; MISSED TRANSFER ERROR
257 WCE=20000 ;; WRITE CHECK ERROR
258 DER=40000 ;; DATA ERROR
259 ERR=100000 ;; ERROR
260 ;;DISK ADDRESS EXTENSION & ERROR REGISTER (RHDAE)
261 DRL=200 ;; DATA REQUEST LATE
262 CMAINH=400 ;; CMA INCREMENT INHIBIT
263 PWR=2000 ;; POWER FAULT
264 ILLACC=40000 ;; ILLEGAL ACCESS
265 CLKERR=100000 ;; CLOCK ERROR
266
267 ;;MAINTENANCE REGISTER (RHMA)
268 MC=1 ;; MAINTENANCE CLOCK
269 MW=2 ;; MAINTENANCE WINDOW
270 MI=4 ;; MAINTENANCE INDEX
271 MRD=10 ;; MAIN READ DATA
272 SEP=20 ;; SECTOR PULSE
273 ADDCON=40 ;; ADDRESS CONFIRMED
274 LSR=100 ;; LOAD SHIFT REGISTER
275 WR=200 ;; WRITE
276 RD=400 ;; READ
277 MWD=1000 ;; MAINTENANCE WRITE DATA
278 LWI=2000 ;; LAST WORD + 1
279 STR=4000 ;; STROBE BUFFER
280 RWCLK=10000 ;; READ OR WRITE CLOCK
```

281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336

```
.ENDM $RHBITS
.MACRO $MTREGS
;; TM11 MAGNETIC TAPE REGISTERS
MTS= 172520
MTC= 172522
MTBRC= 172524
MTCMA= 172526
MTD= 172530
MTRD= 172532
MTVEC= 224 ;; INTERRUPT VECTOR
MTLVL= 240 ;; LEVEL 5
.ENDM $MTREGS
.MACRO $MMBITS
;; MEM MGMT REGISTER SSRO BIT ASSIGNMENTS
ENMM=1 ;; ENABLE MEMORY MANAGEMENT
VPO=0 ;; 'VIRTUAL' PAGE 0
VP1=2 ;; ETC
VP2=4
VP3=6
VP4=10
VP5=12
VP6=14
VP7=16
DS=20 ;; 'D' SPACE PAGE
IS=00 ;; 'I' SPACE PAGE
UPG=140 ;; USER PAGE
SPG=40 ;; SUPERVISOR PAGE
KPG=000 ;; KERNEL PAGE
IC=200 ;; INSTRUCTION COMPLETE
DM=400 ;; DESTINATION MODE
TE=1000 ;; TRAP ENABLE
OSTF=4000 ;; OST ABORT FLAG
MMTF=10000 ;; MEMORY MANAGEMENT TRAP FLAG
AVA=20000 ;; ACCESS VIOLATION ABORT
PLA=40000 ;; PAGE LENGTH ABORT FLAG
NRA=100000 ;; NON-RESIDENT ABORT

;; SR1 BIT ASSIGNMENTS
S1=10
S2=20
S4=40
S6=60
S8=100
SM1=370
SM2=360
SM4=340
SM6=320
SM8=300
D0=0
D1=4000
D2=10000
DM1=174000
DM2=170000
DRO=000
DR1=400
```

337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392

DR2=1000
DR3=1400
DR4=2000
DR5=2400
DR6=3000
DR7=3400

::SR3 BIT ASSIGNMENTS

UDE=1
SDE=2
KDE=4

::USER 'D' SPACE ENABLE
::SUPERVISOR 'D' SPACE ENABLE
::KERNEL 'D' SPACE ENABLE

.ENDM \$MMBITS
.MACRO \$MMREGS

::MEMORY MANAGEMENT REGISTER ADDRESS ASSIGNMENTS

SR0=177572
SR1=177574
SR2=177576
SR3=172516

::ADDRESS OF MEM MGMT REGISTER SR0
:: " " " " SR1
:: " " " " SR2
::ADDRESS OF MEM MGMT REGISTER SR3

.ENDM \$MMREGS

.MACRO \$UMMREGS

UIPDR0=177600
UIPDR1=177602
UIPDR2=177604
UIPDR3=177606
UIPDR4=177610
UIPDR5=177612
UIPDR6=177614
UIPDR7=177616

::ADDRESS OF USER 'I' PAGE DESCRIPTOR
::REGISTERS

UDPDR0=177620
UDPDR1=177622
UDPDR2=177624
UDPDR3=177626
UDPDR4=177630
UDPDR5=177632
UDPDR6=177634
UDPDR7=177636

::ADDRESS OF USER 'D' PAGE DESCRIPTOR
::REGISTERS

UIPAR0=177640
UIPAR1=177642
UIPAR2=177644
UIPAR3=177646
UIPAR4=177650
UIPAR5=177652
UIPAR6=177654
UIPAR7=177656

::ADDRESS OF USER 'I' PAGE ADDRESS
::REGISTERS

UDPAR0=177660
UDPAR1=177662
UDPAR2=177664
UDPAR3=177666
UDPAR4=177670
UDPAR5=177672

::ADDRESS OF USER 'D' PAGE ADDRESS
::REGISTERS


```
393      UDPAR6=177674
394      UDPAR7=177676
395      . ENDM    $SUMMREGS
396      . MACRO  $$SUMMREGS
397      SIPDR0=172200      ;; ADDRESS OF SUPERVISOR 'I' PAGE
398      SIPDR1=172202      ;; DESCRIPTOR REGISTERS
399      SIPDR2=172204
400      SIPDR3=172206
401      SIPDR4=172210
402      SIPDR5=172212
403      SIPDR6=172214
404      SIPDR7=172216
405
406      SDPDR0=172220      ;; ADDRESS OF SUPERVISOR 'D' PAGE
407      SDPDR1=172222      ;; DESCRIPTOR REGISTERS
408      SDPDR2=172224
409      SDPDR3=172226
410      SDPDR4=172230
411      SDPDR5=172232
412      SDPDR6=172234
413      SDPDR7=172236
414
415      SIPAR0=172240      ;; ADDRESS OF SUPERVISOR 'I' PAGE
416      SIPAR1=172242      ;; ADDRESS REGISTERS
417      SIPAR2=172244
418      SIPAR3=172246
419      SIPAR4=172250
420      SIPAR5=172252
421      SIPAR6=172254
422      SIPAR7=172256
423
424      SDPAR0=172260      ;; ADDRESS OF SUPERVISOR 'D' PAGE
425      SDPAR1=172262      ;; ADDRESS REGISTERS
426      SDPAR2=172264
427      SDPAR3=172266
428      SDPAR4=172270
429      SDPAR5=172272
430      SDPAR6=172274
431      SDPAR7=172276
432
433      . ENDM    $$SUMMREGS
434      . MACRO  $$KMMREGS
435      KIPDR0=172300      ;; ADDRESS OF KERNEL 'I' PAGE
436      KIPDR1=172302      ;; DESCRIPTOR REGISTERS
437      KIPDR2=172304
438      KIPDR3=172306
439      KIPDR4=172310
440      KIPDR5=172312
441      KIPDR6=172314
442      KIPDR7=172316
443
444      KOPDR0=172320      ;; ADDRESSES OF KERNEL 'D' PAGE
445      KOPDR1=172322      ;; DESCRIPTOR REGISTERS
446      KOPDR2=172324
447      KOPDR3=172326
448      KOPDR4=172330
```

449 KDPDR5=172332
450 KDPDR6=172334
451 KDPDR7=172336
452
453 KIPAR0=172340 ;: ADDRESSES OF KERNEL 'I' PAGE
454 KIPAR1=172342 ;: ADDRESS REGISTERS
455 KIPAR2=172344
456 KIPAR3=172346
457 KIPAR4=172350
458 KIPAR5=172352
459 KIPAR6=172354
460 KIPAR7=172356

461
462 KDPAR0=172360 ;: ADDRESSES OF KERNEL 'D' PAGE
463 KDPAR1=172362 ;: ADDRESS REGISTERS
464 KDPAR2=172364
465 KDPAR3=172366
466 KDPAR4=172370
467 KDPAR5=172372
468 KDPAR6=172374
469 KDPAR7=172376

470
471 . ENDM \$KMMREGS
472 . MACRO \$PDRBITS
473 ;: ACCESS CONTROL FIELD DEFINITIONS (IN PDR)
474 NRO=0 ;: NON-RESIDENT ABORT ALL REFS.
475 ROOT=1 ;: TRAP ON READ, ABORT ON WRITE
476 RDO=2 ;: READ, ABORT ON WRITE
477 NR3=3 ;: UNUSED ABORT ALL
478 RWT=4 ;: TRAP ON READ & WRITE
479 RWTW=5 ;: READ, TRAP ON WRITE
480 RW=6 ;: READ & WRITE
481 NR7=7 ;: ABORT ALL
482 ED=10 ;: EXPANSION DIRECTION BIT
483 UP=00 ;: EXPAND UP
484 DWN=10 ;: EXPAND DOWN
485 W=100 ;: 'W' BIT
486 A=200 ;: 'A' BIT

487 . ENDM \$PDRBITS
488
489 . MACRO \$STRAPS
490
491 ;: THIS ROUTINE DECODES A TRAP INSTRUCTION AND JUMPS TO THE APPROPRIATE
492 ;: SUBROUTINE.
493 TRAPS: MOV (SP), -(SP)
494 SUB #2, (SP) ;: FORM ADDRESS OF TRAP INSTRUCTION
495 MOV @ (SP), (SP) ;: GET TRAP INSTRUCTION
496 TSTB (SP)
497 BMI 1\$
498 ADD #TRPTAB-TRAP, (SP)
499 MOV @ (SP)+, PC ;: 'JUMP' TO APPROPRIATE ROUTINE
500 1\$: JMP \$GET

501 ;: TABLE OF ROUTINES CALLED BY TRAP INSTRUCTIONS
502 TRPTAB:
503 . NLIST
504

```

505 . QQ. =0
506 . LIST
507
508 . MACRO SET A, B1
509 . IF B <B1>
510 $SET A, 'A, TRAP+. QQ. , . QQ.
511 . IFF
512 $SET A, B1, <TRAP+. QQ. >, . QQ.
513 . ENDC
514 . NLIST
515 . QQ. =. QQ. +2
516 . LIST
517 . ENDM SET
518 . MACRO $SET A, B, C, D
519 . NLIST
520 A=C
521 . LIST
522 . WORD B ; A=TRAP+D (C)
523 . ENDM $SET
524 . ENDM STRAPS
525
526 . MACRO $SCOPE1
527 ;; SCOPE (EMT) SERVICE ROUTINE
528 ;; THIS ROUTINE STORES PC OF LAST SUBTEST SUCCESSFULLY COMPLETED,
529 ;; IN %1 AND RESETS THE STACK PTR (TO INITIAL SETTING)
530 SCOPE: MOV (SP), R1 ;; SAVE LAST PC IN R1
531 MOV #STKPTR, SP ;; SET INITIAL STACK POINTER
532
533 BIT #400, @SWR ;; LOAD PDP11/45 MICRO BREAK REG?
534 BEQ +10
535 MOVB @SWR, @#UBREAK ;; LOAD MICRO BREAK REG WITH SRO-7
536 JMP (R1) ;; GO START NEXT SUB TEST
537
538 . ENDM $SCOPE1
539 . MACRO $SCOPE3 A
540 ;; SCOPE (EMT) SERVICE ROUTINE
541 ;; THIS ROUTINE ALLOWS A SUBTEST TO BE ITERATED 'A' TIMES BEFORE
542 ;; BEGINNING THE NEXT SUBTEST. ROUTINE ALSO RESETS STACK POINTER AND
543 ;; STORES PC OF LAST TEST COMPLETED IN R1
544 SCOPE: BIT #BIT14, @SWR ;; CONTINUOUSLY LOOP TEST?
545 BEQ SCOPEC
546 SCOPEB: MOV #STKPTR, SP ;; SET INITIAL STACK POINTER
547 BIT #400, @SWR ;; LOAD PDP11/45 MICRO BREAK REG?
548 BEQ +10
549 MOVB @SWR, @#UBREAK ;; LOAD MICRO BREAK REG WITH SRO-7
550 JMP (R1) ;; RETURN
551 SCOPEC: DEC (PC)+ ;; DECREMENT ITERATION COUNT
552 SCOPED: 'A' ;; CONTAINS SUB TEST ITERATION COUNT
553 BNE SCOPEB ;;
554 MOV #'A', SCOPED ;; RESET SUBTEST ITERATION COUNT
555 MOV (SP), R1 ;; GET ADDRESS OF NEXT TEST
556 BR SCOPEB
557 . ENDM $SCOPE3
558 . MACRO $SCOPE A
559 ;; SCOPE (TRAP) SERVICE ROUTINE
560 ;; THIS ROUTINE ALLOWS THE SUBTEST TO BE CONTINUOUSLY LOOPED, ITERATED

```

```

561      ;; (OR NOT ITERATED) BEFORE BEGINNING NEXT SUBTEST
562      .SCOPE: NOP
563      NOP
564      NOP
565      TST      @#ERRFLG      ;; CHECK IF ERROR FLAG IS SET
566      BEQ      1$
567      BIT      #SW9, @SWR      ;; CHECK IF LOOP ON ERROR
568      BEQ      1$
569      MOV      @#EPC, (SP)      ;; LOAD ERROR RETURN ADDRESS
570      RTI      ;; AND EXIT
571      1$: CLR      @#ERRFLG      ;; CLEAR ERROR FLAG
572      BIT      #SW14, @SWR      ;; CHECK BIT 14 (CONTINUOUS LOOP)
573      BEQ      4$
574      2$: MOV      @#SCPBLK+4, (SP) ;; GET START OF TEST PC
575      3$: RTI
576      4$: BIT      #SW13, @SWR      ;; CHECK INHIBIT ITERATION SWITCH
577      BEQ      5$
578      5$: MOV      (SP), @#SCPBLK+4 ;; RESET TEST PC
579      MOV      (SP), @#EPC      ;; SET RETURN FROM ERROR PC
580      RTI      ;; RETURN TO FOLLOWING SUBTEST
581      6$: CMP      #A, @#SCPBLK+2 ;; CHECK ITERATION COUNT
582      BNE      7$
583      MOV      (SP), -(SP)
584      MOV      @ (SP), (SP)      ;; GET SCOPE CALL INSTRUCTION
585      MOV      (SP)+, @#SCPBLK+2 ;; SET SUBTEST ITERATION COUNT
586      BNE      7$      ;; DEFAULT TO 'A' ITERATIONS IF 0
587      MOV      #A, @#SCPBLK+2
588      7$: DEC      @#SCPBLK+2      ;; DECREMENT SUBTEST ITERATION COUNT
589      BNE      2$
590      MOV      #A, #SCPBLK+2      ;; SET SUBTEST ITERATION COUNT
591      BR      5$
592      .ENDM      $SCOPE
593      .MACRO      $POWER
594      ;; POWER FAIL SUBROUTINE
595      .POWER: MOV      #PUP, @#PFVEC      ;; SET POWER FAIL VECTOR TO PUP
596      HALT
597
598      ;; POWER UP SERVICE ROUTINE
599      PUP: MOV      #STKPTR, SP      ;; SET STACK PTR
600      MOV      #.POWER, @#PFVEC      ;; RESET POWER FAIL VECTOR TO POWER
601      ;; DOWN ROUTINE
602      CLR      (PC)+      ;; KILL TIME WAITING FOR TTY MOTOR
603      1$: .WORD      0
604      2$: INC      1$
605      BNE      2$
606      TYPE      PFAIL
607      TYPE      HRS
608      TYPE      SCRLF
609      JMP      @#START
610      PFAIL: .ASCIZ  <15><12>'POWER FAILED AT '
611      .EVEN
612
613      .ENDM      $POWER
614
615
616

```

```
617 . MACRO $DONE
618 ;; THIS ROUTINE IS ENTERED AT THE COMPLETION OF THE TEST
619 . DONE: CLR @#PSW ;; CLEAR STATUS WORD
620 MOV #STKPTR, SP ;; SET STACK PTR
621 RESET
622 TYPE
623 BELL
624 MOV @#42, R0 ;; GET MONITOR HOOK FOR ACT11/DDP
625 BEQ DONE1 ;; BRANCH IF (42)=0
626 LOGICAL: JSR PC, (R0) ;; RETURN TO ACT11/DDP VIA 42
627 DONE1: NOP
628 NOP
629 NOP
630 JMP @#START ;; GO TO START OF TEST
631 . ENDM $DONE
632
633 . MACRO $SAVE
634 ;; ROUTINE TO SAVE REGISTERS ON THE STACK
635 . SAVE: MOV %5, -(SP) ;; R5 IS SAVED AT 12(SP)
636 MOV %4, -(SP) ;; R4 IS SAVED AT 10(SP)
637 MOV %3, -(SP) ;; R3 IS SAVED AT 6(SP)
638 MOV %2, -(SP) ;; R2 IS SAVED AT 4(SP)
639 MOV %1, -(SP) ;; R1 IS SAVED AT 2(SP)
640 MOV %0, -(SP) ;; R0 IS SAVED AT (SP)
641 MOV 16(SP), -(SP) ;; PUSH RETURN PSW ON THE STACK
642 MOV 16(SP), -(SP) ;; PUSH RETURN PC ON THE STACK
643 RTI
644
645 . ENDM $SAVE
646 . MACRO $RESTORE
647 ;; ROUTINE TO RESTORE REGISTERS SAVED ON THE STACK
648 . RESTORE: MOV (SP)+, 16(SP)
649 MOV (SP)+, 16(SP)
650 MOV (SP)+, %0
651 MOV (SP)+, %1
652 MOV (SP)+, %2
653 MOV (SP)+, %3
654 MOV (SP)+, %4
655 MOV (SP)+, %5
656 RTI ;; RETURN
657
658 . ENDM $RESTORE
659
660 . MACRO $SHIFT
661 ;; ROUTINE TO SHIFT DATA ON THE STACK
662 ;; CALL: MOV DATA, -(SP) ;; PUSH DATA ONTO STACK
663 ;; SHIFT
664 ;; SHIFT COUNT ;; SHIFT COUNT +/- = LEFT/RIGHT
665 . SHIFT: MOV @#(SP), -(SP) ;; CHECK SHIFT COUNT + OR -
666 BMI 2$ ;; BRANCH IF - (RIGHT)
667 1$: ASL 6(SP) ;; SHIFT DATA LEFT
668 DEC (SP) ;; DECREMENT SHIFT COUNT
669 BNE 1$
670 BR 3$
671 2$: ASR 6(SP) ;; SHIFT DATA RIGHT
672 INC (SP) ;; INCREMENT SHIFT COUNT
```

```

673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728

35:   BNE      25
      TST     (SP)+
      ADD     #2, (SP)
      RTI

      .ENDM   $SHIFT
      .MACRO  $SETTBIT
      MOV     @#PSW, -(SP)
      BIS     #T, (SP)
      MOV     #.+6, -(SP)
      RTI
      .ENDM   $SETTBIT

      .MACRO  LDPDR   ACF, ED, PLF, PDR
      MOV     #'PLF'*256. -400+'ED'+ 'ACF', @#'PDR'
      .ENDM   LDPDR
      .MACRO  $TYPE
;; ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;; THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;; CALL: TYPE
;; MESADR
;; MESADR IS FIRST ADDRESS OF ASCIZ STRING

;; TAGS USED BY THE TYPE ROUTINE BELOW
      $HT=11
SNULL: .BYTE 0
SFILL: .BYTE 2
STPFLG: .BYTE 0
STKFLG: .BYTE 0
STPS: .WORD 177564
STPB: .WORD 177566
SCHARCNT: .BYTE 0
SCNTRLO: .BYTE 0
SCRLF: .ASCIZ <15><12>
      .EVEN

TYPE:  MOV     R0, -(SP)
      MOV     @2(SP), R0
      ADD     #2, 2(SP)
      CLRB   $CNTRLO

TYPE1: TSTB   $CNTRLO
      BEQ    TYPE2
TCRLF: TYPE, SCRLF
      TSTB   RDSW
      BPL   TYPE3
      CLR   RDSW
      RTS   PC

TYPE2: MOVB   (R0)+, -(SP)
      BNE   TYPE4
      TST   (SP)+
TYPE3: MOV     (SP)+, R0
      RTI

TYPE4: CMPB   #$HT, (SP)
      BEQ   9$
  
```

```

729      JSR      PC,55      ;;TYPE CHARACTER
730      35:     CMPB     #12,(SP)+  ;;CHECK IF CHARACTER WAS A LINE FEED
731      BNE      TYPE1     ;;BRANCH IF NOT LINE FEED
732      MOV      $NULL,-(SP)  ;;GET # OF FILLERS REQUIRED AND FILLER
733      ;;CHARACTER.
734
735      45:     DECB     1(SP)     ;;DECREMENT FILLERS REQ. COUNT
736      BLT      35         ;;BRANCH IF NO MORE FILLERS ARE REQUIRED
737      JSR      PC,55      ;;TYPE FILLER CHARACTER
738      BR       45
739
740      55:     TSTB     @STPS     ;;WAIT FOR OUTPUT DEVICE
741      BPL      -4
742      CMPB     #17,@#$CNTRL0  ;;CHECK IF CONTROL 0 WAS TYPED
743      BEQ      65         ;;STOP TYPING MESSAGE IF 0 WAS TYPED
744      MOVB     2(SP),@STPB    ;;OUTPUT CHARACTER
745      65:     CMPB     #15,2(SP)  ;;BRANCH IF NOT <CR>
746      BNE      75
747      CLRB     $CHARCNT     ;;CLEAR CHARACTERS TYPED COUNT
748      BR       85
749      75:     CMPB     #12,2(SP)  ;;BRANCH IF <LF> OR 'NULL'
750      BGE      85
751      INCB     $CHARCNT     ;;INCREMENT CHARACTER TYPED COUNT
752      85:     RTS      PC
753
754      ;;HORIZONTAL TAB <HT> PROCESSER
755      95:     MOVB     #40,(SP)   ;;LOAD 'SPACE'
756      105:    JSR      PC,55     ;;TYPE 'SPACE'
757      BITB     #7,$CHARCNT     ;;TYPE SPACES UNTIL A MULTIPLE
758      BNE     105         ;;OF 8 CHARACTERS HAVE BEEN TYPED
759      TSTB     (SP)+         ;;POP SPACE
760      BR       TYPE1       ;;GET NEXT CHARACTER
761      .ENDM     $TYPE
762      .MACRO    $TYPEF
763      ;;ROUTINE TO TYPE MESSAGE FOLLOWING TYPE CALL. CALL:
764      ;; TYPEF      ;;CALL
765      ;; BR       15      ;;BRANCH AROUND ASCIZ MESSAGE
766      ;; .ASCIZ 'MESSAGE'
767      ;; .EVEN
768
769      .TYPEF:   MOV      (SP),15
770      ADD      #2,15      ;;FORM MESSAGE ADDRESS
771      TYPE
772      15:     .WORD    0      ;;CONTAINS MESSAGE ADDRESS
773      RTI
774      .ENDM     $TYPEF
775      .MACRO    TYPEC
776      JSR      PC,TYPEC
777      .ENDM     TYPEC
778      .MACR     FORMAT
779      JSR      PC,FORMAT  ;;GO TO FORMAT ROUTINE
780      .ENDM     FORMAT
781      .MACRO    $SCNV16
782
783      ;;ROUTINE TO CONVERT 16 BIT OCTAL TO ASCII.
784      ;;CALL: MOV      DATA,R2      ;;PUT DATA TO BE CONVERTED IN R2

```

```
785      ;;      CNV16
786      ;; LEAVES CONVERTED ASCII IN TABLE BEGINNING AT 'DIGITS'
787
788      .CNV16: SAVE
789      MOV      #DIGITS,R4      ;; ADDRESS WHERE ASCII VALUES ARE STORED
790      CLR      R3              ;; WORKING & INDEX REGISTER
791      MOV      R2,R1          ;; SAVE
792      15:     ASL      R2              ;; FIRST DIGIT TO R3
793      ROL      R3
794      MOV      #6,R0          ;; DIGIT COUNT
795      BR       35             ;; TYPE FIRST DIGIT
796      25:     ASL      R2
797      ROL      R3
798      DEC      R1
799      BNE     25
800      35:     MOV      #3,R1          ;; DIGIT SHIFT COUNT
801      MOVB    DIGTAB(3),(4)+      ;; LOAD DIGIT INTO MESSAGE
802      CLR      R3              ;; CLEAR INDEX
803      DEC      R0              ;; DEC DIGIT COUNT
804      BNE     25
805      RESTORE
806      RTI                   ;; RETURN TO CALLER
807
808      .ENDM    %CNV16
809
810      .MACRO  %CNV18
```

```
811
812      ;; ROUTINE TO CONVERT 16 BIT OCTAL VIRTUAL ADDRESS TO 18 BIT ASCII
813      ;; PHYSICAL ADDRESS.
814      ;; CALL: MOV      ADDRESS+2,R1
815      ;;      CNV18
816      ;; LEAVES CONVERTED ASCII IN TABLE BEGINNING AT 'DIGITS'
```

```
817
818      .CNV18: SAVE
819      MOV      #DIGITS,R4      ;; GET ADDRESS OF DIGIT TABLE
820      CLR      R3
821      SUB      #2,R1          ;; ADJUST ADDRESS
822      MOV      R1,R5          ;; SAVE
823      MOV      R5,R2
824      TST     MMAVA          ;; CHECK IF MEM MGMT IS AVAILABLE
825      BEQ     15             ;; BRANCH IF NOT AVAILABLE
826      BIC     #17777,R2      ;; CLEAR ALL BUT PAR SELECTOR BITS
827      ASL     R2              ;; SHIFT BITS 15-13
828      ROL     R2              ;; LEFT TO
829      ROL     R2              ;; 3-1
830      ROL     R2
831      ASL     R2
832      MOV     @PARTAB(2),R1    ;; GET CONTENTS OF PAR REGISTER
833      BIT     #UM,@ERRPSW     ;; CHECK IF WAS IN USER MODE
834      BEQ     105            ;; BRANCH IF NOT IN USER MODE
835      MOV     @UPARTAB(2),R1  ;; GET CONTENTS OF USER PAR REG
836      105:   MOV     #6,R0      ;; SET SHIFT COUNTER
837      ASL     R1              ;; SHIFT PAR BITS IN R1<11-0>
838      ROL     R3              ;; TO R3<1-0> & R1<15-6>
839      SOB     R0,-4
840      BIC     #16000,R5      ;; CLEAR ADDRESS PAR SELECTOR BITS
```



```

841          ADD    R5,R1          ;;FORM 18 BIT PHYSICAL ADDRESS
842          ADC    R3              ;;IN R1 & R3
843    15:     ASL    R1              ;;SHIFT 18/16 BIT ADDRESS
844          ROL    R3              ;;1 PLACE LEFT
845          MOV    #6,R0          ;;SET DIGIT COUNTER
846          BR     35             ;;GO FORM ASCII VALUE OF MSD
847
848    25:     ASL    R1              ;;SHIFT NEXT DIGIT
849          ROL    R3              ;;3 PLACES LEFT INTO R3
850          DEC    R5
851          BNE    25
852
853    35:     MOV    #3,R5          ;;SET DIGIT SHIFT COUNT
854          MOVB   DIGTAB(R3),(R4)+ ;;LOAD DIGIT INTO DIGIT TABLE
855          CLR    R3              ;;CLEAR INDEX & LAST DIGIT
856          DEC    R0              ;;DECREMENT DIGIT COUNT
857          BNE    25             ;;LOOP UNTIL 6 DIGITS DONE
858          RESTORE                ;;RESTORE REGISTERS
859          RTI                    ;;RETURN TO CALLER
860
861    PARTAB: KIPAR0
862            KIPAR1
863            KIPAR2
864            KIPAR3
865            KIPAR4
866            KIPAR5
867            KIPAR6
868    UPARTAB: UIPAR0
869             UIPAR1
870             UIPAR2
871             UIPAR3
872             UIPAR4
873             UIPAR5
874             UIPAR6
875             UIPAR7
876
877          . ENDM   SCN48
878
879          . MACRO  SCN48
880          ;;ROUTINE TO CONVERT 48. OCTAL DIGITS TO ASCII. LEAVES CONVERTED DATA IN
881          ;;TABLE STARTING AT DIGBUF. CALL:
882          ;;      MOV    ADDRESS,XFERS          ;;LOAD ADDRESS OF OCTAL DIGITS
883          ;;      CNV48
884          . CNV48: SAVE
885          MOV    #9,R1          ;;CLEAR DIGIT BUFFER
886          MOV    #DIGBUF+18,R0
887    15:     CLR    -(R0)
888          DEC    R1
889          BNE    15
890
891          MOV    XFERS,R5          ;;GET ADDRESS CONTAINING XFER COUNT
892          MOV    #XFERS+2,R4
893          MOV    (R5)+,(R4)+      ;;MOVE XFER COUNT TO BUFFER AT XFERS+2
894          MOV    (R5)+,(R4)+      ;;THREE WORDS
895          MOV    (R5)+,(R4)+
896          MOV    #16,R1          ;;GET OCTAL DIGIT COUNT
  
```

```

897      25:      MOV      #3,R2          ;; GET BINARY BITS/OCTAL DIGIT COUNT
898      35:      MOV      #XFERS+2,R4    ;; GET ADDRESS OF OCTAL DATA
899      CMP      (R4)+,(R4)+          ;; ADD 4 (POINT TO LSD)
900      ASL      (R4)                  ;; SHIFT 3 BITS
901      ROL      -(R4)
902      ROL      -(R4)
903      ROLB     (R0)                  ;; SHIFT BIT INTO DIGIT BUFFER
904      DEC      R2                    ;; DECREMENT BIT COUNT
905      BNE      35
906      BISB     #260,(R0)+           ;; FORM ASCII VALUE
907      DEC      R1                    ;; DECREMENT OCTAL DIGIT COUNT
908      BNE      25
909      RESTORE
910      RTI                          ;; RETURN TO CALLER
911
912      ;; DIGIT BUFFER
913      XFERS:   .WORD    0             ;; CONTAINS ADDRESS OF XFER COUNT
914      .WORD    0             ;; CONTAINS DEV XFER COUNT <47-32>
915      .WORD    0             ;; CONTAINS DEV XFER COUNT <31-16>
916      .WORD    0             ;; CONTAINS DEV XFER COUNT <15-00>
917
918      DIGBUF:  .BLKW   9
919      .ENDM    $CNV48
920
921      .MACRO   PSPTAGS
922      ;; LIST OF EQUATES FOR GET ROUTINE
923      VEC=    0             ;; COSTANT FOR GETTING A VECTOR ADDRESS
924      UNIT=   1             ;; CONSTANT FOR GETTING A UNIT #
925      WC=     2             ;; CONSTANT FOR GETTING WORD COUNT
926      WBA=    3             ;; CONSTANT FOR GETTING WRITE BUFFER ADDRESS
927      EA=     =4           ;; CONSTANT FOR GETTING EA BITS
928      RBA=    5             ;; CONSTANT FOR GETTING READ BUFFER ADDRESS
929      PAT=    6             ;; CONSTANT FOR GETTING PATTERN #
930      ADR=    7             ;; CONSTANT FOR GETTING
931      WRTLOC= 10            ;; CONSTANT FOR GETTING WRITE LOC
932      CONST=  11            ;; CONSTANT FOR GETTING CONSTANT
933      CYL=    12            ;; CONSTANT FOR GETTING CYLINDER #
934      SURF=   13            ;; CONSTANT FOR GETTING SURFACE #
935      SECT=   14            ;; CONSTANT FOR GETTING SECTOR #
936      CPLOC=  15            ;; CONSTANT FOR GETTING STARTING ADDRESS
937      ;; OF CP TESTS
938      DEVNAM= 16            ;; CONSTANT FOR GETTING OCTAL ID OF DEVICE
939
940      .ENDM    PSPTAGS
941
942      .MACRO   GETANS
943      JSR      PC, GETANS          ;; GO GET USER RESPONSE
944      .ENDM    GETANS
945
946      .MACRO   $GETANS
947      ;; ROUTINE TO GET USER 1 CHAR RESPONSE TO PROGRAM REQUEST
948      ;; CALLED BY GETANS MACRO OR JSR PC, GETANS
949      GETANS:  NOP
950      MOV      (SP), -(SP)          ;; PUSH RETURN PC DOWN ON STACK
951      CLR      2(SP)                ;; CLEAR ANSWER LOCATION
952      15:      TSTB     @#TKS        ;; WAIT FOR RESPONSE
  
```

```

953          BPL      1$
954          CMP      #215, @#TKB      ;; CHECK CARRIAGE RETURN
955          BNE      2$              ;; BRANCH IF NOT CARRIAGE RETURN
956          TYPE
957          $CRLF
958          RTS      PC              ;; EXIT BACK TO CALLER
959          2$:      CMP      #377, @#TPB  ;; CHECK RUBOUT
960          BNE      3$              ;; BRANCH IF NOT RUBOUT
961          MOV      #' , @#TPB      ;; ECHO BACK SLASH ( )
962          BR       1$              ;; AND WAIT FOR NEXT RESPONSE
963          3$:      MOV      @#TKB, @#TPB  ;; ECHO TYPED CHAR.
964          MOV      @#TKB, 2(SP)      ;; PUT CHARACTER ON THE STACK
965          BIC      #200, 2(SP)      ;; STRIP PARITY BIT
966          BR       1$              ;; AND WAIT FOR NEXT RESPONSE
967
968          .ENDM    $GETANS
969
970          .MACRO  $RECO
971          ;; ROUTINE TO GET TYPED ASCII DATA AND CONVERT TO OCTAL.
972          ;; LEAVES THE OCTAL DATA IN THE ADDRESS FOLLOWING THE CALL, (BITS 0-15);;
973          ;; AND BITS 16-17 IN ADDRESS 46.
974          ;; CALL: RECO                ;; (A TRAP TYPE INST.)
975          ;; .WORD 0                    ;; CONTAINS RETURNED DATA
976          ;; CONTINUE                  ;; RETURNS HERE
977          .RECO:  CLR      -(SP)
978          CLR      @#46
979          1$:      TSTB     @#. TKS      ;; WAIT FOR USER RESPONSE
980          BPL      -4
981          MOVB     @#. TKB, -(SP)      ;; GET ASCII CHARACTER
982          CLR      @#. TKS            ;; CLEAR 'READY'
983          CMPB     #177, (SP)         ;; CHECK IF RUBOUT
984          BNE      2$              ;; BRANCH IF NOT RUBOUT
985          MOVB     #' , 4$
986          TYPE
987          4$
988          TST      (SP)+              ;; ECHO BACK SLASH
989          ROR      @#46              ;; POP LAST TYPED CHAR OFF THE STACK
990          ROR      (SP)              ;; SHIFT LAST TYPED CHARACTER OUT
991          ASR      @#46
992          ROR      (SP)
993          ASR      @#46
994          ROR      (SP)
995          BR       1$              ;; AND WAIT FOR NEXT CHAR.
996          2$:      CMPB     #25, (SP)   ;; CHECK IF CONTROL U
997          BNE      20$              ;; BRANCH IF NOT CONTROL U
998          CMP      (SP)+, (SP)+      ;; POP CNTRL U CHAR & PREV DATA OFF STACK
999

```

DZTUD-D TMO2 DRIVE FUNCTION TIMER
DZTUDD.SML 21-OCT-77 09:54

MACY11 30(1046) 27-OCT-77 ^{E 4} 13:42 PAGE 8

SEQ 0043

1007

TYPE

```

1009          SCRLF
1010          BR          .RECO          ;;GO GET INPUT DATA
1011
1012          20$:      CMPB      #15,(SP)      ;;CHECK IF CARRIAGE RETURN
1013          BNE          3$              ;;BRANCH IF NOT A CARRIAGE RETURN
1014          TYPE
1015          SCRLF
1016          TST          (SP)+          ;;POP CARRIAGE RETURN OFF THE STACK
1017          MOV          (SP)+, @ (SP)      ;;PUT DATA IN ADDRESS FOLLOWING CALL
1018          BIC          #177774, @#46      ;;CLEAR UNUSED BITS
1019          ADD          #2, (SP)
1020          RTI
1021
1022          3$:      MOVB      (SP), 4$
1023          TYPE
1024          4$
1025          BIC          #177770, (SP)      ;;ECHO CHARACTER
1026          ASL          2(SP)          ;;SAVE ONLY 3 MSBS
1027          ROL          @#46          ;;SHIFT LAST CHARACTER OVER
1028          ASL          2(SP)          ;;THREE PLACES
1029          ROL          @#46
1030          ASL          2(SP)
1031          ROL          @#46
1032          BIS          (SP)+, (SP)      ;;INSET OCTAL CHAR
1033          BR          1$              ;;GO WAIT FOR NEXT CHAR.
1034          4$:      .WORD      0          ;;CONTAINS CHARACTER TO BE TYPED
1035
1036          .ENDM      $RECO
1037
1038          .MACRO      SAVLDR
1039          JSR          PC, .SAVLDR      ;;GO SAVE LOADER
1040          .ENDM      SAVLDR
1041          .MACRO      $$SAVLDR
1042          ;;ROUTINE TO SAVE ABS LOADER
1043          ;;CALLED BY SAVLDR MACRO OR JSR PC, SAVLDR
1044          .SAVLDR: NOP
1045          MOV          SP, R1          ;;GET LOADER'S STACK PTR
1046          ADD          #4, R1          ;;FIRST ADDRESS OF LOADER
1047          MOV          R1, $LDR1      ;;SAVE FOR RESTORE ROUTINE
1048          MOV          #140, R2      ;;WORD COUNT
1049          MOV          $LDR1+2, R3    ;;WHERE LOADER IS TO BE STORED
1050          1$:      MOV          (R1)+, (R3)+ ;;STORE LOADER
1051          DEC          R2
1052          BNE          1$
1053          RTS          PC          ;;RETURN
1054
1055          .ENDM      $$SAVLDR
1056          .MACRO      RESLDR
1057          JSR          PC, .RESLDR      ;;RESTORE ABSOLUTE LOADER
1058          .ENDM      RESLDR
1059          .MACRO      $RESLDR
1060          ;;ROUTINE TO RESTORE LOADER
1061          ;;CALLED BY RESLDR MACRO OR JSR PC, RESLDR
1062          .RESLDR: NOP
1063          MOV          $LDR1, R5      ;;GET FIRST ADDRESS OF WHERE LOADER IS
1064          ;;TO BE RESTORED

```

1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120

```

MOV #SLDR1+2,R4 ;; ADDRESS WHERE LOADER IS STORED
MOV #140,R2 ;; WORD COUNT
15: MOV (R4)+,(R5)+ ;; RESTORE
DEC R2
BNE 15
TYPE
SLDRM
HALT

SLDR1: 0 ;; FIRST ADDRESS
      = +300 ;; SAVE 300 LOCATIONS
SLDRM: .ASCIZ 'LOADER IS RESTORED'<15><12>
      .EVEN
      .ENDM $RESLDR

      .MACRO $PCCHK

;; THE BELOW ROUTINE ASCERTAINS WHICH CP & CP OPTIONS THE PROGRAM IS RUN-
;; NING ON AND SETS AN INDICATOR IN OPT.CP ACCORDINGLY.
$PCCHK: MOV #RT1,@#ERRVEC+2 ;; SET UP ERROR TRAP TO RETURN
MOV #3,R0
SEC
TST @#PIRQ ;; RO=3 IF 11/45
SBC R0 ;; RO=2 IF 11/40
SEC
TSTB @#PSW+1 ;; RO=1 IF 11/20
SBC R0
CLR @#177700 ;; RO=0 IF 11/05
ASL R0 ;; SHIFT INDICATOR
MOV R0,(PC)+ ;; SET CP INDICATOR
OPT.CP: .WORD 0 ;; CONTAINS OPTION & CP INDICATORS
      ;; EVEN BYTE: 0=11/05, 2=11/20, 4=11/40, 6=11/45
      ;; ODD BYTE: 200=MEM MGMT, 100=EIS
SEC
TST @#SRO ;; CHECK IF MEM MGMT IS AVAILABLE
BCS 15
BISB #200,OPT.CP+1 ;; SET MEM MGMT AVAIL INDICATOR
15: MOV #RT1,@#RESVEC+2 ;; SET RESERVED INSTRUCTION VECTOR
CLR R2
SEC
ASH R2,R2 ;; WILL TRAP IF 11/40 WITHOUT EIS
BCS 25 ;; BRANCH IF NO EIS AVAILABLE
BIS #EISOPT,OPT.CP ;; SET EIS AVAIL INDICATOR
25: CLR @#ERRVEC+2 ;; RESTORE ERROR TRAP TO HALT ON TRAP
CLR @#RESVEC+2

      .ENDM $PCCHK
      .MACRO $KW11

;; ROUTINE TO START EITHER THE KW11-L, OR THE KW11-P CLOCK
;; START KW11-L IF AVAILABLE
$KW11: MOV #RT1,@#ERRVEC+2 ;; SET ERROR TRAP VECTOR
MOV #60,(PC)+ ;; SET TICK COUNTER
45: .WORD 0 ;; CONTAINS CLOCK TICK COUNT
SEV ;; SET V BIT IN PSW
BIT #100,@#LKS ;; CHECK IF KW11-L IS AVAILABLE & IF

```

```

1121                                     ;; IT IS ENABLED
1122                                     ;; GO TO KW11-P START IF NOT AVAILABLE
1123     BVS      1$                       ;; BRANCH IF PREVIOUSLY ENABLED
1124     BNE      5$                       ;; SET INTERRUPT VECTOR
1125     MOV      #2$, @#LKVEC             ;; PRIORITY LEVEL 6 ON INTERRUPT
1126     MOV      #300, @#LKVEC+2         ;; ENABLE INTERRUPT
1127     BIS      #100, @#LKS             ;; GO TO END OF ROUTINE
1128     BR       5$
1129
1130     ;; KW11-P
1131     1$: SEV      ;; SET 'V' BIT IN PSW
1132     BIT      #100, @#KWPCSR          ;; CHECK IF KW11-P IS AVAILABLE &
1133     BVS      5$                       ;; IF IT IS ENABLED
1134     BNE      5$
1135     MOV      #2$, @#KWPVEC           ;; SET INTERRUPT VECTOR
1136     MOV      #300, @#KWPVEC+2       ;; SET PRIORITY LEVEL = 6 ON INTERRUPT
1137     MOV      #114, @#KWPCSR        ;; SET IE, LINE FREQ, AND MODE 1
1138     BR       5$
1139     ;; LINE CLOCK INTERRUPT SERVICE ROUTINE
1140     2$: NOP
1141     MOV      R0, -(SP)                ;; SAVE R0 ON THE STACK
1142     MOV      R2, -(SP)                ;; SAVE R2 ON THE STACK
1143     DEC      4$                       ;; DECREMENT 1 SECOND COUNTER
1144     BNE      3$
1145     MOV      #SECS+1, R0              ;; PUT ASCII VALUE FOR 0 IN R2
1146     MOV      #'0, R2
1147     MOV      #60, 4$                 ;; RESET COUNT
1148     INCB    (R0)                     ;; INCREMENT LSD SECONDS COUNT
1149     CMPB    #72, (R0)                 ;; LSD = 9?
1150     BNE      3$
1151     MOVB    R2, (R0)                  ;; RESET LSD = 0
1152     INCB    -(R0)                     ;; INCREMENT MSD SECONDS COUNT
1153     CMPB    #66, (R0)                 ;; MSD = 6?
1154     BNE      3$
1155     MOVB    R2, (R0) ;; RESET MSD = 0
1156     CMPB    -(R0), -(R0)
1157     INCB    -(R0)                     ;; INCREMENT LSD MINUTES COUNT
1158     CMPB    #72, (R0)                 ;; LSD = 9?
1159     BNE      3$
1160     MOVB    R2, (R0)                  ;; RESET LSD = 0
1161     INCB    -(R0)                     ;; INCREMENT MSD MINUTES COUNT
1162     CMPB    #66, (R0)                 ;; MSD = 6?
1163     BNE      3$
1164     MOVB    R2, (R0) ;; RESET MSD = 0
1165     CMPB    -(R0), -(R0)
1166     INCB    -(R0)                     ;; INCREMENT LSD HOURS
1167     CMPB    #64, (R0)                 ;; LSD = 4?
1168     BNE      3$
1169     MOVB    R2, (R0)                  ;; RESET LSD = 0
1170     INCB    -(R0)                     ;; INCREMENT MSD HOURS
1171     CMPB    #63, (R0)                 ;; MSD HOURS = 3?
1172     BNE      3$
1173     MOVB    R2, (R0) ;; RESET MSD HOURS TO 0
1174     3$: MOV      (SP)+, R2            ;; RESTORE R2
1175     MOV      (SP)+, R0                ;; RESTORE R0
1176     RTI                               ;; RETURN

```

1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232

```
SS: CLR @#ERRVEC+2
      .ENDM $KW11
      .MACRO $KWPDRV
;; ROUTINE TO START KW11-P REAL TIME CLOCK. CALL:
;; JSR PC, .KWPDRV
;; CALLER MUST LOAD FOLLOWING PARAMETERS INTO A TABLE STRUCTURED AS SHOWN BELOW:
;; .KWPCTR: .WORD 0 ;; COUNTER VALUE
;; .KWPCSB: .WORD 0 ;; BUFFER
;; .KWPRET: .WORD 0 ;; RETURN PC WHEN DONE
;; .KWCSR: .WORD 0 ;; CONTOLT STATUS REGISTER
;; AND LEAVE RO POINTING TO .KWPCTR

      $KWPREGS

.KWPDRV: NOP
      MOV R1, -(SP) ;; SAVE R1 ON THE STACK
      MOV .KWPCSR, R1 ;; GET ADDRESS OF CSR
      ADD #6, R1 ;; FORM ADDRESS OF KWPCTR
      MOV (RO)+, -(R1) ;; LOAD CTR
      MOV (RO)+, -(R1) ;; LOAD CSB
      MOV (SP)+, R1 ;; RESTORE R1
      MOV (RO)+, (PC)+ ;; LOAD RETURN PC ON INTERRUPT

.KWPRET: .WORD 0
      MOV (RO), @ (PC)+ ;; LOAD COMMAND

.KWPCSR: .WORD 0
      RTS PC

KWPINT: NOP
      JSR PC, @.KWPRET
      RTI

      .ENDM $KWPDRV
      .MACRO $LPDRV

;; ROUTINE TO PRINT ON LP11 LINE PRINTER. CALL:
;; JSR PC, .LPDRV
;; CALLER MUST LOAD THE FOLLOWING LP PARAMETERS INTO A TABLE STRUCTURED
;; AS SHOWN BELOW:
;; .LPBA: .WORD 0 ;; CONTAINS FIRST ADDRESS OF DATA TO BE
;; PRINTED
;; .LPWC: .WORD 0 ;; CONTAINS # OF CHARACTERS / LINE
;; .LPRET: .WORD 0 ;; CONTAINS RETURN ADDRESS WHEN DONE

;; AND LEAVE RO POINTING TO THE FIRST ADDRESS OF THE TABLE (LPBA)

      $LPREGS

.LPDRV: NOP
      MOV .LPS, .LPB ;; GET ADDRESS OF LP STATUS REG
      ADD #2, .LPB ;; FORM ADDRESS OF LP DATA BUFFER REG
      MOV (RO)+, LPADR ;; GET FIRST ADDRESS
      MOV (RO)+, LPCNT ;; GET # OF CHARACTERS
      MOV (RO)+, (PC)+ ;; GET RETURN ADDRESS

.LPRET: .WORD 0
```



```
1233          BIS      #100, @.LPS      ;;SET INTERRUPT ENABLE BIT
1234          RTS      PC                ;;RETURN
1235
1236 LPINT:      NOP
1237          TST      @.LPS              ;;CHECK FOR ERROR
1238          BPL      2$                 ;;BRANCH IF NO ERROR
1239          JSR      PC, @.LPRET         ;;RETURN TO CALLER
1240          1$:      MOVB @LPADR, @.LPB  ;;LOAD CHAR INTO BUFFER
1241          2$:      INC      LPADR       ;;STEP ADDRESS
1242          TSTB    @.LPS                ;;CHECK IF DONE
1243          BMI      2$                 ;;BRANCH IF NOT DONE
1244          RTI
1245 LPADR:      .WORD  0                  ;;CONTAINS ADDRESS OF FIRST DATA
1246 LPCNT:     .WORD  0                  ;;CONTAINS # OF CHARACTERS / LINE
1247 .LPB:      .WORD  0                  ;;CONTAINS ADDRESS OF LP BUFFER REG.
1248          .ENDM      SLPDRV
```

1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305

```

      MACRO $TCDRV
;; ROUTINE TO READ/WRITE ON DECTAPE. CALL:
;; JSR PC,TCDRV
;; CALLER MUST LOAD THE FOLLOWING TC PARAMETERS INTO A TABLE STRUCTURED
;; AS SHOWN BELOW:
;; TCDT: .WORD 0 ;; BLOCK # TO BE READ/WRITTEN
;; TCBA: .WORD 0 ;; MEMORY TRANSFER ADDRESS
;; TCWC: .WORD 0 ;; # OF WORDS TO TRANSFER
;; TCCM: .WORD 0 ;; UNIT # AND COMMAND
;; TCRET: .WORD 0 ;; RETURN ADDRESS WHEN DONE
;; AND LEAVE R0 POINTING TO FIRST ADDRESS OF TABLE (.TCDT)

      STCREGS

.TCDRV: NOP
      MOV R1,-(SP) ;; SAVE R1 ON THE STACK
      MOV .TCDS,R1 ;; GET ADDRESS OF TCDS REGISTER
      ADD #10,R1 ;; FORM ADDRESS OF TCDT REGISTER
      MOV (R0)+,(R1) ;; LOAD BLOCK #
      MOV (R1),(PC)+ ;; SAVE BLOCK #

.TCDTR: .WORD 0 ;; CONTAINS BLOCK # FOR REVERSE SEARCH
      SUB #3,.TCDTR ;; FORM BLOCK # FOR REVERSE SEARCH
      MOV (R0)+,-(R1) ;; LOAD MEMORY TRANSFER ADDRESS
      MOV (R0)+,-(R1) ;; LOAD WORD COUNT
      MOV (R0),(PC)+ ;; SAVE COMMAND

.TCCMD: .WORD 0 ;; CONTAINS COMMAND TO EXECUTE WHEN BLOCK
      ;; IS FOUND
      MOV (R0)+,(PC)+ ;; FORM SEARCH FORWARD COMMAND
.SRCHF: .WORD 0 ;; CONTAINS SEARCH FWD COMMAND
      MOVB #103,.SRCHF
      MOV .SRCHF,(PC)+ ;; FORM SEARCH REVERSE COMMAND
.SRCHR: .WORD 0 ;; CONTAINS SEARCH REVERSE COMMAND
      BIS #4000,.SRCHR
      MOV .SRCHF,(PC)+ ;; SAVE COMMAND

.TCCOM: .WORD 0 ;; CONTAINS COMMAND ISSUED TO TC
      MOV (R0)+,(PC)+ ;; LOAD RETURN ADDRESS
.TCRET: .WORD 0 ;; CONTAINS RETURN ADDRESS WHEN DONE
      MOV .SRCHF,-(R1) ;; ISSUE SEARCH FORWARD COMMAND
      MOV (SP)+,R1 ;; RESTORE R1
      RTS PC ;; RETURN

;; INTERRUPT SERVICE ROUTINE
TCINT: MOV R1,-(SP) ;; SAVE R1 ON THE STACK
      MOV .TCDS,R1 ;; GET ADDRESS OF TCDS REGISTER
      TST (R1)+ ;; SET R1 = TO ADDRESS OF TCCM REGISTER
      TST (R1) ;; CHECK ERROR BIT IN TCCM
      BMI 65 ;; BRANCH IF ERROR
      CMP .SRCHF,.TCCOM ;; CHECK IF FINISHED SEARCH FWD
      BNE 45
      CMP .TCWDT,6(R1) ;; CHECK BLOCK # READ
      BEQ 55 ;; GO TO 55 IF BLOCK READ IS SAME AS BLOCK
      BLE 35 ;; REQUESTED
15: MOV .TCWDT,6(R1) ;; LOAD BLOCK #
      MOV .SRCHF,.TCCOM ;; SAVE COMMAND
      MOV .SRCHF,(R1) ;; ISSUE SEARCH FWD COMMAND
  
```

```

1306      25:      MOV      (SP)+,R1      ;;RESTORE R1
1307      RTI
1308
1309      ;;SET UP FOR REVERSE SEARCH
1310      35:      MOV      .TCDTR,6(R1)   ;;LOAD REVERSE BLOCK #
1311      MOV      .SRCHR,.TCCOM        ;;SAVE COMMAND
1312      MOV      .TCCOM,(R1)         ;;INITIATE READ REVERSE COMMAND
1313      BR      25                    ;;EXIT
1314
1315      45:      CMP      .SRCHR,.TCCOM   ;;CHECK IF FINISHED SEARCH REVERSE
1316      BNE      55
1317      CMP      .TCDTR,6(R1)         ;;CHECK BLOCK READ
1318      BEQ      15                    ;;GO INITIATE FWD SEARCH
1319      BR      35                    ;;CONTINUE REVERSE SEARCH
1320
1321      ;;INITIATE REQUEST COMMAND
1322      55:      BIT      #4,(R1)        ;;CHECK IF FINISHED WRITE OR READ
1323      BNE      75                    ;;BRANCH IF FINISHED WRITE OR READ
1324      MOV      .TCCMD,.TCCOM        ;;LOAD COMMAND
1325      MOV      .TCCMD,(R1)         ;;LOAD COMMAND
1326      BR      25                    ;;EXIT
1327
1328      ;;ERROR SERVICE
1329      65:      TST      -2(R1)         ;;CHECK IF END ZONE ERROR
1330      BMI      85                    ;;BRANCH IF END ZONE
1331      75:      TST      (R1)          ;;CHECK ERROR BIT IN TCCM
1332      BPL      +4
1333      SEV
1334      JSR      PC,@TCRET            ;;RETURN TO USER
1335      BR      25                    ;;EXIT
1336
1337
1338      85:      BIT      #4,(R1)        ;;CHECK IF READING OR WRITING
1339      BNE      75                    ;;BRANCH IF READING OR WRITING
1340      BIT      #4000,(R1)          ;;CHECK DIRECTION
1341      BNE      15                    ;;IF REV GO FWD
1342      BR      35
1343
1344      .ENDM      STCDRV
1345
1346      .MACRO      SLPMON
1347      ;;ROUTINE TO TYPE A MESSAGE ON THE LINE PRINTER
1348      ;; JSR      PC,.LPMON          ;;CALL TO SERVICE ROUTINE
1349      ;; ADDRESS      ;;ADDRESS OF FIRST CHARACTER
1350      ;;CHARACTER STRING ENDS IN A 0 BYTE
1351
1352      .LPMON:  NOP
1353      TST      .LPS                  ;;CHECK IF PREVIOUS REQUEST IS PENDING
1354      BEQ      +10                  ;;BRANCH IF NO REQUEST
1355      SUB      #4,(SP)              ;;RESET PC ON STACK TO RETURN TO
1356      RTS      PC                  ;;REQUESTER
1357      INC      .LPS                  ;;SET REQUEST PENDING INDICATOR
1358      BIT      #100,@#LPS           ;;CHECK IF LINE PRINTER IS BUSY
1359      BNE      -6
1360      DEC      .LPS                  ;;DECREMENT REQUEST INDICATOR
1361      MOV      #15,@#LPVEC          ;;SET INTERRUPT VECTOR

```

```

1362          MOV      #200, @#LPVEC+2  ;; AND PRIORITY ON INTERRUPT
1363          MOV      @#(SP), LPPTR    ;; GET FIRST ADDRESS OF CHAR STRING
1364          ADD      #2, (SP)         ;; ADJUST RETURN PC
1365          ASR      @#LPS            ;; SET INTERRUPT ENABLE
1366          RTS      PC                ;; RETURN
1367
1368          ;; LINE PRINTER INTERRUPT SERVICE ROUTINE
1369          15:      TST      @#LPS      ;; CHECK ERROR FLAG
1370          BMI      35                ;; BRANCH IF ERROR
1371          25:      TSTB     @LPPTR     ;; CHECK FOR TERMINATOR (0 BYTE)
1372          BEQ      55
1373          MOVB     @LPPTR, @#LPB      ;; LOAD CHAR INTO LINE PRINTER BUFFER
1374          INC      LPPTR             ;; STEP PTR TO NEXT CHARACTER
1375          TSTB     @#LPS            ;; CHECK IF DONE
1376          BMI      25                ;; LOAD NEXT CHAR IF DONE
1377          RTI
1378
1379          ;; HERE IF ERROR
1380          35:      HLT
1381          45:      CLR      @#LPS     ;; LINE PRINTER ERROR
1382          RTI                        ;; DISABLE IE BIT
1383
1384          ;; HERE WHEN DONE
1385          55:      MOV      #65, @#LPVEC
1386          RTI                        ;; EXIT
1387
1388          65:      NOP
1389          CLR      @#LPS             ;; CLEAR INTERRUPT ENABLE
1390          TST      .LPS             ;; CHECK IF REQUEST IS PENDING
1391          BEQ      75                ;; BRANCH IF NO REQUEST
1392          RTI                        ;; OTHERWISE EXIT TO REQUEST
1393
1394          75:      MOV      (PC), PC   ;; GET NEXT LINE PRINTER COMMAND
1395          85:      95                ;; CONTAINS ADDRESS OF NEXT LP COMMAND
1396          ;; PRINT PDP-11 IN BIG LETTERS
1397          95:      JSR      PC, DTSERV
1398          31.     LPDAT
1399          -600
1400          505    ;; DT1+IE+RDATA+DO
1401          CMP      #-540, @#TCWC    ;; WAIT FOR 80 CHARACTERS TO BE READ IN
1402          BNE      -6
1403          JSR      PC, LPSERV
1404          LPDAT
1405          RTI
1406          .LPS:   .WORD    0        ;; CONTAINS REQUEST INDICATOR
1407
1408          .ENDM   SLPMON
1409
1410          .MACRO  SMTDRV
1411          ;; ROUTINE TO READ/WRITE ON MAGTAPE. CALL:
1412          ;; JSR      PC, MTRV
1413          ;; CALLER MUST LOAD THE FOLLOWING PARAMETERS INTO A TABLE STRUCTURED
1414          ;; AS SHOWN BELOW:
1415          ;; .MTCMA: .WORD  0        ;; CONTAINS BUS ADDRESS
1416          ;; .MTBRC: .WORD  0        ;; CONTAINS BYTE TRANSFER COUNT
1417

```

1418 ;: MTRET: . WORD 0 ;: CONTAINS RETURN ADDRESS WHEN DONE
1419 ;: . MTC: . WORD 0 ;: CONTAINS COMMAND
1420
1421 ;: AND LEAVE RO POINTING TO THE FIRST ADDRESS OF THE TABLE (. MTCMA).

SMTREGS

1422
1423
1424
1425 . MTDRV: NOP
1426 MOV R1, -(SP) ;: SAVE R1 ON THE STACK
1427 MOV . MTS, R1 ;: GET ADDRESS OF MTS REGISTER
1428 ADD #6, R1 ;: FORM ADDRESS OF MTCMA REGISTER
1429 MOV (RO)+, (R1) ;: LOAD MTCMA REGISTER
1430 MOV (RO)+, -(R1) ;: LOAD MTBRC REGISTER
1431 MOV (SP)+, R1 ;: RESTORE R1
1432 MOV (RO)+, (PC)+ ;: GET RETURN ADDRESS WHEN DONE
1433 . MTRET: . WORD 0
1434 MOV (RO), @. MTC ;: LOAD COMMAND
1435 RTS PC

1436
1437
1438 MTINT: NOP
1439 MOV RO, -(SP) ;: SAVE RO ON THE STACK
1440 MOV . MTC, RO ;: LOAD ADDRESS OF MTC INTO RO
1441 JSR PC, @. MTRET ;: RETURN TO CALLER
1442 MOV (SP)+, RO ;: RESTORE RO
1443 RTI ;: EXIT TO MONITOR

1444 . ENDM SMTDRV
1445 . MACRO \$RFDRV

1446 ;: ROUTINE TO READ/WRITE ON THE RF DISK. CALL:
1447 ;: JSR PC, \$RFDRV
1448 ;: CALLER MUST LOAD THE FOLLOWING RF PARAMETERS INTO A TABLE STRUCTURED
1449 ;: AS SHOWN BELOW:

1450 ;: . RFDAA: . WORD 0 ;: CONTAINS EXTENDED DISK ADDRESS
1451 ;: . RFDA: . WORD 0 ;: CONTAINS DISK ADDRESS
1452 ;: . RFBA: . WORD 0 ;: CONTAINS BUS ADDRESS
1453 ;: . RFWC: . WORD 0 ;: CONTAINS WORD COUNT
1454 ;: . RFRET: . WORD 0 ;: CONTAINS INTERRUPT ADDRESS
1455 ;: . RFDCA: . WORD 0 ;: CONTAINS COMMAND

1456 ;: AND LEAVE RO POINTING TO THE FIRST ADDRESS OF THE TABLE (. RFDA).

\$RFREGS

1457
1458
1459
1460
1461 . RFDRV: MOV R1, -(SP) ;: SAVE R1 ON THE STACK
1462 MOV . RFDCA, R1 ;: GET ADDRESS OF RFDCA REGISTER
1463 ADD #10, R1 ;: FORM ADDRESS OF RFDAA REGISTER
1464 MOV (RO)+, (R1) ;: LOAD RFDAA REGISTER
1465 MOV (RO)+, -(R1) ;: LOAD RFDA REGISTER
1466 MOV (RO)+, -(R1) ;: LOAD RFCMA REGISTER
1467 MOV (RO)+, -(R1) ;: LOAD RFWC REGISTER
1468 MOV (SP)+, R1 ;: RESTORE R1
1469 MOV (RO)+, (PC)+ ;: LOAD RETURN ADDRESS WHEN DONE
1470 . RFRET: . WORD 0 ;: CONTAINS RETURN ADDRESS WHEN DONE
1471 MOV (RO), @. RFDCA ;: LOAD COMMAND INTO RFDCA REGISTER
1472 RTS PC ;: EXIT
1473

```
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481
```

RFINT: NOP
MOV RO, -(SP) ;:SAVE RO ON THE STACK
MOV .RFDCS, RO ;:LOAD ADDRESS OF RFDCS INTO RO
JSR PC, @.RFRET ;:RETURN TO CALLER
MOV (SP)+, RO ;:RESTORE RO
RTI ;:EXIT TO MONITOR
ENDM \$RFDRV

1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553

```
MACRO SRPDRV
;; ROUTINE TO READ WRITE ON THE RP11 DISK
;; CALL: JSR PC, RPDRV
;; CALLER MUST LOAD THE FOLLOWING RP PARAMETERS INTO A TABLE STRUCTURED AS
;; SHOWN BELOW:
;; .RPDA: .WORD 0 ;; CONTAINS DISK ADDRESS
;; .RPCA: .WORD 0 ;; CONTAINS THE CYLINDER ADDRESS
;; .RPBA: .WORD 0 ;; CONTAINS BUS ADDRESS
;; .RPWC: .WORD 0 ;; CONTAINS WORD COUNT
;; .RPRET: .WORD 0 ;; RETURN ADDRESS WHEN DONE
;; .RPCS: .WORD 0 ;; CONTAINS COMMAND WORD

;; AND LEAVE R0 POINTING TO FIRST ADDRESS OF THE TABLE (.RPDA)

SRPREGS
.RPDRV: NOP
MOV R1, -(SP) ;; SAVE R1 ON THE STACK
MOV .RPDS, R1
ADD #14, R1
MOV (R0)+, (R1) ;; LOAD DISK ADDRESS
MOV (R0)+, -(R1) ;; LOAD CYLINDER ADDRESS
MOV (R0)+, -(R1) ;; LOAD BUS ADDRESS
MOV (R0)+, -(R1) ;; LOAD WORD COUNT
MOV (SP)+, R1 ;; RESTORE R1
MOV (R0)+, (PC)+ ;; LOAD RETURN ADDRESS WHEN DONE
.RPRET: .WORD 0 ;; CONTAINS RETURN ADDRSS WHEN DONE
MOV (R0), @ (PC)+ ;; LOAD COMMAND INTO RPCS
.RPCS: .WORD 0 ;; CONTAINS ADDRESS OF RPCS REGISTER
RTS PC

RPINT: NOP
JSR PC, @.RPRET ;; RETURN TO CALLER
RTI ;; RETURN

ENDM SRPDRV
```


1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592

```
MACRO SRP4DRV
;; ROUTINE TO READ/WRITE ON RPO4/RPO5 DISK. CALL:
JSR PC, RP4DRV
;; CALLER MUST LOAD THE FOLLOWING RP PARAMETERS INTO A TABLE STRUCTURED
;; AS SHOWN BELOW:
;; .RPCS2: .WORD 0 ;; CONTAINS DISK UNIT #
;; .RPDST: .WORD 0 ;; CONTAINS DESIRED SECTOR & TRACK ADDRESS
;; .RPCA: .WORD 0 ;; CONTAINS CYLINDER ADDRESS
;; .RPBA: .WORD 0 ;; CONTAINS BUS ADDRESS
;; .RPWC: .WORD 0 ;; CONTAINS WORD COUNT
;; .RPRET: .WORD 0 ;; CONTAINS RETURN ADDRESS WHEN DONE
;; .RPCS1: .WORD 0 ;; CONTAINS COMMAND

;; AND LEAVE RO POINTING TO FIRST ADDRESS IN THE TABLE (.RPCS2)

SRP4REGS

RP4DRV: NOP
MOV R1, -(SP) ;; SAVE R1 ON THE STACK
MOV .RPCS1, R1 ;; GET ADDRESS OF CS1 REGISTER
ADD #10, R1 ;; FORM ADDRESS OF CS2 REGISTER
MOV (RO)+, (R1) ;; LOAD UNIT # (RPCS2)
MOV (RO)+, -(R1) ;; LOAD TRACK/SECTOR (RPDST)
MOV (RO)+, 26(R1) ;; LOAD CYLINDER ADDRESS (RPCA)
MOV (RO)+, -(R1) ;; LOAD BUS ADDRESS (RPBA)
MOV (RO)+, -(R1) ;; LOAD WORD COUNT (RPWC)
MOV (RO)+, (PC)+ ;; LOAD RETURN ADDRESS

RPRET: .WORD 0
MOV (SP)+, R1 ;; RESTORE R1
NOP
MOV (RO), @.RPCS1 ;; LOAD COMMAND
RTS PC

RPINT: NOP
JSR PC, @.RPRET
RTI
ENDM SRP4DRV
```

1594
 1595
 1596
 1597
 1598
 1599
 1600
 1601
 1602
 1603
 1604
 1605
 1606
 1607
 1608
 1609
 1610
 1611
 1612
 1613
 1614
 1615
 1616
 1617
 1618
 1619
 1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1649

```

    .MACRO $RSDRV
    ;; ROUTINE TO READ/WRITE ON RSD4/RSD5 DISK. CALL:
    ;; JSR PC, RSDRV
    ;; CALLER MUST LOAD THE FOLLOWING RS PARAMETERS INTO A TABLE STRUCTURED
    ;; AS SHOWN BELOW:
    ;; RSCS2: .WORD 0           ;; CONTAINS UNIT #
    ;; RSDA:   .WORD 0           ;; CONTAINS DISK ADDRESS
    ;; RSBA:   .WORD 0           ;; CONTAINS BUS ADDRESS
    ;; RSWC:   .WORD 0           ;; CONTAINS WORD COUNT
    ;; RSRET:  .WORD 0           ;; CONTAINS RETURN ADDRESS WHEN DONE
    ;; RSCS1: .WORD 0           ;; CONTAINS COMMAND
  
```

;; AND LEAVE RO POINTING TO THE FIRST ADDRESS IN THE TABLE (.RSCS2)

\$RSREGS

```

.RSDRV: NOP
        MOV R1, -(SP)           ;; SAVE R1 ON THE STACK
        MOV RSCS2, R1          ;; GET ADDRESS OF RSCS2 REGISTER
        MOV (RO)+, (R1)        ;; LOAD UNIT # (RSCS2)
        MOV (RO)+, -(R1)       ;; LOAD DISK ADDRESS (RSDA)
        MOV (RO)+, -(R1)       ;; LOAD BUS ADDRESS (RSBA)
        MOV (RO)+, -(R1)       ;; LOAD WORD COUNT (RSWC)
        MOV (RO)+, (PC)+       ;; LOAD INTERRUPT RETURN ADDRESS
.RSRET: .WORD 0
        MOV (SP)+, R1          ;; RESTORE R1
        MOV (RO), @.RSCS1      ;; LOAD COMMAND
        RTS PC                 ;; RETURN TO CALLER
  
```

```

RSINT:  NOP
        JSR PC, @.RSRET
        RTI                    ;; EXIT TO MONITOR
        .ENDM $RSDRV
  
```

.MACRO \$HLT

```

    .PAGE
    ;; ERROR SERVICE CALLED BY HLT (EMT) INSTRUCTION
    ;; ROUTINE IS ENTERED BY AN EMT INSTRUCTION AND THE FORMAT CONTROL BYTE
    ;; AS AN ARGUMENT TO THE EMT.
    HLT=EMT
  
```

```

.HLT:  NOP
        INC @#ERTTL           ;; INCREMENT TOTAL ERROR COUNT
        MOV #1, @#ERRFLG     ;; SET ERROR FLAG
        MOV (SP), @#ERRPC    ;; FORM PC OF ERROR CALL (HLT)
        SUB #2, @#ERRPC      ;; GET PSW AT TIME OF ERROR
        MOV 2(SP), @#ERRPSW   ;; CHECK FOR HALT ON ERROR
        TST @SWR              ;; BRANCH IF HALT ON ERROR
        BMI .HLT3            ;; CHECK FOR LOOP ON ERROR SWITCH
        BIT #SW9, @SWR
        BEQ 1$
        MOV @#ERTTL, @#DISPLAY ;; DISPLAY ERROR COUNT
        MOV @#EPC, (SP)      ;; SET UP RETURN FROM ERROR
1$:    BIT #SW11, @SWR        ;; CHECK INHIBIT TYPEOUT SEITCH
        BNE .HLT1           ;; BRANCH IF NO TYPEOUT DESIRED
.HLT1:
  
```

```

1650      SAVE
1651      MOV      @#ERRPC,R2      ;; GET PC OF ERROR CALL
1652      MOV      R2,R3          ;; SAVE IN R3
1653      CNV16
1654      TYPE
1655      ERRPCM
1656      MOVB     (R3),R0        ;; GET ERROR CALL INSTRUCTION
1657      BNE      .HLT4          ;; BRANCH IF EMT+X
1658      HLT0:   RESTORE
1659      HLT1:   BIT      #SW10,@SWR  ;; RING BELL ON ERROR
1660      BEQ      HLT2
1661      TYPE
1662      BELL
1663      HLT2:   TST      @SWR      ;; CHECK HALT ON ERROR SWITCH
1664      BPL      +4
1665      HLT3:   HALT
1666      RTI
1667
1668      HLT4:   NOP
1669      BIC      #177400,R0
1670      CMP      #1,R0          ;; CHECK IF DEVICE ERROR CALL (DEVERR)
1671      BNE      .HLT5
1672      MOV      4(R3),R0        ;; GET # OF REGISTERS TO DUMP
1673      MOV      6(R3),R1        ;; AND ADDRESS OF FIRST DEVICE REGISTER
1674      MOV      10(P3),XFERS    ;; GET ADDRESS OF TRANSFER COUNT
1675      MOV      R1,R2
1676      TYPE
1677      ERRDEV
1678      CNV16
1679      TYPE
1680      DIGITS
1681      TYPE
1682      SCRLF
1683
1684      15:     MOV      (R1)+,R2
1685      CNV16
1686      TYPE
1687      DIGITS
1688      DEC      R0
1689      BNE      15
1690      BR      .HLTEX
1691
1692      HLT5:   CMP      #2,R0      ;; CHECK IF DATA ERROR CALL (DATERR)
1693      BLT      .HLT7
1694      RESTORE
1695      SAVE
1696      TYPE
1697      DEVID
1698      CNV18
1699      TYPE
1700      DIGITS
1701      TYPE
1702      GDDAT
1703      MOV      R0,R2          ;; GET GOOD DATA
1704      CNV16
1705      TYPE
  
```

```
1706          DIGITS
1707          TYPE
1708          BDDAT
1709          MOV      R3,R2          ;; GET BAD DATA
1710          CNV16
1711          TYPE
1712          DIGITS
1713          HLTEX:  TYPE
1714          SCRLF
1715          TYPE
1716          XFER
1717          CNV48
1718          TYPE
1719          DJGBUF
1720          TYPE
1721          SCRLF
1722          BR      .HLTO
1723
1724
1725          HLT7:  HALT
1726
1727          ;; DIGIT TABLE
1728          DIGTAB: "01
1729                  "23
1730                  "45
1731                  "67
1732          ERRPCM: .ASCII 'ERROR PC= '
1733          DIGITS: .ASCIZ '000000 '
1734          SCRLF:  .ASCIZ <15><12>
1735          BELL:   .ASCIZ <7>
1736          ERRDEV: .ASCIZ 'DEVICE ERROR'<15><12>
1737          DEVID:  .ASCII 'XX DATA ERROR'<15><12>
1738          ADRS:   .ASCIZ 'MEMORY ADDRESS= '
1739          GDDAT:  .ASCIZ 'GOOD DATA= '
1740          BDDAT:  .ASCIZ 'BAD DATA= '
1741          XFER:   .ASCIZ 'DEVICE TRANSFER COUNT= '
1742          DIGIT:  .ASCIZ '0'
1743                  .EVEN
1744                  .ENDM          $HLT
1745
1746          .MACRO          $SCATCH
1747          .NLIST
1748          =0
1749          0
1750          0
1751          .REPT          174
1752          +2
1753          HALT
1754          .ENDR
1755          .LIST
1756          .ENDM          $SCATCH
1757          .MACRO          $SST200
1758          =200
1759          JMP          @#START          ;; GO TO START OF TEST
1760          .ENDM          $SST200
1761          .MACRO          $CMTAG
```

1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817

```
;; WORDS LOADED BY 'HLT'  
ERRPC: .WORD 0 ;; CONTAINS PC OF ERROR DETECTED  
ERRPSW: .WORD 0 ;; CONTAINS PSW AT TIME OF ERROR  
ERTTL: .WORD 0 ;; CONTAINS TOTAL ERRORS DETECTED COUNT  
SCPBLK: .WORD 0 ;; CONTAINS PASS COUNT  
ICNT=SCPBLK  
ITCNT: .WORD 2 ;; CONTAINS SUBTEST ITERATION COUNT  
LASTPC: .WORD 4 ;; CONTAINS LAST SCOPE CALL PC  
  
EPC: .WORD 0 ;; CONTAINS SCOPE RETURN FOR ERRORS  
ERRFLG: .WORD 0 ;; CONTAINS ERROR FLAG  
TICKS: .WORD 0 ;; CONTAINS TICK COUNT FOR CLOCKS  
 .ENDM $CMTAG  
  
 .MACRO $VTK$ ADR,LEN  
JSR R5,$VTK$ ;; GO SETUP TRAP VECTORS  
ADR  
LEN  
 .ENDM $VTK$  
 .MACRO $$VTK$  
;; ROUTINE TO SETUP TRAP VECTORS  
$VTK$: MOV R0,-(SP) ;; SAVE R0 ON THE STACK  
MOV (R5)+,R0 ;; GET TABLE ADDRESS  
TST (R5)+ ;; POP LENGTH ARGUMENT  
MOV (R0)+,@#ERRVEC ;; SET ERROR TRAP VECTOR  
MOV (R0)+,@#MMVEC ;; SET MEM MGMT ABORT TRAP VECTOR  
MOV (R0)+,@#BPTVEC ;; SET 'BPT' TRAP VECTOR  
MOV (R0)+,@#IOTVEC ;; SET IOT TRAP VECTOR  
MOV (R0)+,@#RESVEC ;; SET RESERVED INSTRUCTION TRAP VECTOR  
MOV (R0)+,@#EMTVEC ;; SET EMT TRAP VECTOR  
MOV (R0)+,@#TRAPVEC ;; SET TRAP TRAP VECTOR  
MOV (R0)+,@#FPEVEC ;; SET FLOATING POINT TRAP VECTOR  
MOV (SP)+,R0 ;; RESTORE R0  
RTS R5 ;; RETURN  
  
 .ENDM $$VTK$  
 .MACRO $VECTAB  
VECTAB:  
ERRTRP: .WORD ERRVEC+2  
MMVEC: .WORD MMVEC+2  
BPTVEC: .WORD BPTVEC+2  
IOTVEC: .WORD IOTVEC+2  
RESVEC: .WORD RESVEC+2  
EMTVEC: .WORD HLT  
TRPVEC: .WORD TRAPS  
FPEVEC: .WORD FPEVEC+2  
  
 .ENDM $VECTAB  
  
 .MACRO $STARTUP  
;; START OF PROGRAM  
START: MOV #STKPTR,SP ;; SET STACK POINTER  
CLR @#SCPBLK ;; CLEAR PASS COUNT  
$VTK$ VECTAB,8.  
 .ENDM $STARTUP
```

1818
 1819
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1829
 1830
 1831
 1832
 1833
 1834
 1835
 1836
 1837
 1838
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1849
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1859
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1869
 1870
 1871
 1872
 1873

```

      .MACRO $MAMFO
;; ROUTINE TO SET ACTION ENABLE ON MA/MF PARITY MEMORIES
;; CALL: JSR PC, .MAMF

      PARCSR= 172100      ;; ADDRESS OF FIRST MA/MF PARITY CSR
      PARVEC= 114        ;; ADDRESS OF PARITY INTERRUPT VECTOR

.MAMF: BIT #40, @SWR      ;; CHECK IF PARITY ERROR DETECTION IS TO
      BNE 3$             ;; ENABLE. BRANCH IF NOT TO BE ENABLED
      MOV #.PARSRV, @#PARVEC ;; SET PARITY INTERRUPT VECTOR
      MOV #340, @#PARVEC+2 ;; AND PRIORITY LEVEL 7 ON INTERRUPT
      MOV @#ERRVEC, -(SP) ;; SAVE CURRENT ERROR VECTOR
      MOV @#ERRVEC+2, -(SP) ;; AND PRIORITY LEVEL
      MOV #ERRVEC+2, @#ERRVEC
      MOV #RT1, @#ERRVEC+2
      MOV #PARCSR, R0    ;; GET FIRST CSR ADDRESS
      MOV #1, R2

1$: SEZ                ;; SET TIME OUT INDICATOR
      MOV #1, (R0)+    ;; SET ACTION ENABLE IF AVAILABLE
      BEQ 2$           ;; BRANCH IF CSR NOT AVAILABLE
2$: ASL R2              ;; SHIFT AVAILABILITY INDICATOR
      BCC 1$
      MOV (SP)+, @#ERRVEC+2 ;; RESTORE ERROR VECTOR PRIORITY LEVEL
      MOV (SP)+, @#ERRVEC ;; AND INTERRUPT VECTOR
3$: RTS PC

;; PARITY ERROR SERVICE ROUTINE
;; WHEN A PARITY ERROR IS DETECTED THIS ROUTINE TYPES A MESSAGE
;; 'PARITY ERROR'
;; THEN SCANS MEMORY FOR THE ADDRESS CAUSING THE PARITY ERROR. WHEN THE
;; ADDRESS IS LOCATED THE ADDRESS IS PRINTED, AND THE PROGRAM HALTS.
;; PRESS CONTINUE TO RESTART.
.PARSRV: TYPE
      PARERR
      CLR R1           ;; SET FIRST ADDRESS
      MOV #2$, @#PARVEC ;; SET PARITY ERROR VECTOR
      MOV #4$, @#ERRVEC ;; SET ERROR TRAP VECTOR
      MOV #5$, @#MMVEC  ;; SET MEM MGMT ABORT VECTOR
1$: TST (R1)+          ;; SCAN MEMORY
      BR 1$

2$: TYPE
      ADRS
      CNV18
      TYPE
      DIGITS
3$: HALT
      JMP @#START1

4$: MOV 4(SP), R1     ;; GET PC AT TIME OF PARITY ERROR
      CNV18
      TYPE
      NOPAR
      TYPE
  
```

1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929

```
ERRPCM  
BR 3$  
5$: ADD #200, @#KIPAR2  
MOV #40000, R1 ;; RESET ADDRESS  
RTI  
PARERR: .ASCIZ 'PARITY ERROR'  
NOPAR: .ASCIZ 'NO PARITY ERROR DETECTED ON SCAN'  
.EVEN  
.ENDM $MAMFO  
.MACRO DTBOOT  
;; BOOT LOADER  
;; THE FOLLOWING ROUTINE WILL LOAD THE PROGRAM INTO MEMORY FROM DECTAPE  
BLKODAT: CLR R0  
MOV #64, R1  
MOV #20000, R2  
1$: MOV (R0)+, (R2)+ ;; RELOCATE  
DEC R1 ;; DECREMENT WORD COUNT  
BNE 1$  
ADD #20000, PC ;; GO TO RELOCATED CODE  
MOV #-4096, @#TCWC ;; LOAD DECTAPE WORD COUNT  
CLR @#TCBA ;; LOAD MEMORY TRANSFER ADDRESS  
2$: MOV #3, @#TCCM ;; SEARCH FWD ON UNIT 0  
TSTB @#TCCM ;; WAIT FOR READY  
BPL -4  
CMP #10, @#TCDT ;; CHECK BLOCK #  
BNE 2$  
MOV #5, @#TCCM ;; ISSUE READ COMMAND  
TSTB @#TCCM ;; WAIT FOR READ TO FINISH  
BPL -4  
CLR @#TCCM  
HALT  
.ENDM  
.MACR LPDP11  
;; LP DATA BUFFER  
LPDAT: .ASCII 'PPPPPPPP DDDDDDDD PPPPPPPP 11 11'<15><12>  
.ASCII 'PP PP DD DD PP PP 111 111'<15><12>  
.ASCII 'PP PP DD DD PP PP 1111 1111'<15><12>  
.ASCII 'PP PP DD DD PP PP 11 11'<15><12>  
.ASCII 'PP PP DD DD PP PP 11 11'<15><12>  
.ASCII 'PP PP DD DD PP PP 11 11'<15><12>  
.ASCII 'PP PP DD DD PP PP 11 11'<15><12>  
.ASCII '?PPPPPPP DD DD PPPPPPPP 11 11'<15><12>  
.ASCII 'PP DD DD PP 11 11'<15><12>  
.ASCII 'PP DD DD PP 11 11'<15><12>  
.ASCII 'PP DD DD PP 11 11'<15><12>  
.ASCII 'PP DD DD PP 111111 111111'<15><12>  
.ASCII 'PP DDDDDDDD PP 111111 111111'<15><12>  
.ASCIZ 'PDP-11'<15><12>  
.EVEN  
.ENDM
```

DZTUD-D TMO2 DRIVE FUNCTION TIMER
DZTUDD. SML 21-OCT-77 09:54

MACY11 30(1046) 27-OCT-77 ^{L 5} 13:42 PAGE 14-6

SEQ 0063

1930


```
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987
```

```
      .NLIST MC  
      .LIST ME  
      .ENABLE ABS,AMA  
      .MCALL SCPREG,SCPVEC,SCPREG,SCATCH,STYPE  
      .TITLE DZTUD-D TMO2 DRIVE FUNCTION TIMER  
      .SBTTL STARTING INSTRUCTIONS  
; LOADING AND STARTING PROCEEDURE  
;  
; LOAD PROGRAM USING ABS LOADER  
;  
; LOAD ADDRESS 200  
;  
; SET SWITCH OPTIONS  
;  
; PRESS START  
  
; GENERAL REGISTER USAGE:  
;  
; R0=ADDRESS OF 'FC' REGISTER (SET BY SCOPE)  
;  
; R1=ADDRESS OF 'DS' REGISTER (SET BY SCOPE)  
;  
; R2=RETURN PC FROM TIMER (SET BY EACH TEST)  
;  
; R3=INDEX INDICATING PREVIOUS OSCILLATOR POLARITY (SET BY TIMER)  
;  
; R4=CONTAINS 'TICK' COUNT WHEN TIMER IS RUNNING (SET BY TIMER)  
;  
; R5=ADDRESS OF CS1 (SET BY SCOPE)  
  
; SWITCH REGISTER SWITCH ASSIGNMENTS  
SW15= 100000 ;HALT ON ERROR  
SW14= 040000 ;LOOP SUBTEST  
SW13= 020000 ;INHIBIT ERROR TYPE OUT  
SW11= 004000 ;INHIBIT SUBTEST ITERATION  
SW10= 002000 ;INHIBIT PUBLICATION OF FUNCTION TIMES  
SW09= 001000 ;RING BELL ON ERROR  
SW08= 000400 ;TYPE LINE ITEM AFTER EACH ITERATION  
SW07= 000200 ;HALT ON TEST SELECTED IN SW05-SW00  
SW06= 000100 ;CONTINUOUS CYCLE  
  
      .SBTTL MACRO DEFINITIONS  
      .MACRO SAVE  
      JSR PC, .SAVE ;SAVE REGISTERS ON THE STACK  
      .ENDM  
      .MACRO RESTORE  
      JSR PC, .RESTORE ;RESTORE REGISTERS FROM THE STACK  
      .ENDM  
      .MACRO INPUT  
      JSR PC, .INPUT ;GET USER INPUT  
      .ENDM  
      .MACRO REWIND  
      JSR PC, .REWIND ;REWIND SLAVE  
      BVS 99$ ;BRANCH IF ERROR ON REWIND  
      .ENDM  
      .MACRO TIMEON  
      JSR PC, .TIMON ;TURN TIMER ON  
      .ENDM  
      .MACRO TIMCHK  
      JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2  
      .ENDM  
      .MACRO SETGO  
      INC (R5) ;SET 'GO' BIT  
      .ENDM
```

```

1988
1989
1990
(1)
(1) 000000
(1) 000001
(1) 000002
(1) 000003
(1) 000004
(1) 000005
(1) 000006
(1) 000007
(1) 000000
(1) 000001
(1) 000002
(1) 000003
(1) 000004
(1) 000005
(1)
(1)
(1) 177776
(1) 177774
(1) 177772
(1) 177770
(1) 177560
(1) 177562
(1) 177564
(1) 177566
(1)
1991
(1) 000004
(1) 000010
(1) 000014
(1) 000014
(1) 000014
(1) 000020
(1) 000024
(1) 000030
(1) 000034
(1) 000060
(1) 000064
(1) 000114
(1) 000240
(1) 000244
(1) 000250
(1)
1992
1993 172540
1994 000104
1995 177546
1996 000100
1997 177514
1998 177516
1999
2000
2001 172440

      .SBTTL REGISTER ASSIGNMENTS
      ;; DEFINITIONS AND REGISTER ASSIGNMENTS
      ;; GENERAL REGISTER ASSIGNMENTS
      RO=%0
      R1=%1
      R2=%2
      R3=%3
      R4=%4
      R5=%5
      SP=%6
      PC=%7
      R10=%0
      R11=%1
      R12=%2
      R13=%3
      R14=%4
      R15=%5

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

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

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

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

      ;CLOCK ADDRESS AND VECTORS
      PLKCSR= 172540
      PLKVEC= 104
      LKS= 177546
      LKVEC= 100
      LPS= 177514
      LPB= 177516

      ;KW11-P
      ;KW11-L
      ;LP11

      ;RH11, TMO2/TU16 REGISTERS
      TMCS1= 172440

```

2002
2003
2004 000000
2005 000002
2006 000004
2007 000006
2008 000010
2009 000012
2010 000014
2011 000016
2012 000022
2013 000024
2014 000026
2015 000030
2016 000032
2017
2018
2019
2020 000001
2021 000000
2022 000002
2023 000006
2024 000010
2025 000026
2026 000024
2027 000030
2028 000032
2029 000050
2030 000056
2031 000060
2032 000070
2033 000076
2034 000100
2035 000200
2036 000400
2037 001000
2038 002000
2039 004000
2040 020000
2041 040000
2042 100000
2043
2044 000000
2045 000001
2046 000002
2047 000003
2048 000004
2049 000005
2050 000006
2051 000007
2052 000010
2053 000020
2054 000040
2055 000100
2056 000200
2057 000400

; TMO2/TU16 INDEX VALUES

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

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

. SBTTL TMO2/TU16 REGISTER BITS
; RHCS1-CS1(R5)

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

; RHCS2-CS2(R5)

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

2058	001000	MXF=	1000
2059	002000	PGE=	2000
2060	004000	NEM=	4000
2061	010000	NED=	10000
2062	020000	UPE=	20000
2063	040000	WCE=	40000
2064	100000	DLT=	100000
2065		;RHDS-DS(R5)	
2066	000001	SLA=	1
2067	000002	BOT=	2
2068	000004	TMK=	4
2069	000010	IDB=	10
2070	000020	SDWN=	20
2071	000040	PES=	40
2072	000100	SSC=	100
2073	000200	DRY=	200
2074	000400	DPR=	400
2075	002000	EOT=	2000
2076	004000	WRL=	4000
2077	010000	MOL=	10000
2078	020000	PIP=	20000
2079	040000	ERR=	40000
2080	100000	ATA=	100000
2081		;RHER-ER(R5)	
2082	000001	ILF=	1
2083	000002	ILR=	2
2084	000004	RMR=	4
2085			
2086	000020	FMT=	20
2087	000100	INCVAE=	100
2088	000200	PEFLRC=	200
2089	000400	NSG=	400
2090	001000	FCE=	1000
2091	002000	CSITM=	2000
2092	004000	NEF=	4000
2093	010000	DTE=	10000
2094	020000	OPI=	20000
2095	040000	UNS=	40000
2096			
2097		;RHMR-MR(R5)	
2098	000100	OSC=	100
2099			
2100		;RHDT-DT(R5)	
2101	002000	SPR=	2000
2102	010000	CH7=	10000
2103	040000	TAP=	40000
2104			
2105		;RHTC-TC(R5)	
2106	000300	NORM11=	300
2107	000320	CDM11=	320
2108	000000	BP1200=	0
2109	000400	BP1556=	000400
2110	001000	BP1800=	001000
2111	002000	PE1600=	002000
2112	100000	ACCL=	100000
2113			

2114		; INSTRUCTION EQUATES	
2115	104400	HLT= TRAP	
2116	104000	SCOPE= EMT	
2117	000004	TYPE= IOT	
2118			
2119		; MISCELLANEOUS EQUATES	
2120	005750	OUTBUF=INIT	; OUTPUT BUFFER START AT BEG OF PROGRAM
2121	177400	FRMCNT= -256.	; FRAME COUNT
2122	177600	WRDCNT= -128.	; WORD COUNT
2123		; ASCII EQUATES	
2124	000003	CNTRLC= 3	; ASCII CODE FOR CONTROL C (C)
2125	000011	HT= 11	; ASCII CODE FOR HORIZONTAL TAB
2126	000012	LF= 12	; ASCII CODE FOR LINE FEED
2127	000015	CR= 15	; ASCII CODE FOR CARRIAGE RETURN
2128	000017	CNTRLO= 17	; ASCII CODE FOR CONTROL O (O)
2129	000025	CNTRLU= 25	; ASCII CODE FOR CONTROL U (U)

```

2132 ; SETUP TRAP VECTORS
2133   =TBITVEC
2134   .WORD +2 ; SET 'T' TRAP TO TIMER ROUTINE
2135   .WORD HALT ; PRIORITY LEVEL 7
2136   .WORD .TYPE ; SET IOT TRAP TO .TYPE ROUTINE
2137   .WORD 0 ; PRIORITY LEVEL 0
2138   .WORD PFVEC+2 ; POWER FAIL TRAP TO HALT
2139   .WORD HALT ; AT PFVEC+2
2140   .WORD .SCOPE ; SET EMT TRAP TO .SCOPE ROUTINE
2141   .WORD 340 ; PRIORITY LEVEL 7
2142   .WORD .HLT ; SET TRAP TRAP TO .HLT ROUTINE
2143   .WORD 340 ; PRIORITY LEVEL 7
2144   =42 ; ACT11
2145   .WORD 0 ; "
2146   =46 ; "
2147   .WORD ENDAD ; "
2148   =52 ; "
2149   .WORD 0 ; "
2150   =TKVEC
2151   .WORD TKISR
2152   .WORD 340

2153 ; SOFTWARE SWITCH REGISTER LOC. 176
2154   =176
2155   SWREG: 0 ; SOFTWARE SWITCH REGISTER
2156   000176 000000
2157   =200
2158   000200
2159   000200 000137 005750 JMP @#INIT ; GO TO START OF PROGRAM
2160   =500
2161   000500
2162   000600 STKPTR= 600 ; STACK
2163   =1000
2164   001000
2165 ; PROGRAM TAGS
2166   001000 000000 PASS: .WORD 0
2167   001002 177570 SWR: 177570 ; SWITCH REGISTER
2168   001004 000000 SCPADR: .WORD 0
2169   001006 000 DRVNUM: .BYTE 0 ; TMO2 DRIVE UNDER TEST
2170   001007 000 SLVNUM: .BYTE 0 ; TU16 SLAVE UNDER TEST
2171   001010 000000 SLVPTR: .WORD 0 ; POINTER TO SLAVE TABLE (SLVTBL) BELOW
2172   001012 172440 TMBASE: .WORD TMC51 ; BASE ADDRESS OF TMO2/TU16 REGISTERS
2173   001014 000000 ATIME: .WORD 0 ; CONTAINS 'TICK' COUNT
2174   001016 000020 ATIMTBL: .BLKW 16. ; EACH ENTRY CONTAINS TIME FOR FUNCTION
2175   ; ENTRIES ARE MADE BY 'SCOPE' ROUTINE
2176   001056 000020 GAP: .BLKW 16. ; TIMES RECORDED BY 'GAP CONSISTANCY' TEST
2177   001116 000000 DELTIM: .WORD 0 ; VARIABLE DELAY
2178   001120 000000 OCTALO: .WORD 0
2179   001122 000 GAP: .BYTE 0 ; CONTAINS GAP # (USED FOR TST 021)
2180   001123 000 ITCNT: .BYTE 0 ; ITERATION COUNT
2181   001124 000 TSTNUM: .BYTE 0 ; TEST #
2182   001125 000 ERFLG: .BYTE 0 ; ERROR FLAG
2183   001126 000 PRGFLG: .BYTE 0 ; PROGRAM FLAG
2184   001127 000 UNTFND: .BYTE 0 ; UNIT FOUND INDICATOR
2185   001130 000 TYPFLG: .BYTE 0
2186   001131 000 NRZFLG: .BYTE 0 ; INDICATES IF DRIVE IS NRZ ONLY.
2187   001132 000 ASFLG: .BYTE 0 ; 1/0 = YES/NO.
  
```

2188		001134	
2189	001134	030460	
2190	001136	031462	
2191	001140	032464	
2192	001142	033466	
2193	001144	034470	
2194	001146	000006	
2195	001154	000	
2196		001156	
2197	001156	000010	
2198	001166	000100	
2199	001266	000110	
2200	001376	005015	000
2201	001401	134	000
2202	001403	060	000
2203	001405	007	000
2204	001407	055	000
2205	001411	040	
2206	001412	000040	
2207	001414	004476	000
2208		001420	

DIGTAB:	. EVEN	
	"01	
	"23	
	"45	
	"67	
	"89	
ODIGITS:	. BLKB	6
	. BYTE	0
	. EVEN	
DRVTBL:	. BLKB	8.
SLVTBL:	. BLKB	64.
INBUF:	. BLKB	72.
CRLF:	. ASCIZ	<CR><LF>
BKSLSH:	. ASCIZ	' '
ECHO:	. ASCIZ	'0'
BELL:	. ASCIZ	<7>
DASH:	. ASCIZ	'-'
SPACE2:	. ASCII	' '
SPACE:	. ASCIZ	' '
ANGTAB:	. ASCIZ	'>'<HT>
	. EVEN	

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

2210
2211
2212
2213
2214
2215
2216
2217 001420 000000 000000
2218 001424 044560 043740
2219 001430 001666 001546
2220 001434 001572 001522
2221 001440 002506 001452
2222 001444 035540 035064
2223 001450 000500 000404
2224 001454 000721 000651
2225 001460 002506 001452
2226 001464 000500 000404
2227 001470 000562 000512
2228 001474 002506 001452
2229 001500 003206 002056
2230 001504 003206 002056
2231 001510 002412 001666
2232 001514 002234 001522
2233 001520 002710 002532
2234 001524 002544 002330
2235 001530 000000 000000
2236 001534 004552 004406
2237 001540 004540 004374
2238 001544 004540 004374
2239 001550 004716 004552
2240 001554 023564 023254
2241 001560 024404 024074
2242 001564 004336 004172
2243 001570 004336 004172
2244
2245

.SBTTL TIME SPECIFICATION TABLE

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

WORD	MAX, MIN	TIME IN MS	FUNCTION	TEST #
STIMTBL: WORD	0, 0	; SPARE		
WORD	18800. , 18400.	; 188. 0-184. 0	WRITE FROM BOT	TST001
WORD	00950. , 00870.	; 9. 5-8. 7	WRITE START	TST002
WORD	00890. , 00850.	; 8. 9-8. 5	WRITE SHUTDOWN	TST003
WORD	01350. , 00810.	; 13. 5-8. 1	WRITE STLDOWN	TST004
WORD	15200. , 14900.	; 152. 0-149. 0	READ FROM BOT	TST005
WORD	00320. , 00260.	; 3. 2-2. 6	READ START	TST006
WORD	00465. , 00425.	; 4. 65-4. 25	READ SHUTDOWN	TST007
WORD	01350. , 00810.	; 13. 5-8. 1	READ SETTLEDOWN	TST010
WORD	00320. , 00260.	; 3. 2-2. 6	RD REV START	TST011
WORD	00370. , 00330.	; 3. 7-3. 3	RD REV SHTDWN	TST012
WORD	01350. , 00810.	; 13. 5-8. 1	RD REV STLDWN	TST013
WORD	01670. , 01070.	; 16. 7-10. 7	TRN RND DLY F-R	TST014
WORD	01670. , 01070.	; 16. 7-10. 7	TRN RND DLY R-F	TST015
WORD	01290. , 00950.	; 12. 9-9. 5	GAP SIZE STOP	TST016
WORD	01180. , 00850.	; 11. 8-8. 5	GAP SIZE STRT	TST017
WORD	01480. , 01370.	; 14. 8-13. 7	GAP SIZE INTER	TST020
WORD	01380. , 01240.	; 13. 8-12. 4	GAP CONSISANCY	TST021
WORD	0, 0	; 0. 0-0. 0	DUMMY	TST022
WORD	02410. , 02310.	; 24. 1-23. 1	DAT TIME 200BPI	TST023
WORD	02400. , 02300.	; 24. 0-23. 0	DAT TIME 556BPI	TST024
WORD	02400. , 02300.	; 24. 0-23. 0	DAT TIME 800BPI	TST025
WORD	02510. , 02410.	; 25. 1-24. 1	DAT TIME 1600PE	TST026
WORD	10100. , 09900.	; 101. 0-99. 0	ERASE	TST027
WORD	10500. , 10300.	; 105. 0-103. 0	WRT FILE MARK	TST030
WORD	02270. , 02170.	; 22. 7-21. 7	READ 1" TAPE	TST031
WORD	02270. , 02170.	; 22. 7-21. 7	RD REV 1" TAPE	TST032

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

2247
 2248
 2249
 2250
 2251
 2252
 2253
 2254 001574 002602 002412
 2255 001600 002652 002506
 2256 001604 002734 002532
 2257 001610 002734 002532
 2258 001614 002734 002424
 2259 001620 002652 002260
 2260 001624 002652 002260
 2261 001630 002652 002260
 2262 001634 002532 002260
 2263 001640 002532 002260
 2264 001644 002532 002260
 2265 001650 002532 002260
 2266 001654 002532 002260
 2267 001660 002532 002260
 2268 001664 002532 002260
 2269 001670 002532 002260

SBTTL GAP TIME SPECIFICATION TABLE
 ; THIS TABLE CONTAINS THE GAP SIZES (IN TENS OF MICROSECONDS) FOR EACH
 ; OF THE 16. GAPS RECORDED BY THE GAP CONSISTANCY TEST (TSTO21).
 ; NOTE: GAP #'S ARE IN OCTAL.

WORD	MAX, MIN(10)	TIME IN MS(10)	GAP #	DELAY IN MS(10)
GTIMTBL: WORD	01410. , 01290.	; 14. 1-12. 9	GAP-0	0 MS
WORD	01450. , 01350.	; 14. 5-13. 5	GAP-1	1. 0 MS
WORD	01500. , 01370.	; 15. 0-13. 7	GAP-2	2. 0 MS
WORD	01500. , 01370.	; 15. 0-13. 7	GAP-3	3. 0 MS
WORD	01500. , 01300.	; 15. 0-13. 0	GAP-4	4. 0 MS
WORD	01450. , 01200.	; 14. 5-12. 0	GAP-5	5. 0 MS
WORD	01450. , 01200.	; 14. 5-12. 0	GAP-6	6. 0 MS
WORD	01450. , 01200.	; 14. 5-12. 0	GAP-7	7. 0 MS
WORD	01370. , 01200.	; 13. 7-12. 0	GAP-10	8. 0 MS
WORD	01370. , 01200.	; 13. 7-12. 0	GAP-11	9. 0 MS
WORD	01370. , 01200.	; 13. 7-12. 0	GAP-12	10. 0 MS
WORD	01370. , 01200.	; 13. 7-12. 0	GAP-13	11. 0 MS
WORD	01370. , 01200.	; 13. 7-12. 0	GAP-14	12. 0 MS
WORD	01370. , 01200.	; 13. 7-12. 0	GAP-15	13. 1 MS
WORD	01370. , 01200.	; 13. 7-12. 0	GAP-16	14. 1 MS
WORD	01370. , 01200.	; 13. 7-12. 0	GAP-17	15. 1 MS

2271
2272
2273 001674 000000
2274 001676 015015
2275 001700 015037
2276 001702 015057
2277 001704 015101
2278 001706 015125
2279 001710 015147
2280 001712 015166
2281 001714 015210
2282 001716 015233
2283 001720 015255
2284 001722 015302
2285 001724 015331
2286 001726 015362
2287 001730 015413
2288 001732 015441
2289 001734 015470
2290 001736 015520
2291 001740 000000
2292 001742 015543
2293 001744 015567
2294 001746 015613
2295 001750 015637
2296 001752 015664
2297 001754 015706
2298 001756 015731
2299 001760 015753

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

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

; DUMMY TEST

```

2301
2302 ;CHECK SWITCH REGISTER ROUTINE. CHECKS FOR G TO ALLOW CHANGING
2303 ;OF LOC. 176.
2304 ;CALL IS BY WAY OF JSR PC,CKSWR
2305 ;LOCATIONS USED:
2306 001762 000000 TIB: .WORD 0
2307 001764 000000 TEMPST: .WORD 0
2308 001766 000000 COUNT: .WORD 0
2309 001770 000000 RDSW: .WORD 0
2310
2311 001772 022737 000176 001002 CKSWR: CMP #SWREG,SWR ;SOFTWARE SWITCH REG PRESENT
2312 002000 001120 BNE OUT ;NO, GET OUT
2313 002002 013737 177562 001762 MOV TKB,TIB ;AND STRIP OFF
2314 002010 042737 177600 001762 BIC #177600,TIB ;THE GARBAGE
2315 002016 022737 000007 001762 CMP #7,TIB ;IS IT A < G>
2316 002024 001106 BNE OUT
2317 002026 000004 015775 TYPE,L. CNTG
2318 002032 000004 016002 CNTLU: TYPE,L. SWR
2319 002036 017702 176740 MOV @SWR,R2
2320 002042 004737 002632 JSR PC, TYPOCT
2321 002046 000004 016011 TYPE,L. NEW
2322
2323 002052 005037 001764 CLR TEMPST
2324 002056 012737 000007 001766 MOV #7,COUNT
2325 002064 004737 002244 15: JSR PC,TTIN ;GO READ A CHARACTER
2326 002070 042737 177600 001762 BIC #177600,TIB ;STRIP OFF GARBAGE
2327 002076 122737 000025 001762 CMPB #25,TIB ;IS IT A U?
2328 002104 001001 BNE 25 ;BRANCH IF NOT
2329 002106 000751 35: BR CNTLU ;START OVER
2330 002110 122737 000015 001762 25: CMPB #15,TIB ;IS IT A <CR>?
2331 002116 001012 BNE 45 ;BRANCH IF NOT
2332 002120 012737 000200 001770 MOV #200,RDSW
2333 002126 004737 002362 JSR PC,TCRLF ;ECHO IT WITH <LF>
2334 002132 022737 000007 001766 CMP #7,COUNT ;WAS IT FIRST CHARACTER
2335 002140 001034 BNE 75 ;CHANGE SWR IF NOT FIRST ONE
2336 002142 000437 85: BR OUT ;GET OUT
2337 002144 122737 000060 001762 45: CMPB #60,TIB
2338 002152 003004 BGT 55
2339 002154 122737 000067 001762 CMPB #67,TIB
2340 002162 002003 BGE 65
2341 002164 000004 016021 55: TYPE,L. QUEST
2342 002170 000746 BR 35 ;START OVER IF NOT LEGAL CHARACTER
2343 002172 006337 001764 65: ASL TEMPST
2344 002176 006337 001764 ASL TEMPST
2345 002202 006337 001764 ASL TEMPST
2346 002206 142737 000060 001762 BICB #60,TIB ;GET NITTY-GRITTY
2347 002214 153737 001762 001764 BISB TIB,TEMPST
2348 002222 005337 001766 DEC COUNT ;ONLY WANT 6 DIGITS
2349 002226 001756 BEQ 55
2350 002230 000715 BR 15
2351 002232 013777 001764 176542 75: MOV TEMPST,@SWR ;CHANGE SWITCH REGISTER CONTENTS
2352 002240 000740 BR 85
2353 002242 000207 OUT: RTS PC
2354
  
```

2356
 2357
 2358
 2359 002244 005037 177560
 2360 002250 005037 177562
 2361 002254 005037 001762
 2362 002260 005237 177560
 2363 002264 105737 177560
 2364 002270 100375
 2365 002272 013737 177562 001762
 2366 002300 105737 177564
 2367 002304 100375
 2368 002306 113737 001762 177566
 2369 002314 000207
 2370
 2371
 2372
 2373
 (1)
 (1)
 (1)
 (1)
 (1)
 (1) 000011
 (1) 002316 000
 (1) 002317 002
 (1) 002320 000
 (1)
 (1) 002321 000
 (1) 002322 177564
 (1) 002324 177566
 (1) 002326 000
 (1) 002327 000
 (1) 002330 005015 000
 (1) 002334
 (1)
 (1) 002334 010046
 (1) 002336 017600 000002
 (1) 002342 062766 000002 000002
 (1) 002350 105037 002327
 (1)
 (1) 002354 105737 002327
 (1) 002360 001410
 (1) 002362 000004 002330
 (1) 002366 105737 001770
 (1) 002372 100006
 (1) 002374 005037 001770
 (1) 002400 000207
 (1) 002402 112046
 (1) 002404 001003
 (1) 002406 005726
 (1) 002410 012600
 (1) 002412 000002
 (1)
 (1) 002414 122716 000011
 (1) 002420 001445

; TTY READ SUBROUTINE*****

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

.SBTTL PROGRAM SUBROUTINES

.SBTTL TYPE SUBROUTINE

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

;; TAGS USED BY THE TYPE ROUTINE BELOW

\$HT=11 ; ; HORIZONTAL TAB
 \$NULL: .BYTE 0 ; ; CONTAINS NULL CHARACTER
 \$FILL: .BYTE 2 ; ; CONTAINS # OF FILLER CHARACTERS
 \$TPFLG: .BYTE 0 ; ; CONTAINS TELEPRINTER AVAILABLE FLAG
 ; ; 0/377 = AVAIL/NOT AVAIL
 \$TKFLG: .BYTE 0 ; ; CONTAINS KEYBOARD AVAILABLE FLAG
 \$TPS: .WORD 177564 ; ; ADDRESS OF TELEPRINTER STATUS REGISTER
 \$TPB: .WORD 177566 ; ; ADDRESS OF TELEPRINTER DATA BUFFER
 \$CHARCNT: .BYTE 0 ; ; CONTAINS # OF CHARS TYPED
 \$CNTRL0: .BYTE 0 ; ; CONTAINS CONTROL 0 CHAR (IF TYPED)
 \$CRLF: .ASCIZ <15><12>
 .EVEN

.TYPE: MOV RO, -(SP) ; ; SAVE RO
 MOV @2(SP), RO ; ; GET MESSAGE ADDRESS
 ADD #2, 2(SP) ; ; ADJUST RETURN PC
 CLR \$CNTRL0
 TYPE1: TSTB \$CNTRL0 ; ; BRANCH IF CONTROL 0 (0) WASN'T TYPED
 BEQ TYPE2
 TCRLF: TYPE, \$CRLF ; ; TYPE <CR><LF>
 TSTB RDSW
 BPL TYPE3
 CLR RDSW
 RTS PC
 TYPE2: MOVB (RO)+, -(SP) ; ; PUSH CHARACTER TO BE TYPED ONTO STACK
 BNE TYPE4 ; ; BRANCH IF NOT THE TERMINATOR
 TST (SP)+ ; ; POP TERMINATOR CHAR OFF THE STACK
 TYPE3: MOV (SP)+, RO ; ; RESTORE RO
 RTI ; ; RETURN TO CALLER
 TYPE4: CMPB # \$HT, (SP) ; ; BRANCH IF HORIZONTAL TAB <HT>
 BEQ 95

```

(1) 002422 004737 002454      JSR    PC,5$      ;;TYPE CHARACTER
(1) 002426 122726 000012    3$:    CMPB    #12,(SP)+  ;;CHECK IF CHARACTER WAS A LINE FEED
(1) 002432 001350           BNE    TYPE1      ;;BRANCH IF NOT LINE FEED
(1) 002434 013746 002316      MOV    $NULL,-(SP) ;;GET # OF FILLERS REQUIRED AND FILLER
(1)                               ;;CHARACTER.
(1)
(1) 002440 105366 000001    4$:    DECB    1(SP)    ;;DECREMENT FILLERS REQ. COUNT
(1) 002444 002770           BLT    3$         ;;BRANCH IF NO MORE FILLERS ARE REQUIRED
(1) 002446 004737 002454      JSR    PC,5$      ;;TYPE FILLER CHARACTER
(1) 002452 000772           BR     4$
(1)
(1) 002454 105777 177642    5$:    TSTB    @STPS     ;;WAIT FOR OUTPUT DEVICE
(1) 002460 100375           BPL    -4
(1) 002462 122737 000017 002327  CMPB    #17,@#$CNTRLO ;;CHECK IF CONTROL 0 WAS TYPED
(1) 002470 001403           BEQ    6$         ;;STOP TYPING MESSAGE IF 0 WAS TYPED
(1) 002472 116677 000002 177624  MOVB   2(SP),@STPB  ;;OUTPUT CHARACTER
(1) 002500 122766 000015 000002  6$:    CMPB    #15,2(SP)  ;;BRANCH IF NOT <CR>
(1) 002506 001003           BNE    7$
(1) 002510 105037 002326      CLRB   $CHARCNT   ;;CLEAR CHARACTERS TYPED COUNT
(1) 002514 000406           BR     8$
(1) 002516 122766 000012 000002  7$:    CMPB    #12,2(SP)  ;;BRANCH IF <LF> OR 'NULL'
(1) 002524 002002           BGE    8$
(1) 002526 105237 002326      INCB   $CHARCNT   ;;INCREMENT CHARACTER TYPED COUNT
(1) 002532 000207           RTS    PC
(1)
(1)                               ;;HORIZONTAL TAB <HT> PROCESSER
(1) 002534 112716 000040    9$:    MOVB   #40,(SP)  ;;LOAD 'SPACE'
(1) 002540 004737 002454    10$:   JSR    PC,5$      ;;TYPE 'SPACE'
(1) 002544 132737 000007 002326  BITB   #7,$CHARCNT ;;TYPE SPACES UNTIL A MULTIPLE
(1) 002552 001372           BNE    10$        ;;OF 8 CHARACTERS HAVE BEEN TYPED
(1) 002554 105726           TSTB   (SP)+      ;;POP SPACE
(1) 002556 000676           BR     TYPE1      ;;GET NEXT CHARACTER

2374
2375                               ;SUBROUTINE TO SAVE GENERAL REGISTERS ON THE STACK
2376                               ;CALL:  SAVE
2377 002560 010546      .SAVE:  MOV    R5,-(SP)  ;SAVE REGISTERS ON THE STACK
2378 002562 010446      MOV    R4,-(SP)
2379 002564 010346      MOV    R3,-(SP)
2380 002566 010246      MOV    R2,-(SP)
2381 002570 010146      MOV    R1,-(SP)
2382 002572 010046      MOV    R0,-(SP)
2383 002574 016646 000014  MOV    14(SP),-(SP)  ;GET RETURN PC
2384 002600 000207      RTS    PC          ;RETURN
2385
2386                               ;SUBROUTINE TO RESTORE GENERAL REGISTERS FROM THE STACK
2387                               ;CALL:  RESTORE
2388 002602 012666 000014  .RESTORE:MOV (SP)+,14(SP) ;MOVE RETURN PC
2389 002606 012600      MOV    (SP)+,R0    ;RESTORE REGISTERS
2390 002610 012601      MOV    (SP)+,R1
2391 002612 012602      MOV    (SP)+,R2
2392 002614 012603      MOV    (SP)+,R3
2393 002616 012604      MOV    (SP)+,R4
2394 002620 012605      MOV    (SP)+,R5
2395 002622 000207      RTS    PC          ;RETURN
2396
2397                               ;SUBROUTINE TO CONVERT OCTAL DATA TO ASCII

```

```

2398 ;CALL: MOV NUMBER,R2 ;MOVE NUMBER TO R2
2399 ; JSR PC,CNVOCT
2400
2401 002624 110637 001130 CNVOCT: MOVB SP,TYPFLG ;SET DO NOT TYPE FLAG
2402 002630 000402 BR CNVTO
2403
2404 .SBTTL OCTAL TO ASCII & TYPE ROUTINE
2405 ;SUBROUTINE TO CONVERT OCTAL NUMBER TO ASCII AND TYPE IT OUT
2406 ;CALL: MOV NUMBER,R2 ;PUT # IN R2
2407 ; JSR PC,TYPOCT ;CALL ROUTINE
2408
2409 002632 105037 001130 TYPOCT: CLRB @#TYPFLG ;SET TYPE FLAG
2410 002636 CNVTO: JSR PC,.SAVE ;SAVE REGISTERS ON THE STACK
(1) 002636 004737 002560 MOV #ODIGITS,R4 ;SET PTR TO OUTPUT
2411 002642 012704 001146 CLR R3 ;R3 WILL CONTAIN OCTAL DIGIT
2412 002646 005003 MOV R2,R1 ;GET # TO BE TYPED
2413 002650 010201 15: ASL R2 ;SHIFT #
2414 002652 006302 ROL R3
2415 002654 006103 MOV #6,R0 ;SET DIGIT COUNTER
2416 002656 012700 000006 BR 35
2417 002662 000404
2418
2419 002664 006302 25: ASL R2 ;SHIFT # 3 PLACES LEFT
2420 002666 006103 ROL R3
2421 002670 005301 DEC R1
2422 002672 001374 BNE 25
2423 002674 012701 000003 35: MOV #3,R1 ;SET SHIFT COUNTER
2424 002700 116324 001134 MOVB DIGTAB(R3),(R4)+ ;MOVE ASCII EQUIV TO OUTPUT
2425 002704 005003 CLR R3
2426 002706 005300 DEC R0 ;DECREMENT DIGIT COUNT
2427 002710 001365 BNE 25 ;GET NEXT DIGIT
2428 002712 105737 001130 TSTB @#TYPFLG ;BRANCH IF ASCII IS
2429 002716 001002 BNE 45 ;NOT TO BE TYPED
2430 002720 000004 001146 TYPE,ODIGITS
2431 002724 45: JSR PC,.RESTORE ;RESTORE REGISTERS FROM THE STACK
(1) 002724 004737 002602 RTS PC
2432 002730 000207
2433
2434
2435 ;SUBROUTINE TO CONVERT OCTAL DATA TO DECIMAL ASCII
2436 ;CALL: MOV NUMBER,R2 ;MOVE NUMBER TO R2
2437 ; JSR PC,CNVDEC
2438
2439 002732 110637 001130 CNVDEC: MOVB SP,@#TYPFLG ;SET DO NOT TYPE FLAG
2440 002736 000402 BR CNVTD
2441
2442 .SBTTL OCTAL TO DECIMAL & TYPE ROUTINE
2443 ;THIS ROUTINE CONVERTS AN OCTAL # TO DECIMAL ASCII AND TYPES IT OUT
2444 ;CALL: MOV NUMBER,R2 ;PUT # IN R2
2445 ; JSR PC,TYPDEC ;CALL ROUTINE
2446
2446 002740 105037 001130 TYPDEC: CLRB @#TYPFLG ;SET TYPE FLAG
2447 002744 CNVTD: JSR PC,.SAVE ;SAVE REGISTERS ON THE STACK
(1) 002744 004737 002560 CLR R0 ;R0 IS INDEX TO DECIMAL CONSTANT
2448 002750 005000 MOV #ODIGITS,R4 ;SET OUTPUT PTR
2449 002752 012704 001146 15: CLR R3 ;R3 CONTAINS DECIMAL DIGIT
2450 002756 005003

```

2451 002760 166002 003040
2452 002764 103402
2453 002766 005203
2454 002770 000773
2455 002772 066002 003040
2456 002776 116324 001134
2457 003002 062700 000002
2458 003006 005760 003040
2459 003012 001361
2460 003014 112724 000060
2461 003020 105737 001130
2462 003024 001002
2463 003026 000004 001146
2464 003032
(1) 003032 004737 002602
2465 003036 000207
2466
2467 003040 023420
2468 003042 001750
2469 003044 000144
2470 003046 000012
2471 003050 000001
2472 003052 000000
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485 003054 010246
2486 003056 010346
2487 003060 006302
2488 003062 006302
2489 003064 010203
2490 003066 000004 014775
2491 003072 016302 001420
2492 003076 004737 002740
2493 003102 000004 001407
2494 003106 016302 001422
2495 003112 004737 002740
2496 003116 000004 001414
2497 003122 000004 015005
2498 003126 013702 001014
2499 003132 004737 002740
2500 003136 000004 001376
2501 003142 012603
2502 003144 012602
2503 003146 000207
2504
2505

```

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

```

2506
2507
2508
2509
2510
2511
2512
2513 003150 010246
2514 003152 010346
2515 003154 113703 001122
2516 003160 006303
2517 003162 006303
2518 003164 000004 014775
2519 003170 016302 001574
2520 003174 004737 002740
2521 003200 000004 001407
2522 003204 016302 001576
2523 003210 004737 002740
2524 003214 000004 001414
2525 003220 000004 015005
2526 003224 013702 001014
2527 003230 004737 002740
2528 003234 000004 014464
2529 003240 113702 001122
2530 003244 004737 002632
2531 003250 000004 001376
2532 003254 012603
2533 003256 012602
2534 003260 000207
2535
2536
2537
2538
2539 003262
(1) 003262 004737 002560
2540 003266 012700 001266
2541 003272 012701 001120
2542 003276 005011
2543 003300 005061 000002
2544 003304 122710 000015
2545 003310 001414
2546 003312 112002
2547 003314 042702 177770
2548 003320 012703 000003
2549 003324 006311
2550 003326 006161 000002
2551 003332 005303
2552 003334 001373
2553 003336 050211
2554 003340 000761
2555 003342
(1) 003342 004737 002602
2556 003346 000207
2557
2558
2559

```

; THIS SUBROUTINE IS USED TO TYPE THE SPECIFIED GAP SIZES (RECORDED IN
; TST021). IT IS CALLED BY THE GAPOK ROUTINE IF THE GAP SIZE IS OUT OF
; RANGE VIA THE HLT ROUTINE (HLT+2).
; CALL:  MOV#  #GAP, GAP           ; LOAD GAP # INTO GAP
;        MOV#  #TIME, ATIME       ; LOAD ACTUAL TIME INTO ATIME
;        JSR#  PC, OUTGAP
OUTGAP:  MOV#  R2, -(SP)           ; SAVE R2 AND R3
        MOV#  R3, -(SP)
        MOV#  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
        MOV#  @#GAP, R2         ; GET GAP #
        JSR#  PC, TYOCT        ; 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>.

```

CNVTA0:  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)
15:      CMPB#  #CR, (R0)        ; <CR> TERMINATES INPUT
        BEQ#  3$
        MOV#  (R0)+, R2         ; GET 'OCTAL' DATA
        BIC#  #177770, R2       ; STRIP UNUSED BITS
        MOV#  #3, R3            ; SET SHIFT COUNT
25:      ASL#  (R1)             ; SHIFT LAST
        ROL#  2(R1)            ; OCTAL DIGIT
        DEC#  R3
        BNE#  2$
        BIS#  R2, (R1)         ; AND INSERT THIS DIGIT
        BR#  15               ; GO GET NEXT DIGIT
35:      JSR#  PC, .RESTORE     ; RESTORE REGISTERS FROM THE STACK
        RTS#  PC              ; RETURN

```

.SBTTL PUBLISH SUBROUTINE
; THE PUBLISH SUBROUTINE AVERAGES THE RECORDED TIMES FOR EACH TEST IT-


```

2560
2561
2562
2563
2564 003350
  (1) 003350 004737 002560
2565 003354 012700 001016
2566 003360 113701 001123
2567 003364 122701 000001
2568 003370 001423
2569 003372 005002
2570 003374 005003
2571 003376 122701 000020
2572 003402 001402
2573 003404 000000
2574 003406 000777
2575
2576
2577 003410 062002
2578 003412 005503
2579 003414 005301
2580 003416 001374
2581
2582 003420 012700 000004
2583 003424 006203
2584 003426 006002
2585 003430 005300
2586 003432 001374
2587 003434 010237 001014
2588
2589 003440 113700 001124
2590 003444 006300
2591 003446 016037 001674 003456
2592 003454 000004
2593 003456 000000
2594 003460 113702 001124
2595 003464 004737 003054
2596 003470 004737 002602
2597 003474 000207
2598
2599
2600
2601
2602
2603
2604 003476 010046
2605 003500 012700 001266
2606 003504 105737 177560
2607 003510 100375
2608
2609 003512 113746 177562
2610 003516 042716 000200
2611 003522 122716 000177
2612 003526 001004
2613 003530 124026
2614 003532 000004 001401

```

```

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

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

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

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

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

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

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

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

```

2615 003536 000762
2616 003540 122716 000025
2617 003544 001004
2618 003546 005726
2619 003550 000004 001376
2620 003554 000751
2621 003556 111637 001403
2622 003562 111620
2623 003564 122726 000015
2624 003570 001403
2625 003572 000004 001403
2626 003576 000742
2627 003600 000004 001376
2628 003604 012600
2629 003606 000207
2630
2631
2632 003610 113746 177562
2633 003614 042716 000200
2634 003620 122716 000017
2635 003624 001003
2636 003626 112637 002327
2637 003632 000002
2638
2639 003634 122726 000003
2640 003640 001003
2641 003642 000005
2642 003644 000137 005750
2643 003650 000002

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

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

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

```

2645
2646
2647 003652 000000
2648
2649
2650
2651
2652
2653
2654 003654 004737 001772
2655 003660 004737 002560
2656 003664 110637 001125
2657 003670 032777 020000 175104
2658 003676 001075
2659 003700 000004 014265
2660 003704 113702 001124
2661 003710 004737 002632
2662 003714 016600 000016
2663 003720 162700 000002
2664 003724 111000
2665 003726 001417
2666 003730 000004 014350
2667 003734 122700 000002
2668 003740 001005
2669 003742 004737 003150
2670 003746 000004 001376
2671 003752 000447
2672 003754 004737 003054
2673 003760 000004 001376
2674 003764 000442
2675 003766 016500 000014
2676 003772 032765 002000 000032
2677 004000 001403
2678 004002 042700 102100
2679 004006 000402
2680 004010 042700 102300
2681 004014 005700
2682 004016 001003
2683 004020 000004 014241
2684 004024 000436
2685
2686 004026 000004 014275
2687 004032 010500
2688 004034 012701 000007
2689 004040 012002
2690 004042 004737 002632
2691 004046 000004 001411
2692 004052 005301
2693 004054 001371
2694 004056 016502 000032
2695 004062 004737 002632
2696 004066 000004 001376
2697
2698 004072 032777 001000 174702 45:
2699 004100 001402
2700 004102 000004 001405

```

```

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

;ERROR SERVICE ROUTINE
;THIS ROUTINE PROCESSES TWO TYPES OF ERRORS (OUT OF RANGE AND HARDWARE)
;THE CALLS FOR AN OUT OF RANGE ERROR ARE <HLT+1>, <HLT+2> AND, FOR A
;HARDWARE ERROR THE CALL IS <HLT>.

      HLT:  JSR      PC,CKSWR          ;CHECK FOR CNTL G
            JSR      PC,SAVE          ;SAVE REGISTERS ON THE STACK
15:        MOVB     SP,@#ERFLG        ;SET ERROR FLAG
            BIT      #SW13,@SWR       ;BRANCH IF NO TYP0UT
            BNE     4$
            TYPE,E.HDR
            MOVB     @#TSTNUM,R2      ;GET TEST #
            JSR      PC,TYPOCT        ;AND TYPE IT
            MOV      16(SP),R0        ;GET RETURN PC
            SUB      #2,R0            ;NOW PC OF HLT CALL
            MOVB     (R0),R0          ;NOW HLT CALL ITSELF
            BEQ     2$                ;BRANCH IF HLT
            TYPE,E.HDR2
            CMPB     #2,R0            ;BRANCH IF NOT HLT+2
            BNE     10$
            JSR      PC,OUTGAP        ;TYPE GAP SPECIFIED TIMES
            TYPE,CRLF
            BR      4$
10$:       JSR      PC,OUTSPC         ;TYPE SPECIFIED TIMES
            TYPE,CRLF
            BR      4$
2$:        MOV      ER(R5),R0
            BIT      #PE1600,TC(R5)
            BEQ     20$
            BIC     #102100,R0
            BR      21$
20$:       BIC     #102300,R0
21$:       TST      R0
            BNE     22$
            TYPE,E.SFT                ;TYPE SOFT ERROR MESSAGE
            BR      6$
22$:       TYPE,E.HDR1
            MOV      R5,R0            ;GET FIRST ADDRESS OF REGS.
            MOV      #7,R1           ;TYPE FIRST 7 REGS.
3$:        MOV      (R0)+,R2         ;GET REG CONTENTS
            JSR      PC,TYPOCT        ;AND TYPE IT
            TYPE,SPACE2
            DEC      R1
            BNE     3$
            MOV      TC(R5),R2       ;GET CONTENTS OF TC REGISTER
            JSR      PC,TYPOCT
            TYPE,CRLF
4$:        BIT      #SW09,@SWR       ;BRANCH IF NO RING THE BELL
            BEQ     5$
            TYPE,BELL

```



```

2714          .SBTTL          SCOPE SUBROUTINE
2715          ;SCOPE ROUTINE
2716          ;THIS ROUTINE IS ENTERED UPON COMPLETION OF EACH SUBTEST
2717          ;THE SCOPE ROUTINE:
2718          ;   OUTPUTS TIME SPEC ON EACH ITERATION IF SW08 IS SET
2719          ;   REPEATS TEST IF SW14 IS SET
2720          ;   STORES ACTUAL TIME FOR FUNCTION IN TIME TABLE (ATIMTBL)
2721          ;   PUBLISHES TIME IF SW10=0
2722          ;   UPDATES ITERATION COUNT AND IF ITERATIONS COMPLETE CONTINUES
2723          ;   TO NEXT TEST, OTHERWISE REPEATS TEST.
2724          ;   DELAYS BEFORE CONTINUING OR REPEATING TEST.
2725          ;   INITIALIZES DRIVE
2726          ;RETURNS:      R5=BASE ADDRESS OF TMO2 REGISTERS-(ADDRESS OF CS1)
2727          ;              R1='DS' REG ADDRESS
2728          ;              R0='FC' REG ADDRESS
2729
2730          SCOPE: JSR      PC,CKSWR          ;CHECK FOR CNTL G
2731                  MOV      @#TMBASE,R5      ;SET R5 TO FIRST TM REG
2732                  BIT      #SW08,@SWR      ;BRANCH IF SPECIFICATION LINE
2733                  BEQ      10$              ;NOT DESIRED ON EACH ITERATION
2734                  MOVB    @#TSTNUM,R2      ;GET TEST NUMBER
2735                  JSR      PC,OUTSPC       ;OUTPUT TIME RECORDED
2736                  BIT      #SW14,@SWR      ;BRANCH IF CONTINUOUS LOOP
2737                  BEQ      2$              ;NOT DESIRED
2738                  JSR      PC,DELAY        ;DELAY 350 MS
2739                  JSR      PC,RHINIT       ;INIT
2740                  CLRB    @#ERFLG         ;CLEAR ERROR FLAG
2741                  MOV      SCPADR,(SP)
2742                  MOV      R5,R1
2743                  ADD      #DS,R1          ;ADDRESS OF 'DS' REG IS IN R1
2744                  MOV      R5,R0
2745                  ADD      #FC,R0          ;ADDRESS OF 'FC' REG IS IN R0
2746                  RTI
2747
2748          2$:  TSTB      @#ERFLG           ;BRANCH IF ERROR FLAG IS SET
2749                  BNE      3$
2750                  MOVB    @#ITCNT,R0      ;GET ITERATION COUNT
2751                  ASL      R0              ;STORE TIME IN TABLE
2752                  MOV      @#ATIME,ATIMTBL(R0)
2753          3$:  INCB      @#ITCNT           ;INCREMENT ITERATION COUNT
2754                  TST      @#PASS        ;1ST PASS?
2755                  BEQ      4$              ;YES
2756                  BIT      #SW11,@SWR      ;BRANCH IF SINGLE ITERATION DESIRED
2757                  BNE      4$
2758                  CMPB    #16,@#ITCNT     ;BRANCH IF ITERATIONS INCOMPLETE
2759                  BNE      1$
2760          4$:  MOV      (SP),@#SCPADR      ;SET SCOPE ADDRESS TO NEXT TEST
2761                  BIT      #SW10,@SWR      ;BRANCH IF NO PUBLICATION DESIRED
2762                  BNE      5$
2763                  JSR      PC,PUBLISH      ;GO PUBLISH TEST DATA
2764          5$:  CLRB      @#ITCNT           ;RESET ITERATION COUNT
2765                  TSTB    @SWR            ;BRANCH IF USER DOES NOT WANT TO
2766                  BEQ      1$              ;HALT ON A SELECTED TEST
2767                  MOV      @SWR,-(SP)      ;GET SWITCHES
2768                  BIC      #177740,(SP)    ;CLEAR ALL BUT TEST #
2769                  DEC      (SP)            ;FORM TEST # -1

```

```

2770 004362 123726 001124      CMPB  @#TSTNUM, (SP)+      ; BRANCH IF NOT AT TEST
2771 004366 001306              BNE   1$
2772 004370 000000              HALT
2773 004372 004737 001772      JSR   PC, CKSWR          ; CHECK FOR CNTL G
2774 004376 000702              BR    1$
2775
2776                      .SBTTL  TIMER SUBROUTINES
2777
2778                      ; SUBROUTINE TO SYNCHRONIZE THE TIMER AND TURN IT ON.
2779                      ; REGISTER 4 IS CLEARED, AND THE OSCILLATOR POLARITY IS MONITORED
2780                      ; THE ROUTINE IS EXITED WHEN THE OSCILLATOR POLARITY CHANGES WITH R3
2781                      ; SET TO INDICATE THE POLARITY OF THE OSCILLATOR.
2782                      ; CALL:  JSR   PC, TIMON
2783                      ; RETURNS: R3 SET TO INDICATE LAST POLARITY (+24/-24=0/1)
2784                      ;
2785                      ;
2786 004400 005004              TIMON: CLR   R4                ; CLEAR TIME COUNT
2787 004402 012703 000024      MOV   #24, R3           ; SET POLARITY TO '0' STATE
2788 004406 032765 000100 000024  BIT   #OSC, MR(R5)     ; BRANCH IF POLARITY IS '0'
2789 004414 001405              BEQ   2$
2790 004416 032765 000100 000024 1$: BIT   #OSC, MR(R5)     ; WAIT FOR OSCILLATOR TO RETURN
2791 004424 001374              BNE   1$
2792 004426 000405              BR    4$
2793
2794 004430 005403              2$:  NEG   R3                ; NEGATE PREV POLARITY INDICATOR
2795 004432 032765 000100 000024 3$:  BIT   #OSC, MR(R5)     ; WAIT FOR OSCILLATOR TO RETURN
2796 004440 001774              BEQ   3$                ; TO '1' STATE
2797 004442 000207              4$:  RTS   PC
2798
2799                      ; SUBROUTINE TO COUNT TIME
2800                      ; EACH TIME THE OSCILLATOR TOGGLES (BIT <06> IN MR REG) REGISTER
2801                      ; R4 IS INCREMENTED, AND THE REGISTER R3 IS NEGATED TO INDICATE
2802                      ; THE LAST STATE OF THE OSCILLATOR.
2803                      ; CALL  JMP   TIMER(R3)          ; R3 IS SET BY TIMON ROUTINE
2804                      ;      R2=RETURN ADDRESS TO CALLER
2805                      ; NOTE: THE TIME TO EXECUTE THIS ROUTINE IS VERY CRITICAL. IT MUST BE
2806                      ; LESS THAN 40 US.
2807
2808                      ; ENTER HERE VIA JMP  TIMER(R3) WHEN R3=-24 (PREV STATE=1)
2809 004444 032765 000100 000024  TIMER1: BIT   #OSC, MR(R5)     ; BRANCH IF CURRENT STATE IS '0'
2810 004452 001406              BEQ   TIMER            ; GO INCREMENT TIME
2811 004454 000112              JMP   (R2)             ; RETURN TO TEST
2812
2813                      . =TIMER1+24
2814 004470 005403              TIMER: NEG   R3                ; NEGATE PREV STATE INDICATOR
2815 004472 005204              INC   R4                ; INCREMENT 'TICK' COUNT
2816 004474 100401              BMI   TIMERR           ; BRANCH ON OVERFLOW
2817 004476 000112              JMP   (R2)             ; RETURN TO TEST
2818 004500 000004 014376      TIMERR: TYPE, E. TIMOV    ; TYPE 'TIMER OVERFLOWED'
2819 004504 104400              HLT                    ; REPORT HARDWARE ERROR
2820 004506 000177 174272      JMP   @SCPADR          ; RETURN TO BEGINNING OF TEST
2821
2822                      . =TIMER+24
2823                      ; ENTER HERE VIA JMP  TIMER(R3) WHEN R3=+24 (PREV STATE=0)
2824 004514 032765 000100 000024  TIMERO: BIT   #OSC, MR(R5)     ; BRANCH IF CURRENT STATE = '1'
2825 004522 001362              BNE   TIMER

```

```

2826 004524 000112          JMP      (R2)
2827
2828 ; SUBROUTINE TO CHECK TIME RECORDED BY SUBTEST.
2829 ; THIS SUBROUTINE COMPUTES THE ACTUAL TIME (IN MICROSECONDS) AND CHECKS
2830 ; THAT THE TIME RECORDED BY THE SUBTEST IS CORRECT BY COMPARING THE TIME
2831 ; WITH THE HIGH LIMIT (STIMTBL(R0)) AND THE LOW LIMIT (STIMTBL+2(R0)).
2832 ; IF THE TIME IS OUT OF RANGE AN OUT OF RANGE ERROR TYPEOUT RESULTS.
2833 ; THE SUBROUTINE IS ENTERED WITH:
2834 ;     R4=TICK COUNT
2835
2836 004526          TIMOK:
   (1) 004526 004737 002560      JSR      PC, SAVE          ; SAVE REGISTERS ON THE STACK
2837 004532 012700 000070      MOV      #56, R0         ; GET TIME PER TICK
2838 004536 010401          MOV      R4, R1         ; GET TICKS COUNT
2839 004540 005002          CLR      R2            ; CLEAR SUMMING REGISTERS
2840 004542 005003          CLR      R3
2841 004544 060002      1$:  ADD      R0, R2         ; MULTIPLY TIME PER TICK
2842 004546 005503          ADC      R3            ; BY TICK COUNT
2843 004550 005301          DEC      R1
2844 004552 001374          BNE      1$
2845 004554 010246          MOV      R2, -(SP)      ; DIVIDE COUNT BY 10.
2846
2847 004556 010346          MOV      R3, -(SP)
2848 004560 012746 000012      MOV      #10, -(SP)
2849 004564 004737 005052      JSR      PC, DIVIDE
2850 004570 005726          TST      (SP)+         ; DISCARD REMAINDER
2851 004572 012637 001014      MOV      (SP)+, @#ATIME ; STORE QUOTIENT
2852 004576 113700 001124      MOVB    @#TSTNUM, R0   ; GET TEST #
2853 004602 006300          ASL      R0
2854 004604 006300          ASL      R0
2855 004606 023760 001014 001420  CMP      @#ATIME, STIMTBL(R0) ; CHECK THAT TIME IS WITHIN
2856 004614 101004          BHI      2$            ; LIMITS SPECIFIED
2857 004616 023760 001014 001422  CMP      @#ATIME, STIMTBL+2(R0)
2858 004624 101001          BHI      3$
2859 004626 104401      2$:  HLT+1          ; CALL ERROR ROUTINE
2860 004630      3$:
   (1) 004630 004737 002602      JSR      PC, RESTORE     ; RESTORE REGISTERS FROM THE STACK
2861 004634 000207      RTS      PC            ; RETURN
2862
2863 ; SUBROUTINE TO CHECK INDIVIDUAL GAP TIMES (PRODUCED BY TSTO21)
2864 ; SUBROUTINE COMPUTES THE ACTUAL TIME (IN MICROSECONDS) AND CHECKS
2865 ; THAT THE GAP TIME RECORDED BY THE SUBTEST (TSTO21) BY COMPARING THE
2866 ; TIME WITH THE MAX LIMIT (GTIMTBL-GAPTBL(R1)) AND THE MIN LIMIT
2867 ; (GTIMTBL+2-GAPTBL(R1)).
2868 ; CALL:  MOV      #TICK COUNT, R4      ; R4 CONTAINS TICK COUNT
2869 ;        MOVB    #GAP, @#GAP         ; LOCATION GAP CONTAINS GAP #
2870 ;        JSR      PC, GAPOK
2871
2872 004636          GAPOK:
   (1) 004636 004737 002560      JSR      PC, SAVE          ; SAVE REGISTERS ON THE STACK
2873 004642 012700 000070      MOV      #56, R0         ; GET TIME PER TICK
2874 004646 010401          MOV      R4, R1         ; GET TICK COUNT
2875 004650 005002          CLR      R2            ; CLEAR SUMMING REGISTERS
2876 004652 005003          CLR      R3
2877 004654 060002      1$:  ADD      R0, R2         ; MULTIPLY TICK COUNT
2878 004656 005503          ADC      R3            ; BY TIME PER TICK

```

```

2879 004660 005301      DEC      R1
2880 004662 001374      BNE      1$
2881
2882 004664 010246      MOV      R2,-(SP)      ;DIVIDE TIME BY 10.
2883 004666 010346      MOV      R3,-(SP)
2884 004670 012746 000012  MOV      #10,-(SP)
2885 004674 004737 005052  JSR      PC,DIVIDE
2886 004700 005726      TST      (SP)+      ;DISCARD REMAINDER
2887 004702 012637 001014  MOV      (SP)+,@#ATIME ;STORE QUOTIENT
2888 004706 113703 001122  MOVB    @#GAP,R3      ;GET GAP #
2889 004712 006303      ASL      R3          ;MULTPLY BY 4
2890 004714 006303      ASL      R3          ;TO GET AT TABLE ENTRY
2891 004716 023763 001014 001574  CMP      @#ATIME,GTIMTBL(R3) ;CHECK TIME (MAX)
2892 004724 101004      BHI      2$
2893 004726 023763 001014 001576  CMP      @#ATIME,GTIMTBL+2(R3) ;CHECK TIME (MIN)
2894 004734 101002      BHI      3$
2895 004736 104402      HLT+2      ;REPORT OUT-OF RANGE ERROR
2896 004740 000406      BR       100$
2897 004742 032777 000400 174032 3$:  BIT      #SW08,@SWR    ;BRANCH IF TIMES NOT WANTED
2898 004750 001402      BEQ      100$
2899 004752 004737 003150      JSR      PC,OUTGAP    ;TYPE GAP TIMES
2900
2901 004756      100$:
(1) 004756 004737 002602      JSR      PC,RESTORE   ;RESTORE REGISTERS FROM THE STACK
2902 004762 000207      RTS      PC          ;RETURN TO TEST
2903
2904      .SBTTL      DELAY SUBROUTINES
2905      ;THIS SUBROUTINE CAUSES A DELAY OF 350 MS.
2906 004764 004737 004400  DELAY: JSR      PC,TIMON
2907 004770 010246      MOV      R2,-(SP)    ;SAVE R2 ON THE STACK
2908 004772 012702 005002  MOV      #2$,R2      ;SET RETURN ADDRESS FOR TIMER
2909 004776      1$:
(1) 004776 000163 004470      JMP      TIMER(R3)   ;GO TO TIMER & RETURN VIA R2
2910 005002 032704 004000  2$:  BIT      #4000,R4
2911 005006 001773      BEQ      1$
2912 005010 012602      MOV      (SP)+,R2    ;RESTORE R2
2913 005012 000207      RTS      PC
2914
2915      ;THIS SUBROUTINE ALLOWS A CALLER SPECIFIED DELAY (UP TO 65MS.)
2916      ;CALL: MOV      DELAY TIME,DELTIM    ;LOAD DELAY TIME (IN US)
2917      ;      JSR      PC,DELAYV
2918 005014 005737 001116  DELAYV: TST      DELTIM    ;BRANCH IF 0 DELAY
2919 005020 001413      BEQ      3$
2920 005022 004737 004400      JSR      PC,TIMON    ;TURN TIMER ON
2921 005026 010246      MOV      R2,-(SP)    ;SAVE R2 ON THE STACK
2922 005030 012702 005040  MOV      #2$,R2      ;SET RETURN ADDRESS FROM TIMER
2923 005034      1$:
(1) 005034 000163 004470      JMP      TIMER(R3)   ;GO TO TIMER & RETURN VIA R2
2924 005040 023704 001116  2$:  CMP      @#DELTIM,R4
2925 005044 101373      BHI      1$
2926 005046 012602      MOV      (SP)+,R2    ;RESTORE R2
2927 005050 000207  3$:  RTS      PC
2928
2929      .SBTTL      DIVIDE SUBROUTINE
2930      ;THIS SUBROUTINE DIVIDES A DOUBLE PRECISION # AND RETURNS THE RESULT
2931      ;TO THE CALLER ON THE STACK. BOTH DIVIDEND & DIVISOR MUST BE POSITIVE.

```



```

2932 ;CALL: MOV LEAST SIGNIFICANT HALF DIVIDEND,-(SP)
2933 ; MOV #MOST SIGNIFICANT HALF DIVIDEND,-(SP)
2934 ; MOV #DIVISOR,-(SP)
2935 ; JSR PC,DIVIDE
2936 ;RETURN
2937 ; (SP)=REMAINDER ON STACK
2938 ; 2(SP)=QUOTIENT
2939
2940 ;NOTE: THIS SUBROUTINE DESTROYS PREVIOUS CONTENTS OF R0,R1,R2 & R3.
2941
2942 005052 005046 DIVIDE: CLR -(SP) ;SAVE LOC FOR SIGNS
2943 005054 012746 000021 MOV #17,-(SP) ;SET ITERATION COUNT
2944 005060 016601 000012 MOV 12(SP),R1 ;GET LSH DIVIDEND
2945 005064 016600 000010 MOV 10(SP),R0 ;GET MSH DIVIDEND
2946 005070 016602 000006 MOV 6(SP),R2 ;GET DIVISOR
2947 005074 005402 NEG R2 ;NEGATE DIVISOR
2948 005076 000241 CLC ;CLEAR 'C' BIT IN PSW
2949 005100 000405 BR 2$
2950 005102 006100 1$: ROL R0 ;ROTATE MSH DIVIDEND
2951 005104 010003 MOV R0,R3 ;SAVE IN R3
2952 005106 060203 ADD R2,R3 ;SUBTRACT DIVISOR FROM MSH DIVIDEND
2953 005110 103001 BCC 2$ ;BRANCH IF DIVIDEND > DIVISOR
2954 005112 010300 MOV R3,R0 ;SAVE REMAINDER IN R0
2955 005114 006101 2$: ROL R1 ;ROTATE LSH DIVIDEND
2956 005116 005316 DEC (SP) ;DECREMENT ITERATION COUNT
2957 005120 001370 BNE 1$
2958 005122 005726 TST (SP)+ ;POP ITERATION COUNTER
2959 005124 005726 TST (SP)+ ;POP SIGN CORRECTION
2960 005126 010166 000006 MOV R1,6(SP) ;PUSH REMAINDER ON STACK
2961 005132 010066 000004 MOV R0,4(SP) ;PUSH QUOTIENT ONTO STACK
2962 005136 012616 MOV (SP)+,(SP)
2963 005140 000207 RTS PC
  
```

```

2965          .SBTTL  DRIVE SUBROUTINES
2966          ;SUBROUTINE TO CHECK IF DRIVE IS AVAILABLE
2967          ;CALL:  MOVB  DRIVE#,DRVNUM
2968          ;       JSR   PC,DRVAVA
2969          ;RETURN: 'C' BIT SET IF NOT AVAILABLE
2970 005142 113765 001006 000010 DRVAVA: MOVB  @#DRVNUM,CS2(R5) ;LOAD DRIVE #
2971 005150 032765 040000 000026        BIT   #TAP,DT(R5) ;CHECK IF TAPE UNIT
2972 005156 001003                BNE   1$
2973 005160 004737 005220        JSR   PC,RHINIT
2974 005164 000262                SEV                   ;SET 'V' TO IND NOT AVAIL
2975 005166 000207 1$:          RTS   PC ;RETURN
2976
2977          ;SUBROUTINE TO CHECK IF TU16 SLAVE IS AVAILABLE FOR TEST
2978          ;CALL:  MOVB  DRIVE #,@#DRVNUM ;PASS DRIVE # VIA DRVNUM
2979          ;       MOVB  SLAVE #,@#SLVNUM ;PASS SLAVE # VIA SLVNUM
2980          ;       JSR   PC,SLVAVA ;CALL SUBROUTINE
2981 005170 113765 001006 000010 SLVAVA: MOVB  @#DRVNUM,CS2(R5) ;LOAD DRIVE #
2982 005176 113765 001007 000032        MOVB  @#SLVNUM,TC(R5) ;AND SLAVE #
2983 005204 032765 002000 000026        BIT   #SPR,DT(R5) ;BRANCH IF SLAVE PRESENT
2984 005212 001001                BNE   1$
2985 005214 000262                SEV                   ;SET 'V' TO INDICATE NO SLAVE
2986 005216 000207 1$:          RTS   PC
2987
2988          ;SUBROUTINE TO INITIALIZE RH CONTROLLER
2989          ;CALL:  JSR PC,RHINIT
2990
2991 005220 012765 000040 000010 RHINIT: MOV   #40,CS2(R5)
2992 005226 113765 001006 000010        MOVB  @#DRVNUM,CS2(R5)
2993 005234 005046                CLR   -(SP)
2994 005236 113716 001007        MOVB  @#SLVNUM,(SP)
2995 005242 012665 000032        MOV   (SP)+,TC(R5) ;LOAD SLAVE # INTO TC REG
2996 005246 052765 000300 000032        BIS   #NORM11,TC(R5)
2997 005254 000207                RTS   PC
2998
2999          ;SUBROUTINE TO WAIT FOR DRIVE READY (DRY)
3000 005256 005027 WAITRDY: CLR   (PC)+ ;CLEAR WAIT TIMER
3001 005260 000000 WAITTIM: .WORD 0
3002 005262 005765 000012 1$:    TSTB  DS(R5) ;WAIT FOR READY TO SET
3003 005266 100406                BMI   2$
3004 005270 005237 005260        INC   WAITTIM ;INCREMENT WAIT TIMER
3005 005274 001372                BNE   1$ ;BRANCH IF TIME HAS NOT EXPIRED
3006 005276 000004 014423        TYPE,E.TIMEXP ;TYPE 'TIME EXPIRED WAITING FOR RDY'
3007 005302 000425                BR    99$ ;TAKE ERROR EXIT
3008 005304 032765 002000 000012 2$:    BIT   #EOT,DS(R5) ;CHECK FOR END OF TAPE
3009 005312 001415                BEQ   3$ ;BRANCH IF NO EOT
3010 005314 000004 013454        TYPE,M.NAM
3011 005320 000004 014002        TYPE,M.EOT ;TYPE 'END OF TAPE'
3012 005324 004737 005362        JSR   PC,.REWIND ;REWIND SLAVE
3013 (1) 005330 102412                BVS   99$ ;BRANCH IF ERROR ON REWIND
3014 005332 004737 005444        JSR   PC,WRITE ;WRITE A RECORD
3015 005336 005215                INC   (R5) ;SET 'GO' BIT
3016 005340 004737 005256        JSR   PC,WAITRDY ;WAIT FOR READY
3017 005344 000404                BR    99$ ;TAKE ERROR EXIT
3018 005346 032765 040000 000012 3$:    BIT   #ERR,DS(R5) ;CHECK ERROR EXIT
3019 005354 001401                BEQ   100$
3019 005356 000262 99$:        SEV

```

```

3020 005360 000207      100$: RTS      PC
3021
3022      ;SUBROUTINE TO REWIND A UNIT (DRIVE/SLAVE COMBINATION)
3023      ;CALL  MOVB  DRIVE #, @#DRVNUM
3024      ;      MOVB  SLAVE #, @#SLVNUM
3025      ;      JSR   PC, .REWIND
3026      ;SUBROUTINE RETURNS TO CALLER WITH SELECTED SLAVE AT 'BOT', & 'V' SET IF
3027      ;AN ERROR OCCURS.
3028
3029 005362 004737 005220      .REWIND: JSR   PC, RHINIT      ; INITIALIZE CONTROLLER
3030 005366 004337 005600      JSR   R3, TMCMD      ; GO TO TM COMMAND SUBROUTINE
3031 005372 000000      .WORD  0      ; BUS ADDRESS (NOT USED)
3032 005374 000000      .WORD  0      ; WORD COUNT (NOT USED)
3033 005376 000000      .WORD  0      ; FRAME COUNT (NOT USED)
3034 005400 000006      .WORD  RWD      ; REWIND COMMAND
3035 005402 005215      INC   (R5)      ; SET 'GO' BIT
3036 005404 032765 000002 000012 1$: BIT   #BOT, DS(R5)      ; BRANCH IF 'BOT' SET
3037 005412 001005      BNE   2$
3038 005414 032765 040000 000012      BIT   #ERR, DS(R5)      ; CHECK ERROR BIT
3039 005422 001006      BNE   99$      ; BRANCH IF ERROR BIT SET
3040 005424 000767      BR    1$
3041
3042 005426 032765 020000 000012 2$: BIT   #PIP, DS(R5)      ; WAIT FOR TAPE MOTION TO STOP
3043 005434 001374      BNE   2$
3044 005436 000401      BR    100$
3045 005440 000262      99$: SEV
3046 005442 000207      100$: RTS      PC
3047
3048      ;SUBROUTINE TO WRITE 256. WORD RECORD
3049      ;CALL: JSR   PC, WRITE
3050
3051 005444 004337 005600      WRITE: JSR   R3, TMCMD      ; GO TO TM COMMAND SUBROUTINE
3052 005450 016030      .WORD  WTBUF      ; BUS ADDRESS
3053 005452 177600      .WORD  WRDCNT      ; WORD COUNT
3054 005454 177400      .WORD  FRMCNT      ; FRAME COUNT
3055 005456 000060      .WORD  WFWD      ; WRITE FORWARD COMMAND
3056 005460 000207      RTS      PC
3057
3058      ;SUBROUTINE TO READ A 256. WORD RECORD.
3059      ;CALL: JSR   PC, READ
3060
3061 005462 004337 005600      READ: JSR   R3, @#TMCMD
3062 005466 016030      .WORD  RDBUF      ; ADDRESS OF READ BUFFER
3063 005470 177600      .WORD  WRDCNT      ; 2'S COMPLEMENT OF WORD COUNT
3064 005472 177400      .WORD  FRMCNT      ; 2'S COMPLEMENT OF FRAME COUNT
3065 005474 000070      .WORD  RDFWD      ; READ FORWARD COMMAND
3066 005476 000207      RTS      PC
3067
3068      ;SUBROUTINE TO INITIATE READ REVERSE COMMAND
3069      ;CALL: JSR   PC, REVRD
3070
3071 005500 004337 005600      REVRD: JSR   R3, TMCMD
3072 005504 016430      .WORD  RDBUF+256.      ; ADDRESS OF READ REVERSE BUFFER
3073 005506 177600      .WORD  WRDCNT      ; 2'S COMPLEMENT OF WORD COUNT
3074 005510 177400      .WORD  FRMCNT      ; 2'S COMPLEMENT OF FRAME COUNT
3075 005512 000076      .WORD  RDREV      ; READ REVERSE COMMAND
  
```

```

3076 005514 000207          RTS      PC
3077
3078          ; SUBROUTINE TO SPACE FORWARD 1 RECORD
3079 005516 012765 177777 000006 FWDSPC: MOV      #-1,FC(R5)      ; LOAD RECORD COUNT
3080 005524 012715 000031          MOV      #SPCFWD+1,(R5)    ; LOAD COMMAND
3081 005530 004737 005256          JSR      PC,WAITRDY        ; WAIT FOR READY
3082 005534 000207          RTS      PC              ; RETURN
3083
3084          ; SUBROUTINE TO WRITE A RECORD AND BACK SPACE OVER THE RECORD.
3085 005536 004737 005444 WRT. BK: JSR      PC,WRITE      ; WRITE THE RECORD
3086 005542 005215          INC      (R5)              ; SET 'GO' BIT
3087 005544 004737 005256          JSR      PC,WAITRDY
3088 005550 102412          BVS     2$
3089 005552 012765 177777 000006          MOV      #-1,FC(R5)      ; LOAD RECORD COUNT
3090 005560 012715 000033          MOV      #SPCREV+1,(R5)  ; LOAD COMMAND
3091 005564 004737 005256          JSR      PC,WAITRDY
3092 005570 102402          BVS     2$
3093 005572 004737 004764          1$: JSR      PC,DELAY        ; WAIT FOR TAPE MOTION TO STOP
3094 005576 000207          2$: RTS      PC
3095
3096          ; SUBROUTINE TO LOAD A COMMAND
3097          ; CALL: JSR      R3,TMCMD
3098          ;          . WORD  BUS ADDRESS
3099          ;          . WORD  WORD COUNT (2'S COMPLEMENT)
3100          ;          . WORD  FRAME COUNT (2'S COMPLEMENT)
3101          ;          . WORD  COMMAND
3102
3103 005600 012365 000004 TMCMD: MOV      (R3)+,BA(R5)    ; LOAD BUS ADDRESS
3104 005604 012365 000002          MOV      (R3)+,WC(R5)    ; LOAD WORD COUNT
3105 005610 012365 000006          MOV      (R3)+,FC(R5)    ; LOAD FRAME COUNT
3106 005614 012315          MOV      (R3)+,(R5)      ; LOAD COMMAND
3107 005616 000203          RTS      R3              ; RETURN
3108
3109          ; SUBROUTINE TO PRINT TU16 SERIAL NUMBER
3110          ; JSR      PC,SNPT
3111
3112 005620 016503 000030 SNPT:  MOV      SN(R5),R3
3113 005624 012701 001146          MOV      #0DIGITS,R1
3114 005630 000303          SWAB   R3
3115 005632 006003          ROR    R3
3116 005634 006003          ROR    R3
3117 005636 006003          ROR    R3
3118 005640 006003          ROR    R3          ; GET FIRST DIGIT
3119 005642 042703 177760          BIC    #177760,R3
3120 005646 052703 000260          BIS    #260,R3
3121 005652 110321          MOVB   R3,(R1)+      ; FILL FIRST DIGIT
3122 005654 016503 000030          MOV      SN(R5),R3
3123 005660 000303          SWAB   R3
3124 005662 042703 177760          BIC    #177760,R3
3125 005666 052703 000260          BIS    #260,R3
3126 005672 110321          MOVB   R3,(R1)+      ; GET SECOND DIGIT
3127 005674 016503 000030          MOV      SN(R5),R3
3128 005700 006003          ROR    R3
3129 005702 006003          ROR    R3
3130 005704 006003          ROR    R3
3131 005706 006003          ROR    R3
  
```

```

3132 005710 042703 177760      BIC      #177760,R3
3133 005714 052703 000260      BIS      #260,R3
3134 005720 110321                MOV      R3,(R1)+      ;GET THIRD DIGIT
3135 005722 016503 000030      MOV      SN(R5),R3
3136 005726 042703 177760      BIC      #177760,R3
3137 005732 052703 000260      BIS      #260,R3
3138 005736 110321                MOV      R3,(R1)+      ;GET FOURTH DIGIT
3139 005740 105011                CLRB     (R1)
3140 005742 000004 001146      TYPE,ODIGITS          ;TYPE SERIAL NUMBER
3141 005746 000207                RTS      PC              ;RETURN
3142
3143
3144 005750 012706 000600      INIT:  MOV      #STKPTR,SP          ;SET STACK PTR
3145
3146 005754 013746 000006      SUSWR: MOV      @#6,-(SP)          ;SAVE VECTORS
3147 005760 013746 000004      MOV      @#4,-(SP)
3148 005764 012737 006004 000004      MOV      #61$,@#4          ;SET UP FOR TIMEOUT
3149 005772 022777 177777 173002      CMP      #-1,@SWR          ;REFERENCE HARDWARE SWITCH REGISTER
3150 006000 001402                BEQ      60$
3151 006002 000404                BR       62$
3152 006004 022626      61$:  CMP      (SP)+,(SP)+          ;ADJUST STACK
3153 006006 012737 000176 001002      60$:  MOV      #SWREG,SWR          ;POINT TO SOFTWARE SWITCH REG
3154 006014 012637 000004      62$:  MOV      (SP)+,@#4          ;RESTORE VECTORS
3155 006020 012637 000006      MOV      (SP)+,@#6
3156 006024 022737 000176 001002      CMP      #SWREG,@#SWR
3157 006032 001002                BNE      64$
3158 006034 004737 002032      JSR      PC,CNTLU
3159 006040 105037 001126      64$:  CLRB     @#PRGFLG          ;CLEAR PROGRAM FLAG
3160 006044 105037 001123      CLRB     @#ITCNT          ;CLEAR ITERATION COUNT
3161 006050 105037 001124      CLRB     @#TSTNUM          ;SET TEST # 0
3162 006054 105037 001125      CLRB     @#ERFLG          ;CLEAR ERROR FLAG
3163 006060 105037 001132      CLRB     ASFLG            ;CLEAR ASK FLAG
3164 006064 012737 000006 000004      MOV      #ERRVEC+2,@#ERRVEC
3165 006072 012737 000002 000006      MOV      #RT1,@#ERRVEC+2  ;CHECK IF 'LP' IS AVAILABLE
3166 006100 005037 001266      2$:   CLR      @#INBUF
3167 006104 023737 000042 000046      CMP      @#42,@#46          ;ACT11 AUTO MODE?
3168 006112 001415                BEQ      5$                ;YES.NO TITLE, DEFAULT ADDR.
3169 006114 000004 001376      TYPE,CRLF
3170 006120 000004 013454      TYPE,M.NAM          ;TYPE TITLE
3171 006124 000004 013522      TYPE,I.REG          ;ASK USER TO TYPE CONT BASE ADRS
3172 006130 004737 003476      JSR      PC,INPUT          ;GET USER INPUT
3173 006134 004737 003262      4$:   JSR      PC,CNVTAO          ;CONVERT ASCII TO OCTAL
3174 006140 013737 001120 001012      MOV      @#OCTALO,@#TMBASE ;SET NEW ADDRESS
3175 006146 013705 001012      5$:   MOV      @#TMBASE,R5
3176
3177      ;ROUTINE TO CHECK IF CONTROLLER (RH11) IS AVAILAABLE
3178 006152 000261                SEC
3179 006154 005715                TST      (R5)          ;SET 'C' IN PSW
3180 006156 103003                BCC      6$            ;BRANCH IF CONTROLLER AVAIL
3181 006160 000004 014041      TYPE,E.NCON
3182 006164 000671                BR       INIT
3183 006166 012737 003652 000004      6$:   MOV      #ERRTRP,@#ERRVEC ;SET ERROR TRAP VECTOR

```

```
3185 ;ROUTINE TO GET TMO2 DRIVES USER DESIRES TO TEST
3186 006174 105037 001125 DRIVES: CLR B @#ERFLG ;CLEAR ERROR FLAG
3187 006200 012701 001156 MOV #DRVTBL,R1 ;MARK ALL DRIVES AS NOT TO
3188 006204 012700 000004 MOV #4,RO ;BE TESTED. A '0' INDICATES
3189 006210 005021 15: CLR (R1)+ ;THAT A DRIVE IS NOT TO BE
3190 006212 005300 DEC RO ;TESTED
3191 006214 001375 BNE 15
3192 006216 023737 000042 000046 CMP @#42,@#46 ;ACT11 AUTO MODE?
3193 006224 001411 BEQ 1005 ;YES. TEST ALL DRIVES.
3194 006226 000004 013567 TYPE,I,DRVS
3195 006232 004737 003476 JSR PC, INPUT ;GET USER INPUT
3196 006236 012700 001266 MOV #INBUF,RO
3197 006242 122710 000101 CMPB #'A,(RO) ;AN 'A' SPECIFIES ALL
3198 006246 001013 BNE 35 ;DRIVES TO BE TESTED
3199 006250 110637 001126 1005: MOVB SP,PRGFLG ;SET FLAG TO IND ALL DRIVES
3200 006254 012701 001156 MOV #DRVTBL,R1 ;MARK ALL DRIVES TO BE TESTED
3201 006260 012700 000004 MOV #4,RO ;A '-1' INDICATES THAT A DRIVE
3202 006264 012721 177777 25: MOV #-1,(R1)+ ;IS TO BE TESTED
3203 006270 005300 DEC RO
3204 006272 001374 BNE 25
3205 006274 000417 BR CHKDRV ;GO CHECK DRIVE AVAILABILITY
3206
3207 ;GET USER SELECTED DRIVES AND MARK EACH DRIVE SELECTED TO BE TESTED
3208 006276 122710 000015 35: CMPB #CR,(RO)
3209 006302 001414 BEQ CHKDRV
3210 006304 121027 000054 CMPB (RO),'#', ;CHECK IF 'COMMA'
3211 006310 001001 BNE 45
3212 006312 105720 TSTB (RO)+ ;STEP PTR PAST 'COMMA'
3213 006314 112001 45: MOVB (RO)+,R1
3214 006316 042701 177770 BIC #177770,R1
3215 006322 112761 177777 001156 MOVB #-1,DRVTBL(R1)
3216 006330 000240 NOP
3217 006332 000761 BR 35
3218
3219 ;ASCERTAIN THAT DRIVES (TMO2'S) SPECIFIED ARE AVAILABLE
3220 006334 005000 CHKDRV: CLR RO ;A 0/-1 INDICATES THAT THE
3221 006336 105760 001156 15: TSTB DRVTBL(RO) ;DRIVE IS NOT/IS TO BE TESTED
3222 006342 001005 BNE 35
3223 006344 005200 25: INC RO
3224 006346 122700 000010 CMPB #8,RO
3225 006352 001371 BNE 15
3226 006354 000421 BR 45
3227 006356 110037 001006 35: MOVB RO,@#DRVNUM
3228 006362 004737 005142 JSR PC,@#DRVAVA ;CHECK IF AVAILABLE
3229 006366 102366 BVC 25 ;'V' BIT SET INDICATES NOT AVAIL
3230 006370 000004 014106 TYPE,E,NDRV
3231 006374 116037 001134 014140 MOVB DIGTAB(RO),@#E.NAVA ;SET DRIVE # IN MESSAGE
3232 006402 000004 014140 TYPE,E,NAVA
3233 006406 110637 001125 MOVB SP,@#ERFLG ;SET 'ERROR' FLAG
3234 006412 105060 001156 CLR B DRVTBL(RO) ;MARK DRIVE UNAVAILABLE
3235 006416 000752 BR 25 ;CHECK NEXT DRIVE
3236 006420 105737 001125 45: TSTB @#ERFLG ;GO GET SLAVES IF NO ERROR
3237 006424 001403 BEQ SLAVES
3238 006426 105737 001126 TSTB @#PRGFLG ;ASK USER TO RETYPE DRIVES IF
3239 006432 001660 BEQ DRIVES ;'ALL' NOT SPECIFIED
3240
```

```

3241 ;ROUTINE TO GET SLAVES (TU16'S) USER DESIRES TO TEST
3242 006434 105037 001125 SLAVES: CLR      @#ERFLG      ;CLEAR 'ERROR' FLAG
3243 006440 012701 001166      MOV      #SLVTBL,R1      ;MARK ALL SLAVES (64.) AS NOT
3244 006444 012700 000040      MOV      #32.,RO        ;TO BE TESTED. A 0 INDICATES THAT
3245 006450 005021          1$: CLR      (R1)+          ;A DRIVE'S SLAVE IS NOT TO BE
3246 006452 005300          DEC      RO              ;TESTED
3247 006454 001375          BNE     1$
3248 006456 005000          CLR      RO              ;RO = DRIVE # FOR SLAVES
3249 006460 012701 001166      MOV      #SLVTBL,R1      ;R1 POINTS TO DRIVE'S SLAVE
3250 006464 105760 001156      2$: TSTB   DRVTBL(RO)     ;IF DRIVE IS TO BE TESTED
3251 006470 001007          BNE     4$              ;GO TO 4$ OTHERWISE
3252 006472 062701 000010      3$: ADD     #8.,R1        ;STEP SLAVE PTR TO NEXT DRIVE'S
3253 006476 005200          INC     RO              ;SLAVES AND INCREMENT DRIVE #
3254 006500 122700 000010      CMPB   #8.,RO          ;CHECK ALL DRIVES
3255 006504 001367          BNE     2$              ;AND WHEN ALL DRIVES CHECKED
3256 006506 000454          BR      CHKSLV         ;GO CHECK SLAVE AVAILABILITY
3257
3258 006510 105737 001126      4$: TSTB   @#PRGFLG      ;BRANCH IF USER SELECTED ALL
3259 006514 001020          BNE     5$              ;DRIVES
3260 006516 110037 001006      MOVB   RO,DRVNUM        ;GET DRIVE #
3261 006522 116037 001134 013650 MOVB   DIGTAB(RO),@#1.DRV ;PREPARE USER ACTION MESSAGE
3262 006530 000004 013631      TYPE,1,SLVS
3263 006534 004737 003476      JSR    PC.,INPUT        ;GET USER INPUT
3264 006540 012703 001266      MOV    #INBUF,R3        ;SET PTR TO USER INPUT
3265 006544 122710 000101      CMPB   #'A,(RO)        ;BRANCH IF USER DOES NOT WANT
3266 006550 001015          BNE     7$              ;'ALL' SLAVES
3267 006552 110637 001126      MOVB   SP,@#PRGFLG      ;SET 'ALL' INDICATOR
3268 006556 012701 001166      5$: MOV    #SLVTBL,R1      ;MARK ALL SLAVES FOR ALL
3269 006562 012700 000040      MOV    #32.,RO          ;DRIVES AS TO BE TESTED
3270 006566 012721 177777      6$: MOV    #-1,(R1)+
3271 006572 005300          DEC    RO
3272 006574 001374          BNE    6$
3273 006576 105737 001126      TSTB   @#PRGFLG        ;BRANCH IF ALL WAS SELECTED
3274 006602 001016          BNE    CHKSLV
3275
3276 006604 122713 000015      7$: CMPB   #CR,(R3)        ;GET USER SELECTED SLAVES FOR
3277 006610 001730          BEQ    3$              ;DRIVE
3278 006612 121327 000054      CMPB   (R3),#',        ;STEP PTR PAST 'COMMA'
3279 006616 001001          BNE    8$
3280 006620 105723          TSTB   (R3)+
3281 006622 112304          8$: MOVB   (R3)+,R4        ;AND MARK SELECED SLAVE
3282 006624 042704 177770      BIC    #177770,R4      ;AS TO BE TESTED
3283 006630 060104          ADD    R1,R4
3284 006632 112714 177777      MOVB   #-1,(R4)
3285 006636 000762          BR     7$
3286
3287 ;ASCERTAIN THAT SLAVES (TU16'S) SELECTED ARE AVAILABLE
3288 006640 005000      CHKSLV: CLR    RO        ;RO WILL CONTAIN THE DRIVE #
3289 006642 005001      CLR    R1          ;AND R1 THE SLAVE #
3290 006644 012702 001166      MOV    #SLVTBL,R2    ;SET PTR TO SLAVE TABLE
3291 006650 105760 001156      1$: TSTB   DRVTBL(RO)  ;BRANCH IF DRIVE SELECTED
3292 006654 001007          BNE    3$            ;& AVAILABLE FOR TEST
3293 006656 005200          2$: INC    RO          ;INCREMENT DRIVE #
3294 006660 062702 000010      ADD    #8.,R2        ;STEP SLAVE PTR TO NEXT DRIVE'S
3295 006664 022700 000010      CMP    #8.,RO        ;SLAVES. BRANCH TO 1$ IF NOT ALL
3296 006670 001367          BNE    1$            ;DRIVES CHECKED OTHERWISE EXIT

```

```

3297 006672 000434          BR      7%
3298
3299 006674 005001          3%:   CLR      R1           ; SET SLAVE # 0
3300 006676 105712          4%:   TSTB   (R2)         ; BRANCH IF DRIVE'S SLAVE IS SEL-
3301 006700 001006          BNE    6%           ; ECTED FOR TEST
3302 006702 005201          5%:   INC     R1           ; INCREMENT SLAVE #
3303 006704 005202          INC     R2           ; STEP PTR TO NEXT SLAVE
3304 006706 022701 000010    CMP    #8, R1       ; GO TO 4% IF ALL SLAVES NOT
3305 006712 001371          BNE    4%           ; CHECKED
3306 006714 000760          BR     2%           ; OTHERWISE GO TO 2% ABOVE
3307
3308 006716 110037 001006    6%:   MOVB   RD, @#DRVNUM ; PASS DRIVE & SLAVE #
3309 006722 110137 001007    MOVB   R1, @#SLVNUM
3310 006726 004737 005170    JSR    PC, @#SLVAVA ; AND CHECK IF AVAILABLE
3311 006732 102363          BVC    5%           ; 'V' BIT SET ON RETURN IND-
3312 006734 116037 001134 014130    MOVB   DIGTAB(RD), @#E.DRV ; ICATES ERROR. PREPARE ERROR
3313 006742 116137 001134 014140    MOVB   DIGTAB(R1), @#E.NAVA ; MESSAGE
3314 006750 000004 014122    TYPE, E.NSLV
3315 006754 110637 001125    MOVB   SP, @#ERFLG  ; SET ERROR INDICATOR
3316 006760 105012          CLRB   (R2)         ; CLEAR SLAVE TABLE ENTRY
3317 006762 000747          BR     5%           ; GET NEXT SLAVE
3318
3319 006764 105737 001125    7%:   TSTB   @#ERFLG     ; BRANCH IF NO ERROR
3320 006770 001403          BEQ    100%
3321 006772 105737 001126    TSTB   @#PRGFLG     ; BRANCH IF NOT 'ALL'
3322 006776 001616          BEQ    SLAVES       ; ASK USER TO RETYPE SLAVES
3323 007000 012737 003652 000004 100%:  MOV    #ERRTRP, @#ERRVEC
3324
3325          ; SCAN DIVE AND SLAVE TABLE FOR DRIVE/SLAVE COMBINATION TO TEST
3326 007006 105037 001006          CLRB   @#DRVNUM     ; SET DRIVE AND SLAVE # 0
3327 007012 105037 001007          CLRB   @#SLVNUM
3328 007016 012737 001166 001010    MOV    #SLVTBL, @#SLVPTR ; SET PTR TO SLAVE TABLE
3329 007024 105037 001127          CLRB   @#UNTFND     ; CLEAR 'UNIT FOUND' IND.
3330
3331 007030 113700 001006    BEGIN: MOVB   @#DRVNUM, R0 ; GET DRIVE #
3332 007034 113701 001007    MOVB   @#SLVNUM, R1 ; AND SLAVE #

```


3335	007040	013702	001010		MOV	@#SLVPTR,R2		;GET SLAVE PTR
3336	007044	105760	001156	15:	TSTB	DRVTBL(RO)		;BRANCH IF DRIVE AVAIL TO TEST
3337	007050	001011			BNE	3\$		
3338	007052	005001			CLR	R1		;CLEAR SLAVE #
3339	007054	062702	000010		ADD	#8.,R2		;AND STEP PTR TO NEXT DRIVE'S
3340	007060	005200		25:	INC	RO		;SLAVES AND INCREMENT DRIVE #
3341	007062	022700	000010		CMP	#8.,RO		;EXIT TEST IF ALL DRIVES
3342	007066	001366			BNE	1\$;CHECKED OTHERWISE CONTINUE
3343	007070	000137	013016		JMP	@#END		;SCAN FOR NEXT 'UNIT'
3344								
3345	007074	105712		35:	TSTB	(R2)		;BRANCH IF SLAVE ON DRIVE IS
3346	007076	001007			BNE	4\$;AVAILABLE THERWISE STEP
3347	007100	005202			INC	R2		;PTR TO NEXT SLAVE
3348	007102	005201			INC	R1		;INCREMENT SLAVE #
3349	007104	122701	000010		CMPB	#8.,R1		;UNTIL ALL SLAVES CHECKED
3350	007110	001371			BNE	3\$;WHEN ALL SLAVES CHECKED
3351	007112	005001			CLR	R1		;SET SLAVE # 0
3352	007114	000761			BR	2\$;AND CONTINUE SCAN
3353								
3354	007116	110637	001127	45:	MOVB	SP,@#UNTFND		;INDICATE THAT A 'UNIT' IS FOUND
3355	007122	110037	001006		MOVB	RO,@#DRVNUM		;SET DRIVE 3

```

3360 007126 110137 001007      MOVB   R1, @#SLVNUM      ;SET SLAVE #
3361 007132 010237 001010      MOV    R2, @#SLVPTR     ;SAVE SLAVE PTR
3362
3363 007136 105737 001132      55:   TSTB   @#ASFLG
3364 007142 001054              BNE    75
3365 007144 112737 000001 001132  MOVB   #1, ASFLG
3366
3367 007152 105037 001126      CLRB   @#PRGFLG        ;CLEAR PROGRAM INDICATOR
3368 007156 023737 000042 000046  CMP    @#42, @#46      ;ACT11 AUTO MODE?
3369 007164 001417              BEQ    65               ;YES. DON'T RUN SKEW TESTS.
3370 007166 000004 013710      TYPE, I. SKEW         ;ASK USER IF HE WANTS TO RUN SKEW TESTS
3371 007172 004737 003476      JSR    PC, .INPUT     ;GET USER INPUT
3372 007176 012703 001266      MOV    #INBUF, R3     ;GET REPLY
3373 007202 122713 000060      CMPB   #'0, (R3)      ;BRANCH IF 'NO' (0)
3374 007206 001406              BEQ    65
3375 007210 122713 000061      CMPB   #'1, (R3)      ;CHECK IF 'YES' (1)
3376 007214 001350              BNE    55              ;NEITHER SO ASK AGAIN
3377 007216 111337 001126      MOVB   (R3), @#PRGFLG ;SET INDICATOR
3378 007222 000424              BR     75
3379
3380 007224 105037 001131      65:   CLRB   @#NRZFLG        ;CLEAR NRZ INDICATOR
3381 007230 023737 000042 000046  CMP    @#42, @#46      ;ACT11 AUTO MODE?
3382 007236 001416              BEQ    75               ;YES. NOT NRZ ONLY.
3383 007240 000004 013751      TYPE, I. NRZ         ;ASK USER IF DRIVE 'NRZ' ONLY
3384 007244 004737 003476      JSR    PC, .INPUT     ;GET USER INPUT
3385 007250 012703 001266      MOV    #INBUF, R3     ;GET REPLY
3386 007254 122713 000060      CMPB   #'0, (R3)      ;BRANCH IF 'NO' (0)
3387 007260 001405              BEQ    75
3388 007262 122713 000061      CMPB   #'1, (R3)      ;CHECK IF 'YES' (1)
3389 007266 001356              BNE    65              ;ASK AGAIN IF NEITHER
3390 007270 111337 001131      MOVB   (R3), @#NRZFLG ;SET INDICATOR
3391 007274
3392
3393 007274 052737 000100 177560  TYPHDR: BIS    #100, @#TKS      ;SET KEYBOARD IE BIT
3394 007302 000004 014474      TYPE, L. HDR1
3395 007306 116037 001134 014654  MOVB   DIGTAB(R0), @#L. DRV ;SET DRIVE #
3396 007314 116137 001134 014666  MOVB   DIGTAB(R1), @#L. SLV ;AND SLAVE #
3397 007322 112737 000071 014671  MOVB   #'9, @#L. CHAN     ;GET SLAVES CHANNEL TYPE
3398 007330 032765 010000 000026  BIT    #CH7, DT(R5)
3399 007336 001403              BEQ    15
3400 007340 112737 000067 014671  MOVB   #'7, @#L. CHAN     ;SET 7 CHANNEL
3401 007346 000004 014607      15:   TYPE, L. HDR2
3402 007352 004737 005620      JSR    PC, SNPT        ;GO PRINT SERIAL NUMBER
3403 007356 000004 014710      TYPE, L. HDR3
3404 007362 012737 007422 001004  MOV    #TST001, @#SCPADR ;SET 'SCOPE' ADDRESS FOR FIRST TEST
3405 007370 010500              MOV    R5, R0
3406 007372 062700 000006      ADD    #FC, R0         ;R0 CONTAINS ADDRESS OF FC REG
3407 007376 010501              MOV    R5, R1
3408 007400 062701 000012      ADD    #DS, R1        ;R1 CONTAINS ADDRESS OF DS REG
3409 007404 012703 004470      MOV    #TIMER, R3     ;SET JUMP ADDRESS TO TIMER
3410 007410 105737 001126      TSTB   @#PRGFLG       ;BRANCH IF NOT SKEW TESTS
3411 007414 001402              BEQ    TST001
3412 007416 000137 013110      JMP    @#SKEWTST

```

```

3414          .SBTTL START OF TESTS
3415          ;TEST 001 - WRITE FROM BOT
3416          ;THIS TEST WILL MEASURE ACCELERATION DELAY REQUIRED TO
3417          ;MOVE THE TAPE APPROXIMATELY SEVEN (7) INCHES FORWARD
3418          ;FROM DEAD STOP BEFORE STARTING TO TRANSFER DATA.
3419
3420          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
3421 007422 112737 000001 001124 TST001: MOVB #1,@#TSTNUM ;SET TEST #
3422 007430 012702 007454          MOV #15,R2 ;SET RETURN PC FROM TIMER
3423 007434 004737 005362          JSR PC,REWIND ;REWIND SLAVE
(1) 007440 102420          BVS 995 ;BRANCH IF ERROR ON REWIND
3424 007442 004737 005444          JSR PC,WRITE ;GO SETUP WRITE COMMAND
3425 007446 004737 004400          JSR PC,TIMON ;TURN TIMER ON
3426 007452 005215          INC (R5) ;SET 'GO' BIT
3427
3428 007454 005765 000032          15: TST TC(R5) ;BRANCH WHEN 'ACCL'=0
3429 007460 100002          BPL 25
3430 007462 000163 004470          JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3431
3432 007466 004737 005256          25: JSR PC,WAITRDY ;WAIT FOR COMMAND TO FINISH
3433 007472 102403          BVS 995 ;BRANCH IF ERROR
3434 007474 004737 004526          JSR PC,TIMOK ;GO CHECK TIME
3435 007500 000401          BR 1005
3436 007502 104400          995: HLT
3437 007504 104000          1005: SCOPE
3438
3439          ;TEST 002 - WRITE START
3440          ;THIS TST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
3441 007506 112737 000002 001124 TST002: MOVB #2,@#TSTNUM ;SET TEST # 2
3442 007514 004737 005444          JSR PC,WRITE ;INITIATE WRITE COMMAND
3443 007520 012702 007532          MOV #15,R2 ;SET RETURN PC FROM TIMER
3444 007524 004737 004400          JSR PC,TIMON
3445 007530 005215          INC (R5) ;SET 'GO' BIT
3446
3447 007532 005765 000032          15: TST TC(R5) ;BRANCH WHEN 'ACCL'=0
3448 007536 100002          BPL 25
3449 007540 000163 004470          JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3450
3451 007544 004737 005256          25: JSR PC,WAITRDY ;WAIT FOR READY
3452 007550 102403          BVS 995 ;BRANCH IF ERROR
3453 007552 004737 004526          JSR PC,TIMOK ;GO CHECK TIME RECORDED
3454 007556 000401          BR 1005 ;EXIT VIA SCOPE
3455
3456 007560 104400          995: HLT ;REPORT ERROR
3457 007562 104000          1005: SCOPE
3458
3459          ;TEST 003- WRITE SHUTDOWN
3460          ;THIS TEST MEASURES TIME FROM 'FC REG'=0 TO 'SWDN'=1.
3461 007564 112737 000003 001124 TST003: MOVB #3,@#TSTNUM ;SET TEST#3
3462 007572 004737 005444          JSR PC,WRITE ;INITIATE WRITE COMMAND
3463 007576 005215          INC (R5) ;SET 'GO' BIT
3464
3465 007600 005710          15: TST (R0) ;BRANCH WHEN WRITING FINISHED
3466 007602 001404          BEQ 25
3467 007604 032711 040000          BIT #ERR,(R1) ;MONITOR ERROR BIT
3468 007610 001017          BNE 995

```

```

3469 007612 000772          BR      15
3470
3471 007614          25:
(1) 007614 004737 004400      JSR      PC,TIMON      ;TURN TIMER ON
3472 007620 010702          MOV      PC,R2        ;LOAD RETURN PC FROM TIMER
3473 007622 032711 000020      35:      BIT      #SDWN,(R1) ;BRANCH WHEN DS <SDWN> SETS
3474 007626 001002          BNE     45
3475 007630 000163 004470          JMP      TIMER(R3)    ;GO TO TIMER & RETURN VIA R2
3476
3477 007634 004737 005256      45:      JSR      PC,WAITRDY   ;WAIT FOR READY
3478 007640 102403          BVS     99$
3479 007642 004737 004526      JSR      PC,TIMOK     ;GO CHECK TIME RECORDED
3480 007646 000401          BR      100$
3481 007650 104400          99$:     HLT
3482 007652 104000          100$:    SCOPE      ;REPORT ERROR
3483
3484
3485          ;TEST 004 - WRITE SETTLEDOWN
3486 007654 112737 000004 001124  TST004: MOVB     #4,@#TSTNUM
3487 007662 004737 005444          JSR      PC,WRITE
3488 007666 005215          INC     (R5)         ;SET 'GO' BIT
3489
3490 007670 005710          15:      TST     (R0)         ;BRANCH WHEN WRITING FINISHED
3491 007672 001404          BEQ     25
3492 007674 032711 040000      BIT     #ERP,(R1)    ;CHECK ERROR BIT
3493 007700 001026          BNE     99$
3494 007702 000772          BR      15
3495
3496 007704 032711 000020      25:      BIT     #SDWN,(R1)  ;WAIT FOR ASSERTION OF 'SDWN'
3497 007710 001004          BNE     35
3498 007712 032711 040000      BIT     #ERR,(R1)   ;MONITOR ERROR BIT
3499 007716 001017          BNE     99$
3500 007720 000771          BR      25
3501
3502 007722          35:
(1) 007722 004737 004400      JSR      PC,TIMON   ;TURN TIMER ON
3503 007726 010702          MOV      PC,R2     ;SET RETURN PC FROM TIMER
3504 007730 032711 000020      BIT     #SDWN,(R1) ;BRANCH WHEN SWDN CLEARS
3505 007734 001402          BEQ     55
3506 007736 000163 004470          JMP      TIMER(R3) ;GO TO TIMER & RETURN VIA R2
3507
3508 007742 004737 005256      55:      JSR      PC,WAITRDY ;WAIT FOR READY
3509 007746 102403          BVS     99$
3510 007750 004737 004526      JSR      PC,TIMOK
3511 007754 000401          BR      100$
3512
3513 007756 104400          99$:     HLT
3514 007760 104000          100$:    SCOPE
3515
3516          ;TEST 005 - READ FROM BOT
3517          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
3518 007762 112737 000005 001124  TST005: MOVB     #5,@#TSTNUM ;SET TEST #5
3519 007770 004737 005362          JSR      PC,REWIND  ;REWIND SLAVE
(1) 007774 102422          BVS     99$        ;BRANCH IF ERROR ON REWIND
3520 007776 004737 005462          JSR      PC,READ
3521 010002 012702 010014          MOV     #15,R2     ;SET RETURN PC FROM TIMER

```

```

3522 010006 004737 004400      JSR   PC,TIMON      ;TURN TIMER ON
3523 010012 005215              INC   (R5)          ;SET 'GO' BIT
3524
3525 010014 005765 000032      15:   TST   TC(R5)      ;BRANCH WHEN 'ACCL' RESETS
3526 010020 100002              BPL   25
3527 010022 000163 004470      JMP   TIMER(R3)    ;GO TO TIMER & RETURN VIA R2
3528
3529 010026 004737 005256      25:   JSR   PC,WAITRDY   ;WAIT FOR READY
3530 010032 102403              BVS   99$          ;BRANCH IF ERROR
3531 010034 004737 004526      JSR   PC,TIMOK     ;CHECK RECORDED TIME
3532 010040 000401              BR    100$
3533
3534 010042 104400              99$:   HLT
3535 010044 104000              100$:  SCOPE
3536
3537      ;TEST 006 - READ START
3538      ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
3539 010046 112737 000006 001124  TST006: MOVB   #6,#TSTNUM ;SET TEST #6
3540 010054 004737 005536      JSR   PC,WRT.BK    ;WRITE A RECORD & BACK SPACE
3541 010060 102422              BVS   99$
3542 010062 004737 005462      JSR   PC,READ
3543 010066 012702 010100      MOV   #15,R2      ;SET RETURN PC FROM TIMER
3544 010072 004737 004400      JSR   PC,TIMON    ;TURN TIMER ON
3545 010076 005215              INC   (R5)        ;SET 'GO' BIT
3546
3547 010100 005765 000032      15:   TST   TC(R5)      ;BRANCH WHEN 'ACCL' RESETS
3548 010104 100002              BPL   25
3549 010106 000163 004470      JMP   TIMER(R3)    ;GO TO TIMER & RETURN VIA R2
3550
3551 010112 004737 005256      25:   JSR   PC,WAITRDY   ;WAIT FOR READY
3552 010116 102403              BVS   99$          ;BRANCH IF ERROR
3553 010120 004737 004526      JSR   PC,TIMOK     ;CHECK RECORDED TIME
3554 010124 000401              BR    100$
3555
3556 010126 104400              99$:   HLT
3557 010130 104000              100$:  SCOPE
3558
3559      ;TEST 007 - READ SHUTDOWN
3560      ;THIS TEST MEASURES TIME FROM 'FC REG'=FRAME COUNT TO 'SWDN'=1.
3561 010132 112737 000007 001124  TST007: MOVB   #7,#TSTNUM ;SET TEST #7
3562 010140 004737 005536      JSR   PC,WRT.BK    ;WRITE A RECORD & BACK SPACE
3563 010144 102430              BVS   99$          ;BRANCH IF ERROR
3564 010146 004737 005462      JSR   PC,READ
3565 010152 005215              INC   (R5)        ;SET 'GO' BIT
3566
3567 010154 022710 000400      15:   CMP   #-FRMCNT,(R0) ;WAIT FOR FRAME COUNT TO
3568 010160 001404              BEQ   25           ;= # OF FRAMES WRITTEN
3569 010162 032711 040000      BIT   #ERR,(R1)   ;MONITOR ERROR BIT
3570 010166 001017              BNE   99$
3571 010170 000771              BR    15
3572
3573 010172              25:
3574 (1) 010172 004737 004400      JSR   PC,TIMON    ;TURN TIMER ON
3574 010176 010702              MOV   PC,R2      ;SET RETURN PC FROM TIMER
3575 010200 032711 000020      BIT   #SDWN,(R1) ;BRANCH WHEN SDWN SETS
3576 010204 001002              BNE   35

```

```

3577 010206 000163 004470          JMP     TIMER(R3)          ;GO TO TIMER & RETURN VIA R2
3578
3579 010212 004737 005256          3$:   JSR     PC, WAITRDY
3580 010216 102403                    BVS     99$
3581 010220 004737 004526          JSR     PC, TIMOK
3582 010224 000401                    BR      100$
3583
3584 010226 104400                    99$:   HLT
3585 010230 104000                    100$:  SCOPE              ;REPORT ERROR
3586
3587
3588
3589 010232 112737 000010 001124  TST010: MOVB   #10, @#TSTNUM ;TEST 010 - READ SETTLEDOWN
3590 010240 012702 010316          MOV     #4$, R2          ;THIS TEST MEASURES TIME FROM 'SWDN'=1 TO 'SWDN'=0.
3591 010244 004737 005536          JSR     PC, WRT. BK      ;SET TEST #10
3592 010250 102436                    BVS     99$              ;SET RETURN PC FROM TIMER
3593 010252 004737 005462          JSR     PC, READ        ;WRITE A RECORD & BACK SPACE
3594 010256 005215                    INC     (R5)             ;SET 'GO' BIT
3595
3596 010260 105711                    1$:   TSTB   (R1)         ;WAIT FOR READY
3597 010262 100404                    BMI     2$              ;BRANCH WHEN SET
3598 010264 032711 040000          BIT     #ERR, (R1)     ;CHECK ERROR BIT
3599 010270 001026                    BNE     99$
3600 010272 000772                    BR      1$
3601
3602 010274 032711 000020          2$:   BIT     #SDWN, (R1) ;WAIT FOR ASSERTION OF 'SDWN'
3603 010300 001004                    BNE     3$
3604 010302 032711 040000          BIT     #ERR, (R1)     ;MONITOR ERROR BIT
3605 010306 001017                    BNE     99$
3606 010310 000771                    BR      2$
3607
3608 010312
3609 (1) 010312 004737 004400          3$:   JSR     PC, TIMON   ;TURN TIMER ON
3610 010316 032765 000020 000012  4$:   BIT     #SDWN, DS(R5) ;WAIT FOR NEGATION OF SDWN
3611 010324 001402                    BEQ     5$
3612 010326 000163 004470          JMP     TIMER(R3)      ;GO TO TIMER & RETURN VIA R2
3613 010332 004737 005256          5$:   JSR     PC, WAITRDY
3614 010336 102403                    BVS     99$
3615 010340 004737 004526          JSR     PC, TIMOK
3616 010344 000401                    BR      100$
3617
3618 010346 104400                    99$:   HLT
3619 010350 104000                    100$:  SCOPE
3620
3621
3622
3623
3624 010352 112737 000011 001124  TST011: MOVB   #11, @#TSTNUM ;TEST 011-READ REVERSE START
3625 010360 012702 010416          MOV     #1$, R2          ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'ACCL'=0.
3626 010364 004737 005444          JSR     PC, WRITE      ;SET RETURN PC FROM TIMER
3627 010370 005215                    INC     (R5)             ;WRITE A RECORD
3628 010372 004737 005256          JSR     PC, WAITRDY    ;SET 'GO' BIT
3629 010376 102422                    BVS     99$
3630 010400 004737 004764          JSR     PC, DELAY      ;WAIT FOR TAPE MOTION TO STOP
3631 010404 004737 005500          JSR     PC, REVRD

```

```

3632 010410 004737 004400      JSR    PC,TIMON      ;TURN TIMER ON
3633 010414 005215              INC    (R5)          ;SET 'GO' BIT
3634
3635 010416 005765 000032      1$:   TST    TC(R5)      ;BRANCH WHEN 'ACCL' = 0
3636 010422 100002              BPL    2$
3637 010424 000163 004470      JMP    TIMER(R3)     ;GO TO TIMER & RETURN VIA R2
3638
3639 010430 004737 005256      2$:   JSR    PC,WAITRDY
3640 010434 102403              BVS    99$          ;BRANCH IF ERROR
3641 010436 004737 004526      JSR    PC,TIMOK
3642 010442 000401              BR     100$
3643
3644 010444 104400      99$:   HLT
3645 010446 104000      100$:  SCOPE
3646
3647      ;TEST 012-READ REVERSE SHUTDOWN
3648      ;THIS TEST MEASURES TIME FROM 'FC REG' = FRAME COUNT TO 'SDWN'=1.
3649 010450 112737 000012 001124  TST012: MOVB   #12,@#TSTNUM
3650 010456 012702 010526      MOV    #3$,R2      ;SET RETURN PC FROM TIMER
3651 010462 004737 005444      JSR    PC,WRITE     ;WRITE A RECORD
3652 010466 005215              INC    (R5)          ;SET 'GO' BIT
3653 010470 004737 005256      JSR    PC,WAITRDY
3654 010474 102427              BVS    99$
3655 010476 004737 005500      JSR    PC,REVRD
3656 010502 005215              INC    (R5)          ;SET 'GO' BIT
3657
3658 010504 022710 000400      1$:   CMP    #-FRMCNT,(R0) ;BRANCH WHEN FRAME COUNT
3659 010510 001404              BEQ    2$           ;= # OF RECORD WRITTEN
3660 010512 032711 040000      BIT    #ERR,(R1)   ;MONITOR ERROR BIT IN 'DS' REG
3661 010516 001016              BNE    99$
3662 010520 000771              BR     1$
3663
3664 010522      2$:
3664 (1) 010522 004737 004400      JSR    PC,TIMON     ;TURN TIMER ON
3665 010526 032711 000020      3$:   BIT    #SDWN,(R1) ;BRANCH WHEN SDWN SETS
3666 010532 001002              BNE    4$
3667 010534 000163 004470      JMP    TIMER(R3)   ;GO TO TIMER & RETURN VIA R2
3668
3669 010540 004737 005256      4$:   JSR    PC,WAITRDY ;WAIT FOR READY
3670 010544 102403              BVS    99$
3671 010546 004737 004526      JSR    PC,TIMOK
3672 010552 000401              BR     100$
3673
3674 010554 104400      99$:   HLT
3675 010556 104000      100$:  SCOPE
3676
3677      ;TEST 013-READ REVERSE SETTLEDOWN
3678      ;THIS TEST MEASURES TIME FROM 'SDWN'=1 TO 'SDWN'=0.
3679 010560 112737 000013 001124  TST013: MOVB   #13,@#TSTNUM
3680 010566 012702 010652      MOV    #4$,R2      ;SET RETURN PC FROM TIMER
3681 010572 004737 005444      JSR    PC,WRITE     ;WRITE A RECORD
3682 010576 005215              INC    (R5)          ;SET 'GO' BIT
3683 010600 004737 005256      JSR    PC,WAITRDY
3684 010604 102435              BVS    99$
3685 010606 004737 005500      JSR    PC,REVRD
3686 010612 005215              INC    (R5)          ;SET 'GO' BIT

```

3687						
3688	010614	105711		15:	TSTB (R1)	; BRANCH WHEN
3689	010616	100404			BMI 25	; READY SETS
3690	010620	032711	040000		BIT #ERR, (R1)	
3691	010624	001025			BNE 995	
3692	010626	000772			BR 15	
3693						
3694	010630	032711	000020	25:	BIT #SDWN, (R1)	
3695	010634	001004			BNE 35	
3696	010636	032711	040000		BIT #ERR, (R1)	
3697	010642	001016			BNE 995	
3698	010644	000771			BR 25	
3699						
3700	010646			35:		
(1)	010646	004737	004400		JSR PC, TIMON	; TURN TIMER ON
3701	010652	032711	000020	45:	BIT #SDWN, (R1)	; BRANCH WHEN SWDN = 0
3702	010656	001402			BEQ 55	
3703	010660	000163	004470		JMP TIMER(R3)	; GO TO TIMER & RETURN VIA R2
3704						
3705	010664	004737	005256	55:	JSR PC, WAITRDY	; WAIT FOR READY
3706	010670	102403			BVS 995	
3707	010672	004737	004526		JSR PC, TIMOK	
3708	010676	000401			BR 1005	
3709						
3710	010700	104400		995:	HLT	
3711	010702	104000		1005:	SCOPE	
3712						
3713					; REWIND DRIVE	
3714	010704			A:		
(1)	010704	004737	005362		JSR PC, REWIND	; REWIND SLAVE
(1)	010710	102401			BVS 995	; BRANCH IF ERROR ON REWIND
3715	010712	102002			BVC 1005	
3716	010714	104400		995:	HLT	
3717	010716	000772			BR A	
3718	010720			1005:		
3719						
3720					; TEST 014-TURN AROUND DELAY (FORWARD-REVERSE)	
3721					; THIS TEST MEASURES TIME FROM 'GO'=1 (READ REVERSE) TO 'ACCL'=0	
3722	010720	112737	000014	001124	TST014: MOVB #14, @#TSTNUM	
3723	010726	012702	010760		MOV #25, R2	; SET RETURN PC FROM TIMER
3724	010732	004737	005444		JSR PC, WRITE	; WRITE A RECORD
3725	010736	005215			INC (R5)	; SET 'GO' BIT
3726	010740	004737	005256		JSR PC, WAITRDY	
3727	010744	102420			BVS 995	
3728						
3729	010746	004737	005500	15:	JSR PC, REVRD	; READ THE RECORD (REVERSE)
3730	010752	004737	004400		JSR PC, TIMON	; TURN TIMER ON
3731	010756	005215			INC (R5)	; SET 'GO' BIT
3732						
3733	010760	005765	000032	25:	TST TC(R5)	; WAIT FOR 'ACCL' = 0
3734	010764	100002			BPL 35	
3735	010766	000163	004470		JMP TIMER(R3)	; GO TO TIMER & RETURN VIA R2
3736						
3737	010772	004737	005256	35:	JSR PC, WAITRDY	
3738	010776	102403			BVS 995	
3739	011000	004737	004526		JSR PC, TIMOK	


```

3740 011004 000401          BR      100$
3741
3742 011006 104400          99$:  HLT
3743 011010 104000          100$: SCOPE
3744
3745          ; TEST 015- TURN AROUND DELAY (REVERSE-FORWARD)
3746          ; THIS TEST MEASURES TIME FROM 'GO'=1 (READ) TO 'ACCL'=0.
3747 011012 112737 000015 001124 TST015: MOVB  #15, @#TSTNUM
3748 011020 012702 011066          MOV   #2$, R2          ; SET RETURN PC FROM TIMER
3749 011024 004737 005444          JSR  PC, WRITE        ; WRITE A RECORD
3750 011030 005215          INC   (R5)            ; SET 'GO' BIT
3751 011032 004737 005256          JSR  PC, WAITRDY      ; WAIT FOR READY
3752 011036 102426          BVS  99$
3753 011040 004737 005500          JSR  PC, REVRD        ; READ A RECORD IN THE
3754 011044 005215          INC   (R5)            ; SET 'GO' BIT
3755
3756 011046 004737 005256          JSR  PC, WAITRDY
3757 011052 102420          BVS  99$
3758
3759 011054 004737 005462          1$:  JSR  PC, READ      ; READ RECORD FORWARD
3760 011060 004737 004400          JSR  PC, TIMON        ; TURN TIMER ON
3761 011064 005215          INC   (R5)            ; SET 'GO' BIT
3762
3763 011066 005765 000032          2$:  TST  TC(R5)        ; WAIT FOR 'ACCL' = 0
3764 011072 100002          BPL  3$
3765 011074 000163 004470          JMP  TIMER(R3)        ; GO TO TIMER & RETURN VIA R2
3766
3767 011100 004737 005256          3$:  JSR  PC, WAITRDY
3768 011104 102403          BVS  99$
3769 011106 004737 004526          JSR  PC, TIMOK
3770 011112 000401          BR   100$
3771
3772 011114 104400          99$:  HLT
3773 011116 104000          100$: SCOPE
3774
3775          ; TEST 016-GAP SIZE (STOP HALF)
3776 011120 112737 000016 001124 TST016: MOVB  #16, @#TSTNUM
3777 011126 012702 011164          MOV   #1$, R2          ; SET RETURN PC FROM TIMER
3778 011132 004737 005444          JSR  PC, WRITE        ; WRITE A RECORD
3779 011136 005215          INC   (R5)            ; SET 'GO' BIT
3780 011140 004737 005256          JSR  PC, WAITRDY
3781 011144 102421          BVS  99$
3782 011146 004737 004764          JSR  PC, DELAY        ; DELAY 350 MS
3783 011152 004737 005500          JSR  PC, REVRD        ; READ REVERSE RECORD
3784 011156 004737 004400          JSR  PC, TIMON        ; TURN TIMER ON
3785 011162 005215          INC   (R5)            ; SET 'GO' BIT
3786
3787 011164 005710          1$:  TST  (R0)            ; WAIT FOR FRAME COUNT > 0
3788 011166 001002          BNE  2$
3789 011170 000163 004470          JMP  TIMER(R3)        ; GO TO TIMER & RETURN VIA R2
3790
3791 011174 004737 005256          2$:  JSR  PC, WAITRDY      ; WAIT FOR READY BIT TO SET
3792 011200 102403          BVS  99$
3793 011202 004737 004526          JSR  PC, TIMOK        ; CHECK TIME
3794 011206 000401          BR   100$
3795

```

```

3796 011210 104400          99$:   HLT
3797 011212 104000          100$:  SCOPE
3798
3799
3800 011214 112737 000017 001124 ;TEST 017-GAP SIZE (START HALF)
3801 011222 012702 011274 TST017: MOVB  #17, @#TSTNUM
3802 011226 004737 005444      MOV  #15, R2          ;SET RETURN PC FROM TIMER
3803 011232 005215          JSR  PC, WRITE      ;WRITE A RECORD
3804 011234 004737 005256      INC  (R5)          ;SET 'GO' BIT
3805 011240 102427          JSR  PC, WAITRDY    ;WAIT FOR READY
3806 011242 004737 005500      BVS  99$
3807 011246 005215          JSR  PC, REVRD      ;READ REVERSE THE RECORD
3808 011250 004737 005256      INC  (R5)          ;SET 'GO' BIT
3809 011254 102421          JSR  PC, WAITRDY    ;WAIT FOR READY
3810 011256 004737 004764      BVS  99$          ;BRANCH ON ERROR
3811 011262 004737 005462      JSR  PC, DELAY      ;WAIT FOR TAPE MOTION TO STOP
3812 011266 004737 004400      JSR  PC, READ       ;READ RECORD
3813 011272 005215          JSR  PC, TIMON      ;TURN TIMER ON
3814
3815 011274 005710          INC  (R5)          ;SET 'GO' BIT
3816 011276 001002          TST  (R0)          ;WAIT FOR FRAME COUNT > 0
3817 011300 000163 004470      BNE  2$
3818
3819 011304 004737 005256          JMP  TIMER(R3)     ;GO TO TIMER & RETURN VIA R2
3820 011310 102403          JSR  PC, WAITRDY    ;WAIT FOR READY
3821 011312 004737 004526      BVS  99$
3822 011316 000401          JSR  PC, TIMOK      ;CHECK TIME
3823
3824 011320 104400          BR   100$
3825 011322 104000          99$:   HLT
3826
3827
3828
3829 011324 112737 000020 001124 ;TEST 020- GAP SIZE (INTERRECORD)
3830 011332 012702 011414 ;THIS TEST MEASURES TIME FROM 'GO'=1 TO 'FC REG' >0.
3831 011336 004737 005444 TST020: MOVB  #20, @#TSTNUM
3832 011342 005215          MOV  #15, R2          ;SET RETURN PC FROM TIMER
3833 011344 004737 005256      JSR  PC, WRITE      ;WRITE A RECORD
3834 011350 102433          INC  (R5)          ;SET 'GO' BIT
3835 011352 004737 005444      JSR  PC, WAITRDY    ;WAIT FOR READY
3836 011356 005215          BVS  99$
3837 011360 004737 005256      JSR  PC, WRITE      ;WRITE SECOND RECORD
3838 011364 102425          INC  (R5)          ;SET 'GO' BIT
3839 011366 004737 005500      JSR  PC, WAITRDY    ;WAIT FOR READY
3840 011372 005215          BVS  99$
3841 011374 004737 005256      JSR  PC, REVRD      ;READ REVERSE SECOND RECORD
3842 011400 102417          INC  (R5)          ;SET 'GO' BIT
3843 011402 004737 005500      JSR  PC, WAITRDY    ;WAIT FOR READY
3844 011406 004737 004400      BVS  99$
3845 011412 005215          JSR  PC, REVRD      ;READ REVERSE FIRST RECORD
3846
3847 011414 005710          JSR  PC, TIMON      ;TURN TIMER ON
3848 011416 001002          INC  (R5)          ;SET 'GO' BIT
3849 011420 000163 004470      TST  (R0)          ;WAIT FOR FRAME COUNT > 0
3850
3851 011424 004737 005256          BNE  2$
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000

```

3852 011430 102403
 3853 011432 004737 004526
 3854 011436 000401

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

3855
 3856 011440 104400
 3857 011442 104000

99\$: HLT
 100\$: SCOPE

3858
 3859 ; TEST Q21- GAP CONSISTANCY
 3860 ; THIS TEST MEASURES TIME FROM 'GO'=1 TO 'FC REG' > 0.
 3861 ; THE TEST REWINDS THE TAPE, WRITES 17 RECORDS WITH A DELAY FROM 1-16 MS
 3862 ; BETWEEN EACH WRITE COMMAND. AFTER THE 17. RECORDS ARE WRITTEN THE
 3863 ; PROGRAM READ REVERSES 16 RECORDS. AT THIS POINT THE TAPE IS STOPPED BE-
 3864 ; TWEEN THE FIRST AND SECOND RECORD. A READ COMMAND IS EXECUTED TO READ
 3865 ; THE 16 RECORDS WITH THE TIME BETEWEN GO=1 TO FC > 0 STORED IN 'GAPTBL'
 3866 ; FOR EACH RECORD READ. AFTER 16 RECORDS HAVE BEEN READ THE TIME IS VER-
 3867 ; IFIED FOR EACH READ. AFTER ALL RECORD TIMES ARE VERIFIED THEY ARE AVER-
 3868 ; AGED AND PLACED IN THE 'ATIMTBL' (BY SCOPE). THE ABOVE PROCESS IS RE-
 3869 ; PEATED FOR EACH ITERATION.
 3870

3871 011444 112737 000021 001124
 3872 011452 012702 011610
 3873 011456 004737 005362

TSTQ21: MOV #21, @#TSTNUM
 MOV #4\$, R2
 JSR PC, REWIND

; SET RETURN PC FROM TIMER
 ; REWIND SLAVE

(1) 011462 102530
 3874 011464 005037 001116
 3875 011470 012700 000021

BVS 99\$
 CLR DELTIM
 MOV #17, RO
 1\$: JSR PC, WRITE

; BRANCH IF ERROR ON REWIND
 ; CLEAR VARIABLE DELAY TIME
 ; SET # OF RECORDS TO WRITE
 ; WRITE 17. RECORDS

3876 011474 004737 005444
 3877 011500 005215
 3878 011502 004737 005256

INC (R5)
 JSR PC, WAITRDY
 BVS 99\$

; SET 'GO' BIT
 ; WAIT FOR READY
 ; DELAY BEFORE WRITING NEXT REC.

3879 011506 102516
 3880 011510 004737 005014
 3881 011514 062737 000022 001116

JSR PC, DELAYV
 ADD #18, DELTIM
 DEC RO
 BNE 1\$

; SET NEXT DELAY TIME
 ; DECREMENT RECORDS WRITTEN COUNT

3882 011522 005300
 3883 011524 001363
 3884

MOV #17, RO
 2\$: JSR PC, REVRD
 INC (R5)

; SET # OF RECS. TO REVERSE READ
 ; REVERSE READ 17. RECORDS
 ; SET 'GO' BIT

3885 011526 012700 000021
 3886 011532 004737 005500
 3887 011536 005215
 3888 011540 004737 005256

JSR PC, WAITRDY
 BVS 99\$
 DEC RO

; WAIT FOR READY
 ; DECREMENT RECORD COUNT

3889 011544 102477
 3890 011546 005300
 3891 011550 001370
 3892

BNE 2\$
 MOV #16, RO
 MOV #GAPTBL, R1
 JSR PC, READ
 INC (R5)

; SET # OF RECORDS TO READ
 ; SET PTR TO GAP TABLE FOR TEST
 ; READ A RECORD
 ; SET 'GO' BIT

3893 011552 012700 000020
 3894 011556 012701 001056
 3895 011562 004737 005462
 3896 011566 005215

3\$: JSR PC, WAITRDY
 BVS 99\$
 JSR PC, READ
 JSR PC, TIMON
 INC (R5)

; WAIT FOR READY
 ; READ NEXT RECORD
 ; TURN TIMER ON
 ; SET 'GO' BIT

3897
 3898 011570 004737 005256
 3899 011574 102463

4\$: TST FC(R5)
 BNE 5\$
 JMP TIMER(R3)

; WAIT FOR FRAME COUNT > 0
 ; GO TO TIMER & RETURN VIA R2

3900 011576 004737 005462
 3901 011602 004737 004400
 3902 011606 005215
 3903

3904 011610 005765 000006
 3905 011614 001002
 3906 011616 000163 004470

```

3907
3908 011622 004737 005256      55:   JSR   PC, WAITRDY      ;WAIT FOR READY
3909 011626 102446              BVS   99$
3910 011630 010421              MOV   R4, (R1)+        ;STORE TIME IN GAP TBL
3911 011632 005300              DEC   R0                ;DECREMENT # OF RECORDS READ
3912 011634 001355              BNE   3$
3913
3914 011636 105037 001122      CLR B @#GAP            ;SET GAP # 0
3915 011642 012700 000020      MOV   #16., R0
3916 011646 012701 001056      MOV   #GAPTBL, R1
3917
3918 011652 012104              65:   MOV   (R1)+, R4        ;GET GAP TICK COUNT
3919 011654 004737 004636      JSR   PC, GAPOK        ;CHECK TIME
3920 011660 105237 001122      INCB @#GAP            ;INCREMENT GAP #
3921 011664 122737 000020 001122  CMPB  #16., @#GAP      ;BRANCH IF ALL GAPS NOT CHECKED
3922 011672 001367              BNE   6$
3923
3924 011674 012700 000020      MOV   #16., R0        ;SETUP TO AVERAGE GAP SIZES
3925 011700 012701 001056      MOV   #GAPTBL, R1    ;SET PTR TO TABLE
3926 011704 005002              CLR   R2                ;CLEAR 'SUM' REGISTERS
3927 011706 005003              CLR   R3
3928 011710 062102              75:   ADD   (R1)+, R2        ;ADD ALL GAP SIZES TOGETHER
3929 011712 005503              ADC   R3
3930 011714 005300              DEC   R0
3931 011716 001374              BNE   7$
3932 011720 012700 000004      MOV   #4, R0          ;NOW DIVIDE BY 16.
3933 011724 006203              85:   ASR   R3                ;BY SHIFTING 4 PLACES RIGHT
3934 011726 006002              ROR   R2
3935 011730 005300              DEC   R0
3936 011732 001374              BNE   8$
3937 011734 010204              MOV   R2, R4          ;MOVE AVERAGED TIMES TO R4
3938 011736 004737 004526      JSR   PC, TIMOK       ;CHECK AVERAGED TIMES
3939 011742 000401              BR    100$
3940
3941 011744 104400              99$:   HLT
3942 011746 104000              100$: SCOPE
3943
3944      ;TEST 022-DUMMY TEST
3945      ;THIS TEST MEASURES NOTHING
3946 011750 112737 000022 001124  TST022: MOV B #22, @#TSTNUM
3947
3948      ;TEST 023-DATA TIME (200BP1)
3949      ;THIS TEST MEASURES TIME FROM 'FC REG' CHANGES TO 'RDY'=1.
3950 011756 112737 000023 001124  TST023: MOV B #23, @#TSTNUM
3951 011764 012702 012036      MOV   #3$, R2        ;SET RETURN PC FROM TIMER
3952 011770 004737 005362      JSR   PC, REWIND      ;REWIND SLAVE
(1) 011774 102437              BVS   99$            ;BRANCH IF ERROR ON REWIND
3953 011776 004337 005600      JSR   R3, TMCMD       ;WRITE 800 WORD RECORD
3954 012002 016030              .WORD WTBUF          ;SET WRITE BUFFER ADDRESS
3955 012004 176340              .WORD -800.          ;WORD COUNT
3956 012006 174700              .WORD -1600.         ;FRAME COUNT
3957 012010 000060              .WORD WFW           ;WRITE COMMAND
3958 012012 005215              INC   (R5)           ;SET 'GO' BIT
3959
3960 012014 022710 174700      15:   CMP   #-1600., (R0)   ;WAIT FOR FRAME COUNT TO CHANGE
3961 012020 001004              BNE   2$

```

```

3962 012022 032711 040000      BIT      #ERR, (R1)      ; MONITOR ERROR BIT
3963 012026 001022              BNE      99$
3964 012030 000771              BR       1$
3965
3966 012032      2$:          JSR      PC, TIMON      ; TURN TIMER ON
(1) 012032 004737 004400      3$:          TSTB     (R1)          ; WAIT FOR READY TO SET
3967 012036 105711              BMI      4$
3968 012040 100402              JMP      TIMER(R3)    ; GO TO TIMER & RETURN VIA R2
3969 012042 000163 004470      4$:          MOV      #3, R0      ; SET TO DIVIDE BY 8
3970 012046 012700 000003      5$:          ASR      R4          ; BY SHIFTING RIGHT 3 PLACES
3971 012052 006204              DEC      R0
3972 012054 005300              BNE      5$
3973 012056 001375              JSR      PC, WAITRDY
3974 012060 004737 005256              BVS      99$
3975 012064 102403              JSR      PC, TIMOK    ; CHECK TIME
3976 012066 004737 004526              BR       100$
3977 012072 000401
3978
3979 012074 104400      99$:         HLT
3980 012076 104000      100$:        SCOPE
3981
3982      ; TEST 024-DATA TIME (556BP1)
3983 012100 112737 000024 001124  TST024: MOVB   #24, @#TSTNUM
3984 012106 012702 012166              MOV      #3$, R2      ; SET RETURN PC FROM TIMER
3985 012112 004737 005362              JSR      PC, REWIND   ; REWIND SLAVE
(1) 012116 102442              BVS      99$          ; BRANCH IF ERROR ON REWIND
3986 012120 052765 000700 000032  BIS      #BP1556+NORM11, TC(R5) ; LOAD TAPE CONTROL REGISTER
3987 012126 004337 005600              JSR      R3, TCMCD    ; WRITE 2224. WORD RECORD
3988 012132 016030              .WORD   WTBUF
3989 012134 173520              .WORD   -2224.
3990 012136 167240              .WORD   -4448.
3991 012140 000060              .WORD   WFWD
3992 012142 005215              INC      (R5)        ; SET 'GO' BIT
3993
3994 012144 022710 167240      1$:          CMP      #-4448., (R0) ; BRANCH WHEN WRITING BEGINS
3995 012150 001004              BNE      2$
3996 012152 032711 040000      BIT      #ERR, (R1)  ; MONITOR ERROR BIT
3997 012156 001022              BNE      99$
3998 012160 000771              BR       1$
3999
4000 012162      2$:          JSR      PC, TIMON      ; TURN TIMER ON
(1) 012162 004737 004400      3$:          TSTB     (R1)          ; BRANCH WHEN READY SETS
4001 012166 105711              BMI      4$
4002 012170 100402              JMP      TIMER(R3)    ; GO TO TIMER & RETURN VIA R2
4003 012172 000163 004470      4$:          MOV      #3, R0      ; SET SHIFT COUNT
4004
4005 012176 012700 000003      5$:          ASR      R4
4006 012202 006204              DEC      R0
4007 012204 005300              BNE      5$
4008 012206 001375              JSR      PC, WAITRDY
4009 012210 004737 005256              BVS      99$
4010 012214 102403              JSR      PC, TIMOK    ; CHECK TIME
4011 012216 004737 004526              BR       100$
4012 012222 000401
4013
4014 012224 104400      99$:         HLT

```

```

4015 012226 104000          100%: SCOPE
4016
4017
4018 012230 112737 000025 001124 ;TEST 025-DATA TIME (800BPI)
4019 012236 012702 012316          TST025: MOVB #025,@#TSTNUM
4020 012242 004737 005362          MOV #3$,R2 ;SET RETURN PC FROM TIMER
(1) 012246 102442          JSR PC,REWIND ;REWIND SLAVE
4021 012250 052765 001300 000032 BVS 99$ ;BRANCH IF ERROR ON REWIND
4022 012256 004337 005600          BIS #BP1800+NORM11,TC(R5) ;SET 800 BPI
4023 012262 016030          JSR R3,TMCMO ;WRITE 3200. WORD RECORD
4024 012264 171600          .WORD WTBUF
4025 012266 163400          .WORD -3200.
4026 012270 000060          .WORD -6400.
4027 012272 005215          .WORD WFWO
4028          INC (R5) ;SET 'GO' BIT
4029 012274 022710 163400          1%: CMP #-6400.,(R0) ;WAIT FOR WRITING TO START
4030 012300 001004          BNE 2$
4031 012302 032711 040000          BIT #ERR,(R1) ;MONITOR ERROR BIT
4032 012306 001022          BNE 99$
4033 012310 000771          BR 1$
4034
4035 012312          2%:
(1) 012312 004737 004400          JSR PC,TIMON ;TURN TIMER ON
4036 012316 105711          3%: TSTB (R1) ;BRANCH WHEN READY SETS
4037 012320 100402          BMI 4$
4038 012322 000163 004470          JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
4039
4040 012326 012700 000003          4%: MOV #3,R0 ;SET SHIFT COUNT
4041 012332 006204          5%: ASR R4
4042 012334 005300          DEC R0
4043 012336 001375          BNE 5$
4044 012340 004737 005256          JSR PC,WAITRDY
4045 012344 102403          BVS 99$
4046 012346 004737 004526          JSR PC,TIMOK ;CHECK TIME
4047 012352 000401          BR 100%
4048
4049 012354 104400          99%: HLT
4050 012356 104000          100%: SCOPE
4051
4052          ;TEST 026-DATA TIME (1600BPI)
4053 012360 112737 000026 001124 TST026: MOVB #026,@#TSTNUM
4054 012366 105737 001131          TSTB @#NRZFLG ;BRANCH IF DRIVE 'NRZ ONLY'
4055 012372 001046          BNE TST027
4056 012374 012702 012454          MOV #3$,R2 ;SET RETURN PC FROM TIMER
4057 012400 004737 005362          JSR PC,REWIND ;REWIND SLAVE
(1) 012404 102437          BVS 99$ ;BRANCH IF ERROR ON REWIND
4058 012406 052765 002300 000032 BIS #PE1600+NORM11,TC(R5) ;SET 1600 BPI
4059 012414 004337 005600          JSR R3,TMCMO ;WRITE 3200. WORD RECORD
4060 012420 016030          .WORD WTBUF
4061 012422 171600          .WORD -3200.
4062 012424 163400          .WORD -6400.
4063 012426 000060          .WORD WFWO
4064 012430 005215          INC (R5) ;SET 'GO' BIT
4065
4066 012432 022710 163400          1%: CMP #-6400.,(R0) ;BRANCH WHEN WRITING STARTS
4067 012436 001004          BNE 2$

```

```

4068 012440 032711 040000          BIT      #ERR, (R1)          ; MONITOR ERROR BIT
4069 012444 001017                    BNE      99$
4070 012446 000771                    BR       1$
4071
4072 012450          2$:
(1) 012450 004737 004400          JSR      PC, TIMON          ; TURN TIMER ON
4073 012454 105711          3$:          TSTB     (R1)              ; BRANCH WHEN READY SETS
4074 012456 100402                    BMI      4$
4075 012460 000163 004470          JMP      TIMER(R3)         ; GO TO TIMER & RETURN VIA R2
4076
4077 012464 006204          4$:          ASR      R4                ; DIVIDE TIME BY 4
4078 012466 006204                    ASR      R4
4079 012470 004737 005256          JSR      PC, WAITRDY
4080 012474 102403                    BVS     99$
4081 012476 004737 004526          JSR      PC, TIMOK         ; CHECK TIME
4082 012502 000401                    BR       100$
4083
4084 012504 104400          99$:          HLT
4085 012506 104000          100$:         SCOPE
4086
4087          ; TEST 027-ERASE
4088          ; THIS TST MEASURES TIME FROM 'GO'=1 TO 'RDY'=1.
4089 012510 112737 000027 001124  TST027:  MOVB     #27, @#TSTNUM
4090 012516 012702 012544                    MOV      #1$, R2          ; SET RETURN PC FROM TIMER
4091 012522 004337 005600          JSR      R3, @#TMCMD
4092 012526 000000                    .WORD   0
4093 012530 000000                    .WORD   0
4094 012532 000000                    .WORD   0
4095 012534 000024                    .WORD   ERASE
4096 012536 004737 004400          JSR      PC, TIMON          ; TURN TIMER ON
4097 012542 005215                    INC      (R5)              ; SET 'GO' BIT
4098
4099 012544 105711          1$:          TSTB     (R1)              ; BRANCH WHEN READY SETS
4100 012546 100402                    BMI      2$
4101 012550 000163 004470          JMP      TIMER(R3)         ; GO TO TIMER & RETURN VIA R2
4102
4103 012554 004737 005256          2$:          JSR      PC, WAITRDY
4104 012560 102403                    BVS     99$
4105 012562 004737 004526          JSR      PC, TIMOK
4106 012566 000401                    BR       100$
4107
4108 012570 104400          99$:          HLT
4109 012572 104000          100$:         SCOPE
4110
4111          ; TEST-030 TAPE MARK
4112          ; THIS TEST MEASURES TIME FROM 'GO'=1 TO 'RDY'=1.
4113 012574 112737 000030 001124  TST030:  MOVB     #30, @#TSTNUM
4114 012602 012702 012644                    MOV      #1$, R2          ; SET RETURN PC FROM TIMER
4115 012606 004737 005444          JSR      PC, WRITE         ; WRITE A RECORD
4116 012612 005215                    INC      (R5)              ; SET 'GO' BIT
4117 012614 004737 005256          JSR      PC, WAITRDY
4118 012620 102423                    BVS     99$
4119 012622 004337 005600          JSR      R3, @#TMCMD
4120 012626 000000                    .WORD   0
4121 012630 000000                    .WORD   0
4122 012632 000000                    .WORD   0

```

Address	Word	WFMK	Comments
4123	012634 000026		
4124	012636 004737 004400	JSR PC,TIMON	;TURN TIMER ON
4125	012642 005215	INC (R5)	;SET 'GO' BIT
4126			
4127	012644 105711	15: TSTB (R1)	;BRANCH WHEN READY SETS
4128	012646 100402	BMI 25	
4129	012650 000163 004470	JMP TIMER(R3)	;GO TO TIMER & RETURN VIA R2
4130			
4131	012654 004737 005256	25: JSR PC, WAITRDY	
4132	012660 102403	BVS 995	
4133	012662 004737 004526	JSR PC,TIMOK	
4134	012666 000401	BR 1005	
4135			
4136	012670 104400	995: HLT	
4137	012672	1005:	
(1)	012672 004737 005362	JSR PC, REWIND	;REWIND SLAVE
(1)	012676 102774	BVS 995	;BRANCH IF ERROR ON REWIND
4138	012700 104000	SCOPE	
4139			

4141	012702	012700	000012		FINISH:	MOV	#10, R0		;SET LINE FEED COUNT
4142	012706	000004	001376		15:	TYPE, CRLF			
4143	012712	005300				DEC	R0		
4144	012714	001374				BNE	15		
4145	012716	032777	000100	166056		BIT	#SW06, @SWR		
4146	012724	001410				BEQ	25		
4147	012726	113700	001006			MOVB	@DRVNUM, R0		
4148	012732	113701	001007			MOVB	@SLVNUM, R1		
4149	012736	113702	001010			MOVB	@SLVPTR, R2		
4150	012742	000137	007274			JMP	@TYPHDR		
4151	012746	105237	001007		25:	INCB	@SLVNUM		;SET NEXT SLAVE #
4152	012752	005237	001010			INC	@SLVPTR		;AND ITS POINTER
4153	012756	122737	000010	001007		CMPB	#8, @SLVNUM		;BRANCH IF LAST SLAVE (7)
4154	012764	001402				BEQ	35		
4155	012766	000137	007030			JMP	@BEGIN		;BEGIN TEST ON NEXT SLAVE
4156	012772	105037	001007		35:	CLRB	@SLVNUM		;SET SLAVE #0
4157	012776	105237	001006			INCB	@DRVNUM		;AND INCREMENT DRIVE #
4158	013002	122737	000010	001006		CMPB	#8, @DRVNUM		;AND CHECK IF LAST DRIVE
4159	013010	001402				BEQ	END		
4160	013012	000137	007030			JMP	@BEGIN		
4161									
4162	013016	105737	001127		END:	TSTB	@UNTFND		;BRANCH IF A UNIT WAS FOUND
4163	013022	001004				BNE	15		
4164	013024	000004	014173			TYPE, E. UNIT			
4165	013030	000137	013060			JMP	@ENDPAS		
4166	013034	023737	000042	000046	15:	CMP	@42, @46		;ACT11 AUTO MODE?
4167	013042	001401				BEQ	25		;YES
4168	013044	000000				HALT			
4169	013046	004737	001772		25:	JSR	PC, CKSWR		;CHECK FOR CNTL G
4170	013052	000005				RESET			
4171	013054	000137	005750			JMP	@INIT		;RESTART
4172	013060	005237	001000		ENDPAS:	INC	@PASS		;+1 TO PASS CNT
4173	013064	013700	000042			MOV	@42, R0		
4174	013070	001405				BEQ	RETURN		;BRANCH IF NO MONITOR
4175	013072	000005				RESET			
4176	013074	004710			ENDAD:	JSR	PC, (R0)		;EXIT TO MONITOR
4177	013076	000240				NOP			
4178	013100	000240				NOP			
4179	013102	000240				NOP			
4180	013104	000137	005750		RETURN:	JMP	@INIT		

```

4182 ;SKEW TAPE TIMING TESTS
4183 ;THE FOLLOWING TESTS REQUIRE A SPECIALLY WRITTEN 800 BPI SKEW TAPE
4184 013110 012737 013116 001004 SKEWTST: MOV #TST031, @#SCPADR ;SET SCOPE POINTER
4185
4186 ;TEST 031- SKEW TAPE SPEED TEST-FORWARD
4187 ;THIS TEST READS 32" OF TAPE (26400. -800. = 25600. FRAMES), THEN
4188 ;DIVIDES TIME BY 32. TO GET TIME TO READ 1" (800. FRAMES) OF TAPE.
4189 013116 112737 000031 001124 TST031: MOV #31, @#TSTNUM
4190 013124 012702 013202 MOV #25, R2 ;SET RETURN PC FROM TIMER
4191 013130 004737 005362 JSR PC, REWIND ;REWIND SLAVE
(1) 013134 102441 BVS 99% ;BRANCH IF ERROR ON REWIND
4192 013136 052765 001300 000032 BIS #BP1800+NORM11, TC(R5) ;SET 800 BPI
4193 013144 052765 000010 000010 BIS #BA1, CS2(R5) ;INHIBIT BUS ADDRESS INCREMENT
4194 013152 004337 005600 JSR R3, @#TMCMD ;READ 32" OF TAPE-FORWARD
4195 013156 016030 . WORD RDBUF
4196 013160 177777 . WORD -1.
4197 013162 063440 10%: . WORD 26400. ;FRAME COUNT
4198 013164 000070 . WORD RDFWD
4199 013166 005215 INC (R5) ;SET 'GO' BIT
4200
4201 013170 022710 001440 15: CMP #800., (R0) ;WAIT FOR FIRST 800 FRAMES
4202 013174 101375 BHI 15 ;TO BE READ
4203
4204 013176 004737 004400 JSR PC, TIMON ;TURN TIMER ON
4205 013202 023710 013162 25: CMP @#10%, (R0) ;WAIT FOR READING TO FINISH
4206 013206 103402 BLO 3%
4207 013210 000163 004470 JMP TIMER(R3) ;GO TO TIMER & RETURN VIA R2
4208
4209 013214 012700 000005 35: MOV #5, R0 ;DIVIDE TIME BY 32.
4210 013220 006204 45: ASR R4
4211 013222 005300 DEC R0
4212 013224 001375 BNE 4%
4213 013226 004737 005220 JSR PC, RHINIT ;INIT DRIVE
4214 013232 004737 004526 JSR PC, TIMOK ;CHECK TIME
4215 013236 000401 BR 100%
4216
4217 013240 104400 99%: HLT
4218 013242 104000 100%: SCOPE
4219
4220 ;TEST 032-SKEW TAPE SPEED TEST-REVERSE
4221 ;THIS TEST READS FORWARD 40" (32000. FRAMES) OF TAPE, THEN READS REVERSE
4222 ;32" (26400. -800. = 25600. FRAMES) OF TAPE. THE TIME IS THEN DIVIDED BY
4223 ;32. TO GET TIME TO READ 1" (800. FRAMES) OF TAPE.
4224 013244 112737 000032 001124 TST032: MOV #32, @#TSTNUM
4225 013252 012702 013400 MOV #35, R2 ;SET RETURN PC FROM TIMER
4226 013256 004737 005362 JSR PC, REWIND ;REWIND SLAVE
(1) 013262 102465 BVS 99% ;BRANCH IF ERROR ON REWIND
4227 013264 052765 001300 000032 BIS #BP1800+NORM11, TC(R5)
4228 013272 052765 000010 000010 BIS #BA1, CS2(R5)
4229 013300 004337 005600 JSR R3, @#TMCMD ;READ FORWARD 32000. FRAMES
4230 013304 016030 . WORD RDBUF
4231 013306 177777 . WORD -1. ;WORD COUNT
4232 013310 076400 10%: . WORD 32000. ;FRAME COUNT
4233 013312 000070 . WORD RDFWD ;READ FORWARD
4234 013314 005215 INC (R5) ;SET 'GO' BIT
4235

```

4236	013316	023710	013310	15:	CMP	@#10\$, (R0)	
4237	013322	101375			BHI	15	
4238							
4239	013324	004737	005220		JSR	PC, RHINIT	; INIT DRIVE
4240	013330	004737	004764		JSR	PC, DELAY	; WAIT FOR TAPE MOTION TO STOP
4241	013334	052765	001300	000032	BIS	#BP1800+NORM11, TC(R5)	; SET 800 BPI
4242	013342	052765	000010	000010	BIS	#BA1, CS2(R5)	; INHIBIT BUS ADDRESS INCREMENT
4243	013350	004337	005600		JSR	R3, @#TMCMD	; READ REVERSE 32" OF TAPE
4244	013354	016030			. WORD	RDBUF	; READ BUFFER
4245	013356	177777			. WORD	-1.	; WORD COUNT
4246	013360	063440		115:	. WORD	26400.	; FRAME COUNT
4247	013362	000076			. WORD	RDREV	; READ REVERSE
4248	013364	005215			INC	(R5)	; SET 'GO' BIT
4249							
4250	013366	022710	001440	25:	CMP	#800., (R0)	; WAIT FOR FIRST 800 FRAMES
4251	013372	101375			BHI	25	; TO BE READ
4252							
4253	013374	004737	004400		JSR	PC, TIMON	; TURN TIMER ON
4254	013400	023710	013360	35:	CMP	@#11\$, (R0)	; WAIT FOR ALL FRAMES TO BE READ
4255	013404	103402			BLO	45	
4256	013406	000163	004470		JMP	TIMER(R3)	; GO TO TIMER & RETURN VIA R2
4257							
4258	013412	012700	000005	45:	MOV	#5, R0	; DIVIDE TIME BY 32.
4259	013416	006204		55:	ASR	R4	
4260	013420	005300			DEC	R0	
4261	013422	001375			BNE	55	
4262	013424	004737	005220		JSR	PC, RHINIT	
4263	013430	004737	004526		JSR	PC, TIMOK	
4264	013434	000401			BR	100\$	
4265							
4266	013436	104400		99%:	HLT		
4267	013440			100%:			
(1)	013440	004737	005362		JSR	PC, .REWIND	; REWIND SLAVE
(1)	013444	102774			BVS	99\$; BRANCH IF ERROR ON REWIND
4268	013446	104000			SCOPE		
4269							
4270	013450	000137	012702		JMP	@#FINISH	
4271							
4272							
4273							

```
4275
4276
4277 013454 005015 052524 033061
      013462 042040 044522 042526
      013470 043040 047125 052103
      013476 047511 020116 044524
      013504 042515 020122 042050
      013512 052132 042125 042055
      013520 000051
4278 013522 005015 054524 042520
      013530 043040 051111 052123
      013536 040440 042104 042522
      013544 051523 047440 020106
      013552 047503 052116 047522
      013560 046114 051105 020040
      013566 000
4279 013567 124 050131 020105
      013574 046524 031060 042040
      013602 044522 042526 021440
      013610 051447 052040 020117
      013616 042502 052040 051505
      013624 042524 020104 000
4280 013631 106 051117 052040
      013636 030115 020062 051104
      013644 053111 020105
4281 013650 026460 052040 050131
      013656 020105 046123 053101
      013664 020105 023443 020123
      013672 047524 041040 020105
      013700 042524 052123 042105
      013706 000040
4282 013710 050123 042505 020104
      013716 042524 052123 020123
      013724 047117 054514 020077
      013732 054450 051505 047057
      013740 020117 020075 027461
      013746 024460 000
4283 013751 116 055122 047440
      013756 046116 037531 024040
      013764 042531 027523 047516
      013772 036440 030440 030057
      014000 000051
4284 014002 005015 047105 020104
      014010 043117 052040 050101
      014016 006505 000012
4285
4286
4287 014022 005015 051124 050101
      014030 042520 020104 047524
      014036 032040 000
4288 014041 116 020117 047503
      014046 052116 047522 046114
      014054 051105 040440 020124
      014062 042101 051104 051505
      014070 020123 050123 041505
      014076 043111 042511 006504
```

SBTTL PROGRAM MESSAGES
; OPERATOR INSTRUCTIONS
M. NAM: .ASCIZ <CR><LF>'TU16 DRIVE FUNCTION TIMER (DZTUD-D)'

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

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

I. SLVS: .ASCII 'FOR TMO2 DRIVE '

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

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

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

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

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

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

4289	014104	000012				
	014106	046524	031060	042040	E. NDRV: .ASCIZ	'TMO2 DRIVE '
	014114	044522	042526	000040		
4290	014122	051104	053111	020105	E. NSLV: .ASCII	'DRIVE '
4291	014130	020060	046123	053101	E. DRV: .ASCII	'O SLAVE '
	014136	020105				
4292	014140	020060	047516	020124	E. NAVA: .ASCIZ	'O NOT AVAILABLE FOR TEST'<CR><LF>
	014146	053101	044501	040514		
	014154	046102	020105	047506		
	014162	020122	042524	052123		
	014170	005015	000			
4293	014173	101	046114	051440	E. UNIT: .ASCIZ	'ALL SELECTED TMO2/TU16 UNITS TESTED'<CR><LF>
	014200	046105	041505	042524		
	014206	020104	046524	031060		
	014214	052057	030525	020066		
	014222	047125	052111	020123		
	014230	042524	052123	042105		
	014236	005015	000			
4294	014241	123	043117	020124	E. SFT: .ASCIZ	'SOFT ERROR (DATA)'<CR><LF>
	014246	051105	047522	020122		
	014254	042050	052101	024501		
	014262	005015	000			
4295	014265	124	051505	020124	E. HDR: .ASCIZ	'TEST # '
	014272	020043	000			
4296	014275	040	042504	044526	E. HDR1: .ASCII	' DEVICE ERROR'<CR><LF>
	014302	042503	042440	051122		
	014310	051117	005015			
4297	014314	051503	004461	041527	.ASCIZ	'CS1'<HT>'WC'<HT>'BA'<HT>'FC'<HT>'CS2'<HT>'DS'<HT>'ER'<HT>'TC'<CR><LF>
	014322	041011	004501	041506		
	014330	041411	031123	042011		
	014336	004523	051105	052011		
	014344	006503	000012			
4298	014350	047440	052125	047440	E. HDR2: .ASCIZ	' OUT OF RANGE ERROR'<CR><LF>
	014356	020106	040522	043516		
	014364	020105	051105	047522		
	014372	006522	000012			
4299	014376	005015	044524	042515	E. TIMOV: .ASCIZ	<CR><LF>'TIMER OVERFLOWED'<CR><LF>
	014404	020122	053117	051105		
	014412	046106	053517	042105		
	014420	005015	000			
4300	014423	015	052012	046511	E. TIMEX: .ASCIZ	<CR><LF>'TIME EXPIRED WAITING FOR RDY'<CR><LF>
	014430	020105	054105	044520		
	014436	042522	020104	040527		
	014444	052111	047111	020107		
	014452	047506	020122	042122		
	014460	006531	000012			
4301	014464	043440	050101	021440	E. GAP: .ASCIZ	' GAP # '
	014472	000040				
4302						
4303						
4304	014474	025052	025052	025052	.TIME DOCUMENT LINES	
	014502	025052	025052	025052	L. HDR1: .ASCIZ	'*****'
	014510	025052	025052	025052		
	014516	025052	025052	025052		
	014524	025052	025052	025052		
	014532	025052	025052	025052		

	014540	025052	025052	025052		
	014546	025052	025052	025052		
	014554	025052	025052	025052		
	014562	025052	025052	025052		
	014570	025052	025052	025052		
	014576	025052	025052	025052		
	014604	005015	000			
4305	014607	052	052040	030115	L. HDR2: . ASCII	'* TMO2 DRIVE FUNCTION TIMES- DRIVE # '
	014614	020062	051104	053111		
	014622	020105	052506	041516		
	014630	044524	047117	052040		
	014636	046511	051505	020055		
	014644	051104	053111	020105		
	014652	020043				
4306	014654	020060	046123	053101	L. DRV: . ASCII	'0 SLAVE # '
	014662	020105	020043			
4307	014666	020060	040		L. SLV: . ASCII	'0 '
4308	014671	071	041440	040510	L. CHAN: . ASCIIZ	'9 CHAN. SER # '
	014676	027116	051440	051105		
	014704	021440	000040			
4309	014710	006440	025012	005015	L. HDR3: . ASCII	'<CR><LF>'*<CR><LF>'
4310	014716	020052	052506	041516	. ASCIIZ	'* FUNCTION'<HT><HT>'TIME (SPECIFICATION)'<HT>'TIME (ACTUAL)'<CR><LF>'
	014724	044524	047117	004411		
	014732	044524	042515	051450		
	014740	042520	044503	044506		
	014746	040503	044524	047117		
	014754	004451	044524	042515		
	014762	040450	052103	040525		
	014770	024514	005015	000		
4311						
4312	014775	122	047101	042507	L. RNG: . ASCIIZ	'RANGE=<'
	015002	036075	000			
4313	015005	101	052103	040525	L. ACT: . ASCIIZ	'ACTUAL='
	015012	036514	000			
4314						
4315						
4316	015015	052	053440	044522	A. T001: . ASCIIZ	'* WRITE FROM BOT'<HT>'
	015022	042524	043040	047522		
	015030	020115	047502	004524		
	015036	000				
4317	015037	052	053440	044522	A. T002: . ASCIIZ	'* WRITE START'<HT><HT>'
	015044	042524	051440	040524		
	015052	052122	004411	000		
4318	015057	052	053440	044522	A. T003: . ASCIIZ	'* WRITE SHUTDOWN'<HT>'
	015064	042524	051440	052510		
	015072	042124	053517	004516		
	015100	000				
4319	015101	052	053440	044522	A. T004: . ASCIIZ	'* WRITE SETTLEDOWN'<HT>'
	015106	042524	051440	052105		
	015114	046124	042105	053517		
	015122	004516	000			
4320	015125	052	051040	040505	A. T005: . ASCIIZ	'* READ FROM BOT'<HT><HT>'
	015132	020104	051106	046517		
	015140	041040	052117	004411		
	015146	000				
4321	015147	052	051040	040505	A. T006: . ASCIIZ	'* READ START'<HT><HT>'

	015154	020104	052123	051101	
	015162	004524	000011		
4322	015166	020052	042522	042101	A. T007: .ASCIZ '* READ SHUTDOWN' <HT><HT>
	015174	051440	052510	042124	
	015202	053517	004516	000011	
4323	015210	020052	042522	042101	A. T010: .ASCIZ '* READ SETTLEDOWN' <HT>
	015216	051440	052105	046124	
	015224	042105	053517	004516	
	015232	000			
4324	015233	052	051040	040505	A. T011: .ASCIZ '* READ REV START' <HT>
	015240	020104	042522	020126	
	015246	052123	051101	004524	
	015254	000			
4325	015255	052	051040	040505	A. T012: .ASCIZ '* READ REV SHUTDOWN' <HT>
	015262	020104	042522	020126	
	015270	044123	052125	047504	
	015276	047127	000011		
4326	015302	020052	042522	042101	A. T013: .ASCIZ '* READ REV SETTLEDOWN' <HT>
	015310	051040	053105	051440	
	015316	052105	046124	042105	
	015324	053517	004516	000	
4327	015331	052	052040	051125	A. T014: .ASCIZ '* TURN AROUND DELAY F-R' <HT>
	015336	020116	051101	052517	
	015344	042116	042040	046105	
	015352	054501	043040	051055	
	015360	000011			
4328	015362	020052	052524	047122	A. T015: .ASCIZ '* TURN AROUND DELAY R-F' <HT>
	015370	040440	047522	047125	
	015376	020104	042504	040514	
	015404	020131	026522	004506	
	015412	000			
4329	015413	052	043440	050101	A. T016: .ASCIZ '* GAP SIZE-STOP HALF' <HT>
	015420	051440	055111	026505	
	015426	052123	050117	044040	
	015434	046101	004506	000	
4330	015441	052	043440	050101	A. T017: .ASCIZ '* GAP SIZE-START HALF' <HT>
	015446	051440	055111	026505	
	015454	052123	051101	020124	
	015462	040510	043114	000011	
4331	015470	020052	040507	020120	A. T020: .ASCIZ '* GAP SIZE-INTERRECOPD' <HT>
	015476	044523	042532	044455	
	015504	052116	051105	042522	
	015512	047503	042122	000011	
4332	015520	020052	040507	020120	A. T021: .ASCIZ '* GAP CONSISTANCY' <HT>
	015526	047503	051516	051511	
	015534	040524	041516	004531	
	015542	000			
4333	015543	052	042040	052101	A. T023: .ASCIZ '* DATA TIME-200BPI' <HT>
	015550	020101	044524	042515	
	015556	031055	030060	050102	
	015564	004511	000		
4334	015567	052	042040	052101	A. T024: .ASCIZ '* DATA TIME-556BPI' <HT>
	015574	020101	044524	042515	
	015602	032455	033065	050102	
	015610	004511	000		
4335	015613	052	042040	052101	A. T025: .ASCIZ '* DATA TIME-800BPI' <HT>

	015620	020101	044524	042515				
	015626	034055	030060	050102				
	015634	004511	000					
4336	015637	052	042040	052101	A. T026:	. ASCIZ	'* DATA TIME-1600BPI'	<HT>
	015644	020101	044524	042515				
	015652	030455	030066	041060				
	015660	044520	000011					
4337	015664	020052	051105	051501	A. T027:	. ASCIZ	'* ERASE GAP TIME'	<HT>
	015672	020105	040507	020120				
	015700	044524	042515	000011				
4338	015706	020052	051127	052111	A. T030:	. ASCIZ	'* WRITE FILE MARK'	<HT>
	015714	020105	044506	042514				
	015722	046440	051101	004513				
	015730	000						
4339	015731	052	052040	050101	A. T031:	. ASCIZ	'* TAPE SPEED-FWD'	<HT>
	015736	020105	050123	042505				
	015744	026504	053506	004504				
	015752	000						
4340	015753	052	052040	050101	A. T032:	. ASCIZ	'* TAPE SPEED-REV'	<HT>
	015760	020105	050123	042505				
	015766	026504	042522	004526				
	015774	000						
4341								
4342	015775	015	057012	000107	L. CNTG:	. ASCIZ	<CR><LF>' G'	
4343	016002	005015	053523	036522	L. SWR:	. ASCIZ	<CR><LF>'SWR='	
	016010	000						
4344	016011	040	047040	053505	L. NEW:	. ASCIZ	' NEW= '	
	016016	020075	000					
4345	016021	015	037412	005015	L. QUEST:	. ASCIZ	<CR><LF>'?'<CR><LF>	
	016026	000						
4346		016030				. EVEN		
4347		016030				ROBUF=.		
4348		016030				WTBUF=.		
4349	016030	000200				. BLKW	128	
4350		000001				. END		

A	010704	CNTRLO=	000017	E. DRV	014130	L. HDR1	014474	RDSW	001770
ACCL	= 100000	CNTRLU=	000025	E. GAP	014464	L. HDR2	014607	RDY	= 000200
ANGTAB	001414	CNVDEC	002732	E. HDR	014265	L. HDR3	014710	READ	005462
AS	= 000016	CNVOC	002624	E. HDR1	014275	L. NEW	016011	RESVEC=	000010
ASFLG	001132	CNVTA0	003262	E. HDR2	014350	L. QUES	016021	RETURN	013104
ATA	= 100000	CNVTD	002744	E. NAVA	014140	L. RNG	014775	REVRD	005500
ATIME	001014	CNVTO	002636	E. NCON	014041	L. SLV	014666	RHINIT	005220
ATIMTB	001016	COUNT	001766	E. NDRV	014106	L. SWR	016002	RMR	= 000004
A. T001	015015	CR	= 000015	E. NSLV	014122	MCPE	= 020000	RWD	= 000006
A. T002	015037	CRLF	001376	E. SFT	014241	MDPE	= 000400	RWDOFF=	000002
A. T003	015057	CS1TM	= 002000	E. TIME	014423	MMVEC	= 000250	R10	=%000000
A. T004	015101	CS1	= 000000	E. TMO	014376	MOL	= 010000	R11	=%000001
A. T005	015125	CS2	= 000010	E. TRP4	014022	MR	= 000024	R12	=%000002
A. T006	015147	DASH	001407	E. UNIT	014173	MXF	= 001000	R13	=%000003
A. T007	015166	DB	= 000022	FC	= 000006	M. EOT	014002	R14	=%000004
A. T010	015210	DCONST	003040	FCE	= 001000	M. NAM	013454	R15	=%000005
A. T011	015233	DELAY	004764	FINISH	012702	NAMPTR	001674	SC	= 100000
A. T012	015255	DELAYV	005014	FMT	= 000020	NED	= 010000	SCOPE	= 104000
A. T013	015302	DELTIM	001116	FPEVEC=	000244	NEF	= 004000	SCPADR	001004
A. T014	015331	DIGTAB	001134	FRMCNT=	177400	NEM	= 004000	SDWN	= 000020
A. T015	015362	DIVIDE	005052	FWDSPC	005516	NOP	= 000000	SKEWTS	013110
A. T016	015413	DLT	= 100000	GAP	001122	NORM11=	000300	SLA	= 000001
A. T017	015441	DPR	= 000400	GAPOK	004636	NRZFLG	001131	SLAVES	006434
A. T020	015470	DRIVES	006174	GAPTBL	001056	NSG	= 000400	SLR	= 177774
A. T021	015520	DRVAVA	005142	GO	= 000001	OCTALO	001120	SLVAVA	005170
A. T023	015543	DRVNUM	001006	GTIMTB	001574	ODIGIT	001146	SLVNUM	001007
A. T024	015567	DRVTBL	001156	HLT	= 104400	OPI	= 020000	SLVPTR	001010
A. T025	015613	DRY	= 000200	HT	= 000011	OR	= 000200	SLVTBL	001166
A. T026	015637	DRYCLR=	000010	IDB	= 000010	OSC	= 000100	SN	= 000030
A. T027	015664	DS	= 000012	IE	= 000100	OUT	002242	SNPT	005620
A. T030	015706	DT	= 000026	ILF	= 000001	OUTBUF=	005750	SPACE	001412
A. T031	015731	DTE	= 010000	ILR	= 000002	OUTGAP	003150	SPACE2	001411
A. T032	015753	DVA	= 004000	INBUF	001266	OUTSPC	003054	SPCFWD=	000030
A16	= 000400	DVO	= 000000	INCVAE=	000100	PARVEC=	000114	SPCREV=	000032
A17	= 001000	DV1	= 000001	INIT	005750	PASS	001000	SPR	= 002000
BA	= 000004	DV2	= 000002	IOTVEC=	000020	PAT	= 000020	SSC	= 000100
BA1	= 000010	DV3	= 000003	IR	= 000100	PEFLRC=	000200	STIMTB	001420
BEGIN	007030	DV4	= 000004	ITCNT	001123	PES	= 000040	STKPTR=	000600
BELL	001405	DV5	= 000005	I. DRV	013650	PE1600=	002000	SUSWR	005754
BKSLSH	001401	DV6	= 000006	I. DRVS	013567	PFVEC	= 000024	SWR	001002
BOT	= 000002	DV7	= 000007	I. NRZ	013751	PGE	= 002000	SWREG	000176
BP1200=	000000	ECHO	001403	I. REG	013522	PIP	= 020000	SW06	= 000100
BP1556=	000400	EMTVEC=	000030	I. SKEW	013710	PIRQ	= 177772	SW07	= 000200
BP1800=	001000	END	013016	I. SLVS	013631	PIRVEC=	000240	SW08	= 000400
BPTVEC=	000014	ENDAD	013074	LF	= 000012	PLKCSR=	172540	SW09	= 001000
CDM11	= 000320	ENDPAS	013060	LKS	= 177546	PLKVEC=	000104	SW10	= 002000
CHKDRV	006334	EOT	= 002000	LKVEC	= 000100	PRGFLG	001126	SW11	= 004000
CHKSLV	006640	ER	= 000014	LPB	= 177516	PSEL	= 002000	SW13	= 020000
CH7	= 010000	ERASE	= 000024	LPS	= 177514	PSW	= 177776	SW14	= 040000
CKSWR	001772	ERFLG	001125	L. ACT	015005	PUBLIS	003350	SW15	= 100000
CLR	= 000040	ERR	= 040000	L. CHAN	014671	RDBUF	= 016030	TAP	= 040000
CNTLU	002032	ERRTRP	003652	L. CNTG	015775	RDFWD	= 000070	TBITVE=	000014
CNTRLC=	000003	ERRVEC=	000004	L. DRV	014654	RDEV	= 000076	TC	= 000032

TCRLF	002362	TRAPVE=	000034	TST021	011444	TYPHDR	007274	SCNTRL	002327
TEMPST	001764	TRE =	040000	TST022	011750	TYPOCT	002632	SCRLF	002330
TIB	001762	TRTVEC=	000014	TST023	011756	UBREAK=	177770	SFILL	002317
TIMER	004470	TSTNUM	001124	TST024	012100	UNS =	040000	SHT =	000011
TIMERR	004500	TST001	007422	TST025	012230	UNTFND	001127	\$NULL	002316
TIMERO	004514	TST002	007506	TST026	012360	UPE =	020000	\$TKFLG	002321
TIMER1	004444	TST003	007564	TST027	012510	WAITRD	005256	STPB	002324
TIMOK	004526	TST004	007654	TST030	012574	WAITTI	005260	STPFLG	002320
TIMON	004400	TST005	007762	TST031	013116	WC =	000002	STPS	002322
TKB =	177562	TST006	010046	TST032	013244	WCE =	040000	. =	016430
TKISR	003610	TST007	010132	TTIN	002244	WCHKF =	000050	. HLT	003654
TKS =	177560	TST010	010232	TTIN1	002264	WCHKR =	000056	. INPUT	003476
TKVEC =	000060	TST011	010352	TTIN2	002300	WFMK =	000026	. RESTO	002602
TMBASE	001012	TST012	010450	TYPDEC	002740	WFWD =	000060	. REWIN	005362
TMCMD	005600	TST013	010560	TYPE =	000004	WRDCNT=	177600	. SAVE	002560
TMCS1 =	172440	TST014	010720	TYPE1	002354	WRITE	005444	. SCOPE	004144
TMK =	000004	TST015	011012	TYPE2	002402	WRL =	004000	. TYPE	002334
TPB =	177566	TST016	011120	TYPE3	002410	WRT. BK	005536		
TPS =	177564	TST017	011214	TYPE4	002414	WTBUF =	016030		
TPVEC =	000064	TST020	011324	TYPFLG	001130	\$CHARC	002326		

. ABS. 016430 000

ERRORS DETECTED: 0

DZTUDD, DZTUDD=DZTUDD. SML, DZTUDD. P11
RUN-TIME: 59.3 SECONDS
RUN-TIME RATIO: 581/15=37.4
CORE USED: 20K (39 PAGES)