

RH70,RS04

MAINTENANCE MODE
CERSDDO

AH-8026D-MC

APR 1978

COPYRIGHT ©75-78

digital

FICHE 1 OF 1

MADE IN USA

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

.REM !

IDENTIFICATION

PRODUCT CODE: AC-8024D-MC
PRODUCT NAME: CERSDDO RH70-R504 MAINT MODE
(RH70-R504 MAINTENANCE MODE DIAGNOSTIC)
DATE: JANUARY , 1978
MAINTAINER: DIAGNOSTIC GROUP
AUTHORS: STANLEY HARACKIEWICZ

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.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1975, 1978 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

C01

TABLE OF CONTENTS

CONTENTS

1.	ABSTRACT
1.1	DESIGN PHILOSOPHY
2.	REQUIREMENTS
2.1	EQUIPMENT
2.3	PRELIMINARY PROGRAMS
3.	LOADING PROCEDURE
4.	STARTING PROCEDURE
4.1	CONTROL SWITCH SETTINGS
4.2	STARTING ADDRESS
4.3	PROGRAM AND/OR OPERATING PROCEDURE
5	OPERATIONAL SWITCH SETTINGS
5.1	SUBROUTINE ABSTRACT
6.	ERRORS
7.	RESTRICTIONS
8.	MISCELLANEOUS
8.1	EXECUTION TIME
8.2	STACK POINTER
9.	TEST DESCRIPTION

DESCRIPTION

1. ABSTRACT

THIS DIAGNOSTIC WILL LET THE OPERATOR SELECT ONE OF TWO MODES OF OPERATION. THE OPERATOR MAY SELECT WHICH DRIVE HE WANTS TESTED OR HE CAN LET THE PROGRAM SEQUENCE THROUGH ALL THE DRIVES ON THE SYSTEM.

THE FIRST PART OF THIS DIAGNOSTIC WILL TEST THE DRIVE REGISTERS ASSOCIATED WITH THE DRIVE UNDER TEST. THE PROGRAM WILL ALSO TEST THE RH CONTROLLER REGISTERS TO CONFIRM THAT, FOR THE MOST PART, THE CONTROLLER IS WORKING CORRECTLY.

THE SECOND PART OF THIS DIAGNOSTIC WILL TEST THE DRIVE II. "MAINTENANCE MODE".

THE R504 HAS BEEN DESIGNED WITH BUILT-IN TEST CAPABILITIES. THIS "MAINTENANCE MODE" TEST CAPABILITY ISOLATES THE DIGITAL ELECTRONICS FROM THE ANALOG AND ALLOWS INDEPENDENT TESTING OF THE DIGITAL LOGIC. THEREFORE, FAILURES LOCATED ENTIRELY IN THE LOGIC CAN BE SEPARATED FROM FAILURES OCCURRING IN THE ANALOG ELECTRONICS OR THE HEAD/DISK SUBASSEMBLY.

1.1 DESIGN PHILOSOPHY

BY SETTING BIT 00 IN THE MAINTENANCE REGISTER, THE MAINTENANCE MODE LOGIC IS ENABLED, AND THE REMAINING READ/WRITE BITS IN THE MAINTENANCE REGISTER ARE SUBSTITUTED FOR THE CORRESPONDING SIGNALS NORMALLY ORIGINATING FROM THE HEAD/DISK SUBASSEMBLY. THE READ-ONLY BITS IN THE MAINTENANCE REGISTER REFLECT THE STATES OF MAJOR SIGNALS DURING DRIVE OPERATION. BY SETTING AND CLEARING THE READ/WRITE BITS IN PREDETERMINED SEQUENCES AND SIMULTANEOUSLY MONITORING THE READ-ONLY BITS, IT IS POSSIBLE TO VERIFY THE OPERATION OF ALL OF THE DRIVE'S LOGIC. THIS INCLUDES ALL DRIVE TIMING AS WELL AS THE LOGIC ASSOCIATED WITH READING AND WRITING DATA.

--CAUTION--

A THOROUGH UNDERSTANDING OF THE R504 LOGIC IS REQUIRED TO UTILIZE THIS DIAGNOSTIC EFFECTIVELY. REFER TO SECTIONS 2 AND 3 OF THE "R504 DECDISK SERVICE MANUAL" (DEC-DD-HRS4A-A-D) FOR DESCRIPTIONS OF THE DRIVE LOGIC.

2. REQUIREMENTS

2.1 EQUIPMENT

DESCRIPTION

PDP-11 WITH A MINIMUM OF 8K OF MEMORY AND AN RH11 CONTROLLER WITH A R504 DISK.

2.3 PRELIMINARY PROGRAMS

NONE

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR ABS TAPES.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE SECTION 5 (ALL DOWN FOR WORST CASE TESTING)

4.2 STARTING ADDRESSES

4.3 PROGRAM AND/OR OPERATOR ACTION

LOAD PROGRAM INTO MEMORY USING ABS LOADER.

STARTING ADDRESSES

1. STARTING ADDRESS 200

A. SET SWITCHES (SEE SECTION 5)

B. PRESS START

C. THE PROGRAM WILL TYPE:

TEST ALL DRIVES? (Y OR N)

D. IF THE OPERATOR TYPES "Y" THE PROGRAM WILL TEST ALL R504 DRIVES ON THE SYSTEM

E. IF THE OPERATOR TYPES "N" THE PROGRAM WILL TYPE

DESCRIPTION

TYPE UNIT

THE PROGRAM WILL ONLY TEST THAT DRIVE. THE PROGRAM
WILL THEN TYPE:

"ALL ERROR LIGHTS ON SELECTED UNIT SHOULD
BE ON - CHECK - THEN HIT CONT"

THE OPERATOR SHOULD CHECK THESE LIGHTS TO MAKE SURE
THAT THEY ARE ALL ON - THEN HIT CONTINUE THE PROGRAM
WILL THEN START TESTING THE UNIT THAT WAS SELECTED.

2. STARTING ADDRESS 220

- A. SET SWITCHES (SEE SECTION 5)
- B. PRESS START
- C. THE PROGRAM WILL THEN TEST ALL R504 DRIVES ON THE
SYSTEM.

5. OPERATIONAL SWITCH SETTINGS

SWITCH SETTINGS ARE:

SW<15> = 1 HALT ON ERROR
 SW<14> = 1 LOOP ON TEST
 SW<13> = 1 INHIBIT TYPEOUTS
 SW<12> = 1 TYPEOUT ALL ERRORS IN DATA COMPARE ROUTINE
 SW<11> = 1 RUN MAINTENANCE MODE VERIFY TEST
 SW<10> = 1 BELL ON ERROR
 0 BELL ON PASS COMPLETE
 SW<09> = 1 LOOP ON ERROR
 SW<08> = 1 LOOP ON TEST IN SW<7:0>

5.1 SUBROUTINE ABSTRACTS

THIS PROGRAM USES TRAP INSTRUCTIONS TO EXECUTE CLOCKING AND REGISTER
CHECKING. THE TRAP INSTRUCTIONS THAT WE USED, ARE LISTED BELOW WITH A
BREIF DESCRIPTION OF WHAT EACH ONE DOES.

5.1.1 CLRDK

TRAPS TO A TAG CALLED ".CLRDK". THIS ROUTINE CLEARS ALL REGISTERS BY
SETTING THE "CLEAR BIT" IN RSCS2. (MOV#40, JRHCS2) THE NUMBER OF THE
UNIT UNDER TEST IS THEN RELOADED INTO RSCS2 AND THE PROGRAM RETURNS TO

DESCRIPTION

THE NEXT INSTRUCTION FOLLOWING THE CLRDK INSTRUCTION.

5.1.2 MRDMD

TRAPS TO A TAG CALLED ".MRDMD". THIS ROUTINE PUTS THE DRIVE INTO MAINTENANCE MODE BY LOADING #000001 INTO RSMR AND THEN RETURNS TO THE NEXT INSTRUCTION FOLLOWING THE MRDMD INSTRUCTION.

5.1.3 MRINT

TRAPS TO A TAG CALLED ".MRINT". CLOCKS THE MAINTENANCE REGISTER TWICE WITH AN 11 AND A 1 AND RETURNS TO THE NEXT INSTRUCTION FOLLOWING THE MRINT INSTRUCTION.

5.1.4 MRIND

TRAPS TO A TAG CALLED ".MRIND". CLOCKS AN INDEX PULSE INTO THE MAINTENANCE REGISTER THEN RETURNS TO THE NEXT INSTRUCTION FOLLOWING THE MRIND INSTRUCTION.

5.1.5 MRCLK

TRAPS TO A TAG CALLED ".MRCLK". CLOCKS THE MAINTENANCE REGISTER WITH AN 11 AND A 1, UPDATES THE CLOCK COUNTER, AND THEN RETURNS TO THE NEXT INSTRUCTION FOLLOWING THE MRCLK INSTRUCTION.

5.1.6 MRCK

TRAPS TO A TAG CALLED ".MRCK". THIS ROUTINE CHECKS THE MAINTENANCE REGISTER TO EQUAL THE VALUE FOLLOWING THE MRCK INSTRUCTION. IF THE MAINTENANCE REGISTER DOES NOT COMPARE, THE PROGRAM RETURNS TO THE "HLT" INSTRUCTION FOLLOWING THE CORRECT VALUE AND PRINTS OUT THE ERROR. IF THE MAINTENANCE REGISTER IS CORRECT, THE PROGRAM RETURNS TO THE INSTRUCTION FOLLOWING THE "HLT" INSTRUCTION.

5.1.7 DSCK

TRAPS TO A TAG CALLED ".DSCK". THIS ROUTINE CHECKS THE DRIVE STATUS REGISTER AND WORKS THE SAME WAY AS THE MRCK ROUTINE.

DESCRIPTION

5.1.8 XBIT

TRAPS TO A TAG CALLED ".XBIT". THIS ROUTINE GETS THE TWO DATA BITS THAT ARE CURRENTLY BEING WRITTEN FROM THE DATA BUFFER IN CORE AND STORES ONE BIT IN A LOCATION CALLED NOWOD AND THE OTHER BIT IN LOCATION NOWEV. THE PREVIOUS CONTENTS OF NOWOD AND NOWEV ARE STORED IN LASTOD AND LASTEV, RESPECTIVELY. THIS INFORMATION IS USED BY THE CLKD1 AND CLKD2 ROUTINES TO DETERMINE THE CORRECT STATES OF THE MWDB (BIT 12) AND MWDI (BIT 14) IN BITS IN RSMR WHEN WRITING. THIS ROUTINE MAKES BITS 16 AND 17 OF EACH DATA WORD (R504 WRITES 18 BIT WORDS) EQUAL ZERO. THE PROGRAM RETURNS TO THE NEXT INSTRUCTION FOLLOWING THE XBIT INSTRUCTION.

5.1.9 CLKD1 AND CLKD2

TRAPS TO LOCATIONS ".CLKD1" AND ".CLKD2". THESE TWO ROUTINES USE THE DATA BITS RECEIVED FROM THE XBIT ROUTINE TO DETERMINE THE CORRECT STATES OF MWDB (BIT 12) AND MWDI (BIT 14) IN RSMR WHEN WRITING. THESE ROUTINES ALSO CALCULATE THE CORRECT STATES OF THE CRCW, SB, AND LSR BITS IN RSMR AND DOES A COMPARE FOR THE CORRECT ANSWER. IF THE MAINTENANCE REGISTER DOES NOT COMPARE, THE PROGRAM RETURNS TO THE "HLT" INSTRUCTION FOLLOWING THE TRAP AND TYPES OUT THE ERROR. IF THE MAINTENANCE REGISTER WAS CORRECT, THE PROGRAM RETURNS TO THE NEXT INSTRUCTION FOLLOWING THE "HLT."

5.1.10 RBIT

TRAPS TO A TAG CALLED ".RBIT". THIS ROUTINE GETS THE TWO DATA BITS THAT ARE CURRENTLY BEING "READ" FROM THE DISK FROM THE INBUF DATA TABLE IN CORE AND STORES ONE BIT IN A LOCATION CALLED NOWOD AND THE OTHER BIT IN LOCATION NOWEV. THE PROGRAM THEN RETURNS TO THE NEXT INSTRUCTION FOLLOWING THE RBIT INSTRUCTION.

5.1.11 CLKR1 AND CLKR2

TRAPS TO LOCATIONS ".CLKR1" AND ".CLKR2". THESE TWO ROUTINES USING THE DATA BITS RECEIVED FROM THE RBIT ROUTINE SET AND CLEAR THE MRDB (BIT 2) AND MRDI (BIT 5) BITS IN RSMR IN THE PROPER SEQUENCE CORRESPONDING TO THE DATA PATTERN WHICH IS BEING "READ". THESE ROUTINES ALSO CALCULATE THE CORRECT STATES OF THE CRCW AND SB BITS IN RSMR AND DOES A COMPARE FOR THE CORRECT ANSWER. IF THE MAINTENANCE REGISTER DOES NOT COMPARE, THE PROGRAM RETURNS TO THE "HLT" INSTRUCTION FOLLOWING THE TRAP AND TYPES OUT THE ERROR. IF THE MAINTENANCE REGISTER WAS CORRECT, THE PROGRAM RETURNS TO THE NEXT INSTRUCTION FOLLOWING THE HLT.

DESCRIPTION

5.1.12 SCOPE

THIS SUBROUTINE CALL IS PLACED BETWEEN EACH SUBTEST IN THE INSTRUCTION SECTION. IT RECORDS THE STARTING ADDRESS OF EACH SUBTEST AS IT IS BEING ENTERED IN LOCATION "LAD". IF A SCOPE LOOP IS REQUESTED, THE CURRENT SUBTEST WILL BE LOOPED UPON. THE CONTENTS OF LAD MAY BE USED TO DETERMINE THE LAST SUBTEST SUCCESSFULLY COMPLETED.

5.1.13 HLT

THIS ROUTINE PRINTS OUT AN ERROR MESSAGE (SEE 6.1). TO INHIBIT TYPEOUTS, PUT SW<13> ON A 1.

5.1.14 TRAPCATCHER

A ".+2" - "HALT" SEQUENCE IS REPEATED FROM 0 - 776 TO CATCH ANY UNEXPECTED TRAPS. THUS ANY UNEXPECTED TRAPS OR INTERRUPTS WILL HALT AT THE VECTOR + 2.

6. ERRORS

6.1 ERROR PRINTOUT

THE FORMAT IS AS FOLLOWS:

```
ADR  CS1 = ----- CS2 = ----- ER = -----
GOOD      = ----- BAD = -----
```

WHERE:

```
CS1, CS2, ER ETC.      = RH11/RS04 REGISTERS.
GOOD                   = EXPECTED DATA.
BAD                     = DATA RECEIVED.
```

TO FIND THE FAILING TEST, LOOK AT THE LISTING ABOVE THE ADDRESS TYPED.

6.2 ERROR RECOVERY

RESTART AT 200 OR AT 220

DESCRIPTION

7. RESTRICTIONS

NONE

8. MISCELLANEOUS

8.1 EXECUTION TIME

A BELL WILL RING WITHIN ONE AND A HALF MINUTES WITH ALL SWITCHES DOWN.

8.2 STACK POINTER

STACK IS INITALLY SET TO 500

9. TEST DESCRIPTION

1. TEST FOR ONLINE DRIVES

SET ERROR BITS IN RSER. THIS CAUSES ATTENTION SUMMARY BITS TO SET IN RSAS. DO FOR ALL DRIVES. RSAS HAS NOT YET BEEN TESTED. SO IN THE CASE OF NO BITS IN RSAS SETTING, DRIVE 0 IS TESTED.

2. RESET TEST FOR REGISTERS

SET ALL R/W BITS IN RSCS1, RSCS2, RSBA, RSDA, RSER, RSWC, RSDB, AND RSMR. DO A RESET AND TEST ALL R/W BITS TO BE CLEARED.

3. SET AND CLEAR ALL REGISTERS

SET ALL R/W BITS IN RSCS1, RSCS2, RSBA, RSDA, RSER, RSWC, RSDB AND RSMR AND TEST. SET ALTERNATE BITS AND CHECK TO MAKE SURE BITS ARE NOT TIED TOGETHER. NOW SET ALL BITS AND CLEAR THEM TO MAKE SURE ALL CAN BE CLEARED ONCE SET.

4. TEST "CLEAR BIT" IN RSCS2

SET ALL R/W BITS IN RSCS1, RSCS2, RSBA, RSDA, RSER, RSWC, RSDB, AND RSMR. SET CLEAR BIT IN RSCS2. NOW TEST ALL R/W BITS FOR 0 IN ALL THE ABOVE REGISTERS.

5. LOAD RSDB WITH ALL ONES AND ALL ZEROS

DESCRIPTION

LOAD RSDB WITH A WORD OF ZEROS AND A WORD OF ONES. WAIT FOR "OR" TO SET AND THEN CHECK OUTPUT OF SILO. IF OR DID NOT SET ERROR MESSAGE APPEARS.

6. TEST PROGRAM INTERRUPT

THE PROGRAM FORCES A INTERRUPT BY MOVING A 300 INTO RSC51.

7. MAINTENANCE TIMING TEST

THE FOLLOWING TEST ON THE R504 DISK IS A SINGLE-STEPPED MAINTENANCE MODE TEST ON THE R504 TIMING LOGIC. THE ACTUAL DISK SURFACE IS SUBSTITUTED BY THE MAINTENANCE REGISTER, I.E., THE PROGRAM WILL SUPPLY ALL "DISK CLOCK" PULSES TO DRIVE THE TIMING LOGIC. WE ARE TESTING THE ENTIRE "TIMING TRACK", INDEX PULSE FUNCTION, RESYNC AREA, SECTOR COUNTER, ETC.

- PUT DRIVE INTO MAINTENANCE MODE.
- ASSERT INDEX PULSE TO INITIALIZE DRIVE TIMING LOGIC.
- INDEX PULSE SHOULD CLEAR LOOK-AHEAD REGISTER.
- CLOCK TIMING TO STEP THROUGH RESYNC PERIOD.
- CHECK FOR SECTOR PULSE.
- PERFORM MAINTENANCE CLOCK OPERATION TO CHECK FOR 64 SECTOR PULSES.
- THE LOOK-AHEAD REGISTER SHOULD NOW POINT TO THE CURRENT SECTOR.
- REPEAT STEPS TO CLOCK THROUGH ALL THE SECTORS TO CHECK SECTOR COUNT.

8. SECTOR FRACTION TEST

CLOCK THROUGH AN ENTIRE TRACK IN MAINTENANCE MODE WHILE CHECKING FOR THE PROPER OPERATION OF THE LOOK-AHEAD REGISTER AND THE SECTOR FRACTION COUNTER.

- INITIALIZE DRIVE AND STEP THROUGH RESYNC AREA.
- CHECK FOR SECTOR PULSE.
- LOOK-AHEAD REGISTER SHOULD = 0.
- STEP THROUGH THE PREAMBLE AREA AND SECTOR DATA AREA WHILE CHECKING THE SECTOR FRACTION.
- CHECK FRACTIONS TO CHANGE AFTER THE CORRECT NUMBER OF MAINTENANCE CLOCKS.

WHEN THE LAST WORD IS BEING TRANSFERRED, SECTOR AND FRACTION IS EQUAL TO 7777 TO INDICATE LAST WORD ON THIS TRACK -- HANDLE END OF TRACK SPECIAL FOR THE LOOK-AHEAD REGISTER WILL CLEAR THE FRACTION BITS IF ANOTHER WORD IS CLOKED. RSLA SHOULD INDICATE 7700 ON ANOTHER MAINTENANCE CLOCK.

9. DISK ILLEGAL FUNCTION TEST

DESCRIPTION

TEST ILLEGAL FUNCTION (ILF) IN RSER. SEND AN ILLEGAL FUNCTION CODE TO THE DRIVE CONTROL REGISTER WITHOUT SETTING THE GO BIT. THE "ILF" BIT SHOULD NOT BE SET. THE "GO" BIT IS THEN SET. A CHECK IS THEN MADE FOR "ATA" AND "ERR" TO BE SET IN THE DRIVE STATUS REGISTER (RSDS) AND "ILF" IN THE DRIVE ERROR REGISTER (RSER). ALL ILLEGAL FUNCTION CODES ARE CHECKED.

10. TEST THE DRIVE NO-OP CODES 1 AND 21

THIS IS TESTED WITH AND WITHOUT ERRORS BEING SET TO PROVE THAT IT DOESN'T CHANGE ANYTHING.

11. DRIVE SEARCH TEST 1

A DRIVE SEARCH FUNCTION IS GIVEN TO THE DRIVE FOR SECTOR 3. (SECTOR 41 IF SECTOR INTERLEAVING IS ENABLED) THE POSITIONING IN PROGRESS BIT (PIP) AND THE DRIVE READY BIT (DRY) IN THE DRIVE STATUS REGISTER (RSDS) ARE CHECKED. THE ADDRESS CONFIRM BIT (AC) IS ALSO CHECKED.

12. DRIVE SEARCH TEST 2

THIS TEST INITIALIZES A DRIVE SEARCH FUNCTION FOR SECTOR 0 WHEN THE DRIVE IS CURRENTLY AT THE DESIRED SECTOR. THE SEARCH FUNCTION SHOULD NOT BE COMPLETED UNTIL THE DRIVE MAKES A COMPLETE REVOLUTION AND REACHES THE BEGINNING OF THE DESIRED SECTOR.

13. REGISTER MODIFICATION REFUSED TEST

RMR IN THE DRIVE ERROR REGISTER (RSER) SHOULD SET BY TRYING TO MODIFY ONE OF THREE DRIVE REGISTERS WHILE THE DRIVE IS BUSY DURING A DRIVE SEARCH FUNCTION.

1. RSCS1
2. RSDA
3. RSER

TEST THAT RMR DOES NOT SET WHEN MODIFYING THE ATTENTION SUMMARY REGISTER (RSAS).

14. DRIVE SELECT TEST

THE PROGRAM LOADS A DRIVE REGISTER, OF THE DRIVE UNDER TEST, TO ALL ONES. THE PROGRAM THEN FINDS A NON-EXISTENT DRIVE AND TRIES TO LOAD ITS REGISTER WITH ALTERNATE ONES AND ZEROS. THIS SHOULD CAUSE "NED" TO SET IN RSCS2. THE PROGRAM RE-SELECTS THE DRIVE UNDER TEST AND CHECKS ITS REGISTER TO SEE IF IT WAS MODIFIED. IT SHOULD CONTAIN ALL ONES.

15. MAINTENANCE WRITE TEST

DESCRIPTION

THIS IS AN R504 DISK MAINTENANCE MODE (SINGLE-STEPPED) SECTOR WRITE TEST. WE ARE TESTING THE COMPLETE DATA PATH FOR A DATA TRANSFER TO THE DISK. MILLER ENCODED DATA TO BOTH SURFACES IS CHECKED ALONG WITH CORRECT GENERATION OF THE CRC WORD AT THE END OF THE SECTOR. INDEX PULSES, RESYNC, TIMING PREAMBLE, AND SECTOR PULSES ARE ALSO CHECKED.

16. MAINTENANCE READ TEST

THIS IS AN R504 DISK MAINTENANCE MODE (SINGLE-STEPPED) SECTOR READ TEST. WE ARE TESTING THE COMPLETE DATA PATH FROM THE DISK DECODING LOGIC TO CORE MEMORY. (THE PHASE LOCK LOOP IS NOT TESTED IN MAINTENANCE MODE.)

17. MAINTENANCE MODE DATA WRITE CHECK TEST

A ONE SECTOR TRANSFER IS DONE WITH A WRITE CHECK FUNCTION. WITHIN THE R504, A WRITE CHECK FUNCTION IS IDENTICAL TO A READ FUNCTION.

18. MAINTENANCE MODE CRC TEST 1 (NO DCK ERRORS)

THE R504 DISK IS SET UP TO READ (IN MAINTENANCE MODE) ONE SECTOR OF A SPECIALLY CREATED DATA PATTERN WHICH LEAVES ONLY ONE BIT SET IN THE CRC REGISTER PRIOR TO CHECKING THE CRC WORD. THE CORRESPONDING CRC WORD IS THEN "READ", RESULTING IN NO DCK ERROR. THE DATA PATTERN IS THEN MODIFIED (BY SHIFTING) AND THE ENTIRE READ SEQUENCE REPEATED UNTIL ALL 16 BITS IN THE CRC REGISTER HAVE BEEN CHECKED.

19. MAINTENANCE MODE CRC TEST 2 (CAUSE DCK ERRORS)

THIS TEST IS SIMILAR TO CRC TEST 1 EXCEPT THAT THE DATA PATTERN HAS BEEN MODIFIED TO LEAVE A SINGLE BIT SET IN THE CRC REGISTER AFTER BOTH DATA AND CRC WORDS HAVE BEEN "READ". THIS CAUSES A DCK ERROR. THE READ SEQUENCE IS REPEATED 16 TIMES TO TEST THAT EACH BIT IN THE CRC REGISTER CAN CAUSE A DCK ERROR.

20. IGNORE FUNCTION TEST

PUT THE DISK IN MAINTENANCE MODE AND SET ERROR CONDITIONS IN THE DRIVE ERROR REGISTER (R5ER). TRY TO START A READ TRANSFER. THE "GO" BIT IN R5CS1 SHOULD NOT SET. MISSED TRANSFER ERROR (MXF) SHOULD SET IN R5CS2 WHICH IN TURN SHOULD CAUSE "TRE" AND "SC" TO SET IN R5CS1.

21. INVALID ADDRESS TEST

FLOAT A 1 THROUGH THE FOUR SPARE ADDRESS BITS IN THE DISK ADDRESS REGISTER (R5DA). THIS SHOULD CAUSE "IAE" TO SET IN THE ERROR REGISTER (R5ER) WHEN A READ FUNCTION IS LOADED INTO

NO1

DESCRIPTION

RSCS1 WHICH IN TURN SHOULD CAUSE ATTENTION TO SET IN THE DRIVE STATUS REGISTER (RSDS) AND "TRE" AND "SC" TO SET IN THE CONTROL REGISTER (RSCS1).

22. DISK OPERATION INCOMPLETE (OPI) ERROR TEST

PUT DISK IN MAINTENANCE MODE AND START A READ COMMAND. THEN ISSUE THREE DISK "INDEX" PULSES TO SIMULATE A COMPLETE ROTATION OF THIS DISK SURFACE. THE THIRD INDEX PULSE SHOULD CAUSE OPERATION INCOMPLETE (OPI) TO SET IN THE DRIVE ERROR REGISTER (RSE) AND "ATA" AND "ERR" IN THE DRIVE STATUS REGISTER (RSDS).

23. PARITY ERROR TEST

SET "PAT" BIT IN RSCS2. WRITE A DRIVE REGISTER. "PAR" SHOULD SET IN THE DRIVE ERROR REGISTER (RSE) WHICH SHOULD CAUSE "ATA" TO SET IN RSDS AND "SC" TO SET IN RSCS1.

24. MAINTENANCE MODE INTERRUPT TEST

IN THIS TEST THE INTERRUPT ENABLE (I.E.) BIT IS SET. A TWO SECTOR WRITE COMMAND IS GIVEN. AN "RMA" ERROR IS THEN CAUSED WHILE THE FIRST SECTOR IS BEING WRITTEN. WHEN THE FUNCTION IS COMPLETED, THE DRIVE SHOULD INTERRUPT.

25. DISK ADDRESS OVERFLOW (AOE) TEST

SET UP TO TRANSFER 2 SECTORS TO THE DISK, STARTING AT TRACK 77 SECTOR 77 TO CAUSE A DISK ADDRESS OVERFLOW CONDITION. ALSO CHECK LAST BLOCK TRANSFER (LBT) BIT TO SET IN THE RSDS REGISTER.

26. MAINTENANCE VERIFY TEST

THIS TEST WILL ONLY RUN IF SWITCH 11 IS SET IN THE "SWITCH REGISTER" FOR IT WILL ACTUALLY WRITE DATA ONTO THE DISK. IT WILL WRITE ONE TRACK OF ALL ONES. THE DRIVE IS THEN PLACED IN MAINTENANCE MODE AND IT WILL THEN WRITE ONE SECTOR OF THE SAME TRACK WITH ALL ZEROS. THE DRIVE IS THEN TAKEN OUT OF "MAINTENANCE MODE" AND THE TRACK IS THEN READ. THE TRACK SHOULD CONTAIN ALL ONES.

! TITLE ZZ-CERSD-D RH70-R504 MAINTENANCE MODE DIAGNOSTIC
; COPYRIGHT 1974, 1975, 1976, 1978 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
; PROGRAM BY STANLEY HARACKIEWICZ

;	SWITCH	USE
;	-----	-----

	100000			SW15=	100000		:HALT ON ERROR
	040000			SW14=	40000		:LOOP ON TEST
	020000			SW13=	20000		:INHIBIT ERROR TYPEOUTS
	010000			SW12=	10000		:TYPEOUT ALL ERRORS IN DATA COMPARE ROUTINE
	004000			SW11=	4000		:RUN MAINTENANCE MODE VERIFY TEST
	002000			SW10=	2000		:0 - BELL ON PASS COMPLETE
							:1 - BELL ON ERROR
	001000			SW9=	1000		:LOOP ON ERROR
	000400			SW8=	400		:LOOP ON TEST IN SW(7:0)
	000000				0		:TRAP CATCHER FROM 0 - 776
	000046				46		:HOOKS FOR ACT 11
000046	021274			SENDAD			
	000052				52		
000052	040000			BIT14			
	000200				200		
000200	023737	000042	000046	CMP		2#42,2#46	:ACT11?
000206	001402			BEQ		NOMI	:BR IF YES
000210	000137	000232		JMP		2#BEGIN1	
000214	000137	000220		NOMI: JMP		2#NOMI1	
	000220				220		
000220	052767	000100	000716	NOMI1: BIS		#BIT6,FLAG2	:TEST ALL DRIVES
000226	000137	001234		BEGIN2: JMP		2#BEGIN	
	000232						
000232	042767	000100	000704	BEGIN1: BIC		#BIT6,FLAG2	:CLEAR MULTI DRIVE FLAG
000240	000772			BR		BEGIN2	

C02

000001
104000
177776
177776
177570
177570
000007
000000
000001
000002
000003
000004
000005
000006
000007
000001
000002
000004
000010
000020
000040
000100
000200
000400
001000
002000
004000
010000
020000
040000
100000
000001
000000

N= 1
HLT= EMT
PS= 177776
PSW= PS
SWR= 177570
DISPLAY=SWR
BELL= 7
R0= %0
R1= %1
R2= %2
R3= %3
R4= %4
R5= %5
SP= %6
PC= %7
BIT0= 1
BIT1= 2
BIT2= 4
BIT3= 10
BIT4= 20
BIT5= 40
BIT6= 100
BIT7= 200
BIT8= 400
BIT9= 1000
BIT10= 2000
BIT11= 4000
BIT12= 10000
BIT13= 20000
BIT14= 40000
BIT15= 100000

GOOD= %1
BAD= %0

:INITALIZE FOR NEWTST
:SET HLT TO EMT FOR ERROR TYPEOUTS
:PROCESSOR STATUS
:PROCESSOR STATUS WORD
:SWITCH REGISTER
:DISPLAY REGISTER
:BELL
:R0 - DEFINE REGISTERS
:R1
:R2
:R3
:R4
:R5
:R6 - STACK POINTER
:R7 - PROGRAM COUNTER
:BIT EQUATES

:FOR GOOD DATA
:FOR BAD DATA

002

001000 000000
001002 000000
001004 000000 000000
001010 000000
001012 000000
001014 001000
001016 177564
001020 177566

001100

. = 1000
ICNT: 0
ERRORS: 0
PCNT: 0,0
LAD: 0
HLTADR: 0
FILCHR: 1000
TPS: 177564
TPB: 177566

;LH = ITERATION COUNT ;RH = TEST NO.
;ERROR COUNT
;2 WORD PASS COUNT
;LOOP ADDRESS FOR SCOPE
;ADDRESS OF LAST HLT INSTRUCTION EXECUTED
;FILCHR=0 (CHAR) ;FILCHR+1=2 (COUNT)
;OUTPUT STATUS REGISTER
;OUTPUT BUFFER

. = 1100
;DISK I/O REGISTERS

001100 172040
001102 172050
001104 172042
001106 172044
001110 172046
001112 172052
001114 172054
001116 172056
001120 172060
001122 172062
001124 172064
001126 172066
001130 000204
001132 000206
001134 172041
001136 172051
001140 172043
001142 172045

RSCS1: 172040
RSCS2: 172050
RSCW: 172042
RSBA: 172044
RSDA: 172046
RSDS: 172052
RSEB: 172054
RSAS: 172056
RSLA: 172060
RSDB: 172062
RSMR: 172064
RSDT: 172066
RSVEC: 204
RSVCPS: 206
RSCS1B: 172041
RSCS2B: 172051
RSCWB: 172043
RSBAB: 172045

;DISK CONTROL + STATUS REGISTER
;DISK CONTROL + STATUS REGISTER
;WORD COUNT REGISTER
;BUS ADDRESS
;DISK ADDRESS (DESIRED ADDRESS)
;DRIVE STATUS
;ERROR REG.
;ATTENTION SUMMARY
;LOOK AHEAD
;DATA BUFFER REGISTER
;MAINTENANCE REGISTER
;DRIVE TYPE REGISTER
;INTERUPT VECTOR
;INTERUPT PRIO. VECTOR
;ODD BYTE ADD FOR CS1
;ODD BYTE ADD FOR CS2
;ODD BYTE ADD FOR CW
;ODD BYTE ADD FOR BA

E02

:BIT ASSIGNMENTS FOR ERROR TYPEOUTS
:THE RS REGISTERS ARE DIVIDED INTO 3 GROUPS.
:CS1,CS2 AND ER ARE IN THE FIRST GROUP. THIS GROUP IS ALWAYS
:TYPED WITH EITHER OF THE OTHER GROUPS. AS,BA,DA, WC AND DS
:ARE IN THE SECOND GROUP. DT,DB,MR, AND LA ARE IN THE 3RD
:GROUP. YOU CAN NOT INTERMIX GROUP 2 OR 3. THEY HAVE
:TO BE TYPED SEPERATELY.
:EXAMPLE: HLT !CS1 AS BA
: HLT !CS1!DT!DB

000001	CS1=1	:CONTROL AND STATUS 1
000002	ER=2	:CONTROL AND STATUS 2
000004	DA=4	:DESIRED ADD
000010	WC=10	:WORD COUNT
000020	BA=20	:BUS ADDRESS
000040	DS=40	:DRIVE STATUS
000100	AS=100	:ATTENTION SUMMARY
000200	CS2=200	:CONTROL AND STATUS REG
000204	LA=204	:LOOK AHEAD
000210	DB=210	:DATA BUFFER
000220	MR=220	:MAINTENANCE
000240	DT=240	:DRIVE TYPE

:BIT ASSIGNMENTS FOR THE REGISTER BITS

040000	TRE=40000	:TRANSFER ERROR CS1
100000	SC=100000	:SPECIAL CONDITIONS CS1
000100	IR=100	:INPUT READY CS2
000200	OR=200	:OUTPUT READY CS2
002000	PGE=2000	:PROGRAM ERROR-CS2
010000	NED=10000	:NON-EXISTENT DRIVE CS2
040000	WCE=40000	:WRITE CHECK ERROR-CS2
100000	DLT=100000	:DATA LATE ERROR CS2
000200	DRY=200	:DRIVE READY DS
020000	PIP=20000	:POSITIONING IN PROGRESS DS
002000	LBT=2000	:LAST BLOCK TRANSFER-DS
040000	ERR=40000	:ERROR DS
100000	ATA=100000	:ATTENTION ACTIVE-DS
001000	DAO=1000	:DISK OVERFLOW ERROR-ER
100000	DCK=100000	:DATA CHECK ERROR-ER
000010	BAI=10	:BUS ADDR INCREMENT INHIBIT
000100	IE=100	:INTERRUPT INABLE CS1

;WORKING LOCATIONS

```

001144 000000
001146 000000
001150 000000
001152 000000
001154 000000
001156 000000
001160 000000
001162 000000
001164 000000
001166 000000
001170 000000
          172100
001172 000000
001174 000000 000000
001200 000000
001202 000000
001204 000000
001206 000000
001210 000000
001212 000000
001214 000000
001216 000000
001220 000000
001222 000000
001224 000000
001226 000000
001230 000000
001232 000000

```

```

FLAG2: 0
LSTEV: 0
LSTOD: 0
NOWEV: 0
NOWOD: 0
RSD: 0
UNNUM: 0
UNITSV: 0
UNCMP: 0
ONCEE: 0
TIMSV: 0
MPRO=172100
SAVEE: 0
MCCNT: 0,0
WCRC: 0
REPT: 0
REPT1: 0
CLKCNT: 0
INBIT: 0
WK15: 0
WORK: 0
WORK0: 0
WORK1: 0
WORK2: 0
WORK3: 0
WORK4: 0
WORK5: 0
WORK6: 0

```

```

;SECOND FLAG WORD
;LAST EVEN BIT TRANSFERED
;LAST ODD BIT TRANSFERED
;PRESENT EVEN BIT BEING XFERED
;PRESENT ODD BIT BEING XFERED
;SAME
;UNIT CURRENTLY BEING TESTED
;SET BIT=UNIT ON BUS
;FOR COMPARING FOR # OF DEVICE
;DID WE TEST ANY DRIVES
;SAVE LOC FOR TIME
;PARITY REG
;WORK LOC
;MAINT CLOCK COUNT
;WORK LOC FOR CREATING CRC WORD
;REPEAT COUNTER
;REPEAT COUNTER
;CLOCK COUNTER FOR EACH WORD
;USED IN CRC CAL ROUTINE
;USED IN CRC CAL ROUTINE

```

G02

;DISCRIPTION OF BITS IN LOCATION ONCEE

;BIT0 MEANS FOUND DRIVE
;BIT1 ERROR DO NOT CHANGE ILLEGAL FUNCTION
;BIT2 ERROR FLAG
;BIT3 TESTING CODE 21 FLAG
;BIT4 TEST ONLY ONE DRIVE
;BIT5 TIMEOUT CLOCK COUNT
;BIT6 1ST TRANSFER WORD FLAG
;BIT7 WRITTING LAST WORD OF SECOTR
;BIT8 TRANSFERRING CRC WORD
;BIT9 FOR INTERLEAVED DRIVES
;BIT10 1ST TIME FLAG IN SECTOR FRACTION TEST
;BIT11 DO TKSEL TEST
;BIT12 TYPE COULD NOT FIND NED ONLY ONCE
;BIT13 TYPE NO MEM ON B PORT ONLY ONCE
;BIT14 0- DO WCE WITH 0 -1 DO WCE WITH 1
;BIT15 MEANS ERROR FOUND

;DISCRIPTION OF BITS IN LOCATION FLAG2

;BIT0 SWITCH FOR RWCLK IN MR REG
;BIT1 MAINTENANCE MODE VERIFY TEST
;BIT2 IN WRITE CK TEST FOR CLKR1 ROUTINE
;BIT3 DONE 1ST CRC WD IN CRC TEST
;BIT4 1ST TIME THROUGH IN CRC TEST
;BIT5 IN CRC TEST
;BIT6 FLAG TO TEST ALL DRIVES

001234	012706	000500		BEGIN:	MOV	#500,SP	:SET STACK TO *** 500 ***
001240	012737	025110	000024		MOV	#.POWER,2#24	:SET UP PF VECTOR
001246	012737	000340	000026		MOV	#340,2#26	:LOCK OUT THE WORLD
001254	012737	024526	000030		MOV	#.HLT,2#30	:SET EMT VECTOR
001262	012737	000340	000032		MOV	#340,2#32	:LOCK UP
001270	012737	025512	000034		MOV	#.TRAP,2#34	:SET TRAP VECTOR
001276	012737	000340	000036		MOV	#340,2#36	:LOCK UP
001304	005067	177470			CLR	ICNT	:INIT ICNT
001310	005067	177474			CLR	LAD	:INIT LAD
001314	042767	177677	177622		BIC	#177677,FLAG2	
001322	042767	153777	177636		BIC	#153777,ONCE	
001330	005227	177777			INC	#-1	:FIRST START?
001334	001027				BNE	6\$	:BR IF NO
001336	023737	000042	000046		CMP	2#42,2#46	:ACT11?
001344	001423				BEQ	6\$	:BR IF YES
001346	104402	001352			TYPE	..+2	:ASCIZ <15><12>"ZZ-CERSD D RH70-R504 MAINT MODE"
001414	032767	000100	177522	6\$:	BIT	#BIT6,FLAG2	:TEST ALL DRIVES?
001422	001402				BEQ	5\$	:ASK
001424	000137	001766			JMP	2#MULT11	
001430				5\$:			
001430	104402	001434			TYPE	..+2	:ASCIZ <15><12>"TEST ALL DRIVES? (Y OR N) "
001472	104412				RDLIN		
001474	122767	000131	023770		CMPB	#Y,INPUT	:TEST FOR YES
001502	001531				BEQ	MULT11	:YES
001504	052767	000020	177454		BIS	#BIT4,ONCE	:SET TEST ONLY ONE DRIVE FLAG
001512				1\$:			
001512	104402	001516			TYPE	..+2	:ASCIZ "TYPE UNIT #"
001532	104410				RDOCT		
001534	012604				MOV	(6)+,R4	:GET NUMBER
001536	022704	000010			CMP	#10,R4	:CORRECT #
001542	101763				BLOS	1\$	:NO
001544	010467	177410			MOV	R4,UNNUM	:SET UNIT #
001550	005002				CLR	R2	:CLEAR WORK AREA
001552	000261				SEC		:SET CARRY
001554	006102			2\$:	ROL	R2	:SET WORK BIT
001556	005704				TST	R4	:IS THIS BIT CORRESPOND WITH CORRECT DRIVE #
001560	001402				BEQ	3\$	:YES
001562	005304				DEC	R4	:NO TRY AGAIN
001564	000773				BR	2\$	:TEST AGAIN
001566	010267	177370		3\$:	MOV	R2,UNITSV	:SET DRIVE BIT IN UNITSV
001572	010267	177366			MOV	R2,UNCOMP	:SET UNIT COMPARE
001576	016777	177356	177276		MOV	UNNUM,2#RSCS2	:LOAD DRIVE
001604	012777	177777	177302		MOV	#-1,2#RSE	:LOAD ERRORS
001612	023737	000042	000046		CMP	2#42,2#46	:ACT11?
001620	001447				BEQ	7\$	:BR IF YES
001622	104402	001626			TYPE	..+2	:ASCIZ "ALL ERROR LIGHTS ON SELECTED UNIT SHOULD BE ON
001736	000000				HALT		:WAIT FOR LIGHTS TO BE CHECKED
001740	026777	177216	177150	7\$:	CMP	UNITSV,2#RSAS	:DID CORRECT ATA SET
001746	001405				BEQ	4\$	
001750	017700	177142			MOV	2#RSAS,BAD	:GET RSAS
001754	016701	177202			MOV	UNITSV,GOOD	:GET CORRECT AND
001760	104000				HLT		:RSAS=BAD GOOD=CORRECTIONS
							:ATA BIT SHOULD SET FOR ERRORS
							:WERE SET IN RSE
001762	000167	000430		4\$:	JMP	NOWGO	:START TESTING

;NOW TEST FOR DRIVES

001766	012701	000010		MULTII:	MOV	#8, R1	:PUT 8 INTO R1 FOR COUNT
001772	005077	177104			CLR	DRSCS2	:SET DEVICE TO ZERO
001776	012777	177777	177110	TRY:	MOV	#-1, DRSER	:CAUSE AN ERROR +SETS BIT IN RSAS REG
002004	005301				DEC	R1	:DO A MAXIMUM OF 8 TIMES
002006	001403				BEQ	DVNUM	:TESTED FOR ALL DRIVES GET OUT
002010	005277	177066			INC	DRSCS2	:INCREMENT DRIVE UNIT
002014	000770				BR	TRY	:REPEAT FOR NEXT DRIVE
002016	017767	177074	177136	DVNUM:	MOV	DRSAS, UNITSV	:SAVE
002024	012767	000401	177132		MOV	#401, UNCMP	:SETUP TO CMP WITH UNITSV
002032	012767	000000	177120		MOV	#0, UNNUM	:PUT 0 INTO UNIT NO.
002040	032767	020000	175522		BIT	#BIT13, SWR	:INHIBIT TYPE OUT?
002046	001015				BNE	STTEST	:YES
002050	104402	002054			TYPE	,.+2	:ASCIZ <15><12>"TESTING UNIT "
002074	042767	100000	177064		BIC	#BIT15, ONCEE	:CLEAR ERROR FLAG
002102	036767	177056	177052	STTEST:	BIT	UNCMP, UNITSV	:IS THIS DRIVE ON THE SYSTEM
002110	001440				BEQ	TRYNX	:NO
002112	016777	177042	176762		MOV	UNNUM, DRSCS2	:YES PUT UNIT # INTO CS2
002120	022777	000002	177000	3\$:	CMP	#2, DRSDT	:IS THIS A RS04?
002126	001404				BEQ	1\$	:YES
002130	022777	000003	176770		CMP	#3, DRSDT	:IS IT A RS04?
002136	001025				BNE	TRYNX	:GET A NEW NUMBER
002140	032767	020000	175422	1\$:	BIT	#BIT13, SWR	:INHIBIT TYPE OUT?
002146	001020				BNE	4\$	:YES
002150	032767	100000	177010		BIT	#BIT15, ONCEE	:ANY ERRORS?
002156	001404				BEQ	5\$	:NO
002160	104402	002164		5\$:	TYPE	,.+2	:ASCIZ <15><12><12>
002170							
002170	016746	176764			MOV	UNNUM, -(6)	:PUT UNNUM ON STACK
002174	104406				TYPES		:TYPE STACK IN OCTAL - SUPRESS
002176	104402	000040			TYPE	,40	:TYPE SPACE
002202	042767	100000	176756		BIC	#BIT15, ONCEE	:CLEAR ERROR FLAG
002210	000502			4\$:	BR	NOWGO	:NOW TEST
002212	032767	000020	176746	TRYNX:	BIT	#BIT4, ONCEE	:MULTI DRIVE
002220	001074				BNE	DONEE	:NO
002222	006367	176736		1\$:	ASL	UNCMP	:CHECK NEXT BIT FOR DRIVE
002226	103403				BCS	CHCKDV	:DID WE TEST ANY REG?
002230	005267	176724			INC	UNNUM	:INC UNIT #
002234	000722				BR	STTEST	:CHECK FOR NEXT DRIVE


```

002236 032767 000001 176722 CHCKDV: BIT      #BIT0,ONCEE      :DID WE TEST ANY DRIVES?
002244 001062          BNE      DONEE      :YES WE DID TEST A DRIVE
002246 012767 100000 176710      MOV      #100000,UNCMP      :NO DRIVES TESTED, COULD NOT SET
002254 005067 176700          CLR      UNNUM      :ANY AS BITS, THUS DEFAULTS TO
002260 032767 020000 175302      BIT      #BIT13,SWR      :INHIBIT TYPE OUT?
002266 001050          BNE      4$      :YES
002270 016746 176664          MOV      UNNUM,-(6)      :PUT UNNUM ON STACK
002274 104406          TYPES      :TYPE STACK IN OCTAL - SUPRESS
002276 104402 000040          TYPE      ,40      :TYPE SPACE
002302 104402 002306          TYPE      ,+2      :.ASCIZ <15><12>"COULD NOT FIND DRIVE WILL TEST DRIVE 0
002400 012767 000001 176556      MOV      #1,UNCMP
002406 000000          HALT
002410 000402          4$: BR      NOWGO      :WAIT
002412 000167 016612      DONEE: JMP      DONE      :TEST DRIVE 0
                                :GET OUT

```

```

;THIS TEST IS DESIGNED TO TEST THE ABILITY OF RESET
;TO CLEAR ALL THE RH AND RS REGISTERS

```

```

002416 052767 000001 176542 NOWGO: BIS      #BIT0,ONCEE      :SET FOUND DRIVE FLAG
002424 016767 022074 176536      MOV      TIMES,TIMSV      :SAVE TIME
002432 012767 000001 022064      MOV      #1,TIMES      :ONLY TEST ONCE

```

```

;*****
;TEST 1      RESET TEST FOR REGISTERS
;*****

```

```

002440 104400          ST1: SCOPE
002442 012737 000340 177776      MOV      #340,2#PS      :LOCK OUT INTERRUPTS
002450 016777 176504 176424      MOV      UNNUM,2#RSCS2      :LOAD UNIT #
002456 012777 177776 176414      MOV      #177776,2#RSCS1      :SET ALL
002464 012777 177777 176414      MOV      #177777,2#RSBA      :POSSIBLE R/W
002472 012777 177777 176410      MOV      #177777,2#RSDA      :BITS IN THESE REGISTERS
002500 012777 177777 176406      MOV      #177777,2#RSER
002506 012777 177777 176410      MOV      #177777,2#RSMR
002514 012777 177777 176362      MOV      #177777,2#RSWC
002522 012777 177737 176352      MOV      #177737,2#RSCS2
002530 000005          RESET      :CLEAR ALL BITS IN ALL REG.

```

```

;TEST RSCS2 FOR CLEARED BITS

```

```

002532 022777 000100 176342      CMP      #100,2#RSCS2      :DID THESE BITS GET CLEARED?
002540 001401          BEQ      +4      :YES
002542 104200          HLT      :CS2      : (417) SHOULD BE CLEARED IN CS2
002544 016777 176410 176330      MOV      UNNUM,2#RSCS2      :PUT # OF UNIT IN TEST IN CS2
002552 022777 010600 176332      CMP      #10600,2#RSDS      :IS DPR AND MOL SET?
002560 001401          BEQ      +4      :YES
002562 104040          HLT      :DS      :NO WHY NOT?

```

```

;TEST CONTROL AND STATUS REG 1

```

```

002564 022777 004200 176306      CMP      #4200,2#RSCS1      :DID THE READY BIT SET?
002572 001401          BEQ      +4      :YES
002574 104001          HLT      :CS1      :READY SHOULD BE SET

```


;TEST BUS ADDRESS REGISTER

002576	005777	176304	TST	QRSBA	: IS BA REG. CLEARED
002602	001401		BEQ	+4	: YES
002604	104020		HLT	!BA	: SHOULD BE 0

;TEST DISK ADDRESS REGISTER

002606	005777	176276	TST	QBSDA	: IS DA CLEARED
002612	001401		BEQ	+4	: YES
002614	104004		HLT	!DA	: SHOULD BE 0

;TEST ERROR REG RSER

002616	005777	176272	TST	QRSER	: DID RSER CLEAR?
002622	001401		BEQ	+4	: YES
002624	104002		HLT	!ER	: BITS(157015) SHOULD BE CLEARED

;TEST RS MAINTENANCE REGISTER

002626	032777	000077	176270	BIT	#77, QRSMR	: DID THESE BITS GET CLEARED
002634	001401			BEQ	+4	: YES
002636	104220			HLT	!MR	: BITS(77) SHOULD BE 0

;TEST WC REG IT SHOULD NOT CHANGE

002640	022777	177777	176236	CMP	#177777, QRSWC	: DID IT CHANGE?
002646	001401			BEQ	+4	: NO
002650	104010			HLT	!WC	: RESET SHOULD NOT MODIFY RSWC

;TEST RSAS

002652	005777	176240	TST	CRSAS	: IS REG CLEAR
002656	001401		BEQ	+4	: YES
002660	104100		HLT	!AS	: NO

```

*****
;TEST 2 TEST CLEAR BIT IN CS2 ON ALL THE R/W BITS
*****
TST2: SCOPE

```

002662 104400

```

002664 012737 000340 177776
002672 016777 176262 176202
002700 012777 043576 176172
002706 012777 177777 176172
002714 012777 177777 176166
002722 012777 177017 176164
002730 012777 177777 176164
002736 012777 177777 176140
002744 012777 020417 176130
002752 012777 000071 176144
002760 012777 000040 176114
002766 022777 000100 176106
002774 001401
002776 104200
003000 016777 176154 176074
003006 032777 173577 176064
003014 001401
003016 104001

```

```

TTAGG: MOV #340, @PS ;LOCK OUT INTERRUPTS
        MOV UNNUM, @RSCS2
        MOV #43576, @RSCS1 ;SET ALL
        MOV #177777, @RSBA ;POSSIBLE
        MOV #177777, @RSDA ;REGISTERS
        MOV #177017, @RSER
        MOV #177777, @RSDB
        MOV #177777, @RSWC
        MOV #20417, @RSCS2
        MOV #71, @RSMR
        MOV #40, @RSCS2 ;CLEAR ALL BITS
        CMP #100, @RSCS2 ;DID THE RIGHT BITS CLEAR?
        BEQ +4 ;YES
        HLT ;CS2 ;(417) SHOULD BE CLEARED IN CS2
        MOV UNNUM, @RSCS2 ;GET DRIVE NUMBER
        BIT #173577, @RSCS1 ;DID ALL BITS GET CLEARED
        BEQ +4 ;YES
        HLT ;CS1 ;NO, ALL BITS SHOULD BE 0

```

;TEST BUS ADDRESS REGISTER

```

003020 005777 176062
003024 001401
003026 104020

```

```

TST @RSBA ;IS BA REG. CLEARED
BEQ +4 ;YES
HLT ;BA ;SHOULD BE 0

```

;TEST DISK ADDRESS REGISTER

```

003030 005777 176054
003034 001401
003036 104020

```

```

TST @RSDA ;IS DA CLEARED
BEQ +4 ;YES
HLT ;BA ;SHOULD BE 0

```

;TEST ERROR REG RSER

```

003040 032777 177777 176046
003046 001401
003050 104002

```

```

BIT #177777, @RSER ;DID THESE BITS GET CLEARED
BEQ +4 ;YES
HLT ;ER ;BITS(157015) SHOULD BE CLEARED

```

;TEST RS MAINTENANCE REGISTER

```

003052 032777 000077 176044
003060 001401
003062 104220

```

```

BIT #77, @RSMR ;DID THESE BITS GET CLEARED
BEQ +4 ;YES
HLT ;MR ;BITS(77) SHOULD BE 0

```

;TEST WC REG. IT SHOULD NOT CHANGE

```

003064 022777 177777 176012
003072 001401
003074 104010

```

```

CMP #177777, @RSWC ;DID WC CHANGE
BEQ +4 ;NO
HLT ;WC ;WHY DID IT CHANGE?

```


003076 104400

```

*****
;TEST 3          SET AND CLEAR ALL REGISTERS
*****
TST3:  SCOPE
;CAN WE SET THE FUNCTION BITS IN THE RSCS1 REG.
;BITS 7,6,5,4,3,2&1

```

```

003100 104414
003102 016767 176062 021414
003110 012777 003576 175762
003116 022777 005776 175754
003124 001401
003126 104001
003130 012777 002524 175742
003136 022777 004724 175734
003144 001401
003146 104001
003150 012777 001052 175722
003156 022777 005252 175714
003164 001401
003166 104001
003170 104400

```

```

CLRDK
MOV TIMSV,TIMES ;CLEAR ALL RS REG
MOV #3576,RSCS1 ;GET TIME
CMP #5776,RSCS1 ;SET DISK FUNCTION BITS
BEQ +4 ;ARE THESE BITS SET?
HLT !CS1 ;NO
MOV #2524,RSCS1 ;SHOULD = 3776
CMP #4724,RSCS1 ;SET THESE BITS
BEQ +4 ;DID THEY SET
HLT !CS1 ;YES
MOV #1052,RSCS1 ;SHOULD BE 2725
CMP #5252,RSCS1 ;SET THESE BITS
BEQ +4 ;ARE THEY =?
HLT !CS1 ;YES
;SHOULD = 1252
TST4:  SCOPE
;CLEAR THE FUNCTION BITS

```

```

003172 012777 043576 175700
003200 005077 175674
003204 022777 004200 175666
003212 001401
003214 104001

```

```

MOV #43576,RSCS1 ;SET DISK FUNCTION BITS
CLR RSCS1
CMP #4200,RSCS1 ;IS THE READY BIT SET
BEQ +4 ;YES
HLT !CS1 ;RSCS1 SHOULD = 4200

```

```

*****
;TEST 5          TEST RSCS2
*****
TST5:  SCOPE

```

003216 104400

```

003220 000005
003222 022777 000100 175652
003230 001401
003232 104200
003234 012777 021037 175640
003242 022777 000137 175632
003250 001405
003252 017700 175624
003256 012701 000137
003262 104000

```

```

RESET
CMP #100,RSCS2 ;CLEAR WORLD
BEQ +4 ;DID THEY CLEAR?
HLT !CS2 ;YES
MOV #21037,RSCS2 ;NO
CMP #137,RSCS2 ;SET BITS 21017
BEQ 1$ ;DID THESE BITS GET SET
MOV RSCS2,BAD ;YES
MOV #137,GOOD ;WHAT CS2 SHOULD =
HLT ;CS2 = BAD GOOD = CORRECT ANS

```



```

003264 012777 020025 175610 1S: MOV #20025,ARSCS2 ;SET THESE BITS
003272 022777 000125 175602 CMP #125,ARSCS2 ;DID THESE BITS GET SET
003300 001401 BEQ +4 ;YES
003302 104200 HLT ;CS2 ;NO,CS2 SHOULD = 20125
003304 012777 000012 175570 MOV #12,ARSCS2 ;LOAD THESE BITS
003312 022777 000112 175562 CMP #112,ARSCS2 ;DID THESE BITS GET SET IN CS2
003320 001401 BEQ +4 ;YES
003322 104200 HLT ;CS2 ;BAD = CS2 GOOD = CORRECT ANS
003324 012777 177777 175550 MOV #-1,ARSCS2 ;SET BITS
003332 005077 175544 CLR ARSCS2 ;CLEAR THEM
003336 022777 000100 175536 CMP #100,ARSCS2 ;DID CLEAR WORK
003344 001401 BEQ +4 ;YES
003346 104200 HLT ;CS2 ;R/W BITS DID NOT CLEAR
003350 016777 175604 175524 MOV UNNUM,ARSCS2 ;GET UNIT #
003356 104400

TST6: SCOPE
;CAN WE SET ALL THE RSBA BITS

003360 012777 177777 175520 MOV #177777,ARSB A ;SET THE BITS
003366 022777 177776 175512 CMP #177776,ARSB A ;DID THEY SET
003374 001401 BEQ +4 ;YES
003376 104020 HLT ;BA ;BITS 17776 SHOULD BE SET
003400 012777 125252 175500 MOV #125252,ARSB A ;SET THESE BITS
003406 022777 125252 175472 CMP #125252,ARSB A ;ARE THEY =
003414 001401 BEQ +4 ;YES
003416 104020 HLT ;BA ;SHOULD BE 125252
003420 012777 052524 175460 MOV #52524,ARSB A ;SET THESE BITS
003426 022777 052524 175452 CMP #52524,ARSB A ;ARE THEY =
003434 001401 BEQ +4 ;YES
003436 104020 HLT ;BA ;SHOULD BE 52524

003440 104400

TST7: SCOPE
;FLOAT A 1 THROUGH RSBA

FLOTBA: MOV #2,GOOD ;GET A 2
CLC ;CLEAR CARRY
1S: MOV GOOD,ARSB A ;FLOAT NUMBER
MOV ARSB A,BAD ;GET BA
CMP GOOD,BAD ;COMPARE BA
BEQ +4 ;BA CORRECT
HLT ;BAD=BA GOOD=CORRECT ANS
ROL GOOD ;ROTATE NUMBER
BCC 1S ;LOOP TILL DONE

```

003472 104400

TST10: SCOPE

;CLEAR THE RSBA REGISTER

003474 012777 177777 175404
 003502 005077 175400
 003506 005777 175374
 003512 001401
 003514 104020
 003516 104400

MOV #177777, @RSBA
 CLR @RSBA
 TST @RSBA
 BEQ +4
 HLT !BA

;SET RSBA EQUAL TO ALL ONES
 ;TEST FOR BIT0 SET IN RSBA (READ ONLY BIT)
 ;YES
 ;NO

TST11: SCOPE

;CAN WE SET ALL BITS IN RSWC REGISTER

003520 012777 177777 175356
 003526 022777 177777 175350
 003534 001401
 003536 104010
 003540 012777 125252 175336
 003546 022777 125252 175330
 003554 001401
 003556 104010
 003560 012777 052525 175316
 003566 022777 052525 175310
 003574 001401
 003576 104010
 003600 104400

MOV #177777, @RSWC
 CMP #177777, @RSWC
 BEQ +4
 HLT !WC
 MOV #125252, @RSWC
 CMP #125252, @RSWC
 BEQ +4
 HLT !WC
 MOV #52525, @RSWC
 CMP #52525, @RSWC
 BEQ +4
 HLT !WC

;SET WC BITS
 ;ARE ALL BITS SET
 ;YES
 ;NO
 ;SET THESE BITS
 ;ARE THEY =
 ;YES
 ;SHOULD BE 125252
 ;SET THESE BITS
 ;ARE THEY =
 ;YES
 ;SHOULD BE 152525

TST12: SCOPE

;FLOAT A 1 THROUGH RSWC

003602 012701 000001
 003606 000241
 003610 010177 175270
 003614 017700 175264
 003620 020100
 003622 001401
 003624 104000
 003626 006101
 003630 103367

FLOTWC: MOV #1, GOOD
 CLC
 1\$: MOV GOOD, @RSWC
 MOV @RSWC, BAD
 CMP GOOD, BAD
 BEQ +4
 HLT
 ROL GOOD
 BCC 1\$

;GET A 1
 ;CLEAR CARRY
 ;FLOAT NUMBER
 ;GET WC
 ;COMPARE WC
 ;WC CORRECT
 ;BAD=WC GOOD=CORRECT ANS
 ;ROTATE NUMBER
 ;LOOP TILL DONE


```

003632 104400          ;CLEAR THE WORD COUNT REGISTER
TST13: SCOPE

003634 012777 177777 175242      MOV    #177777,RSWC      ;SET RSWC REGISTER EQUAL TO ALL ONES
003642 005077 175236              CLR    RSWC
003646 005777 175232              TST    RSWC
003652 001401              BEQ    +4              ;DID ALL BITS GET CLEARED
003654 104010              HLT    !WC              ;YES
003656 104400          TST14: SCOPE              ;NO

          ;CAN WE SET ALL THE BITS IN THE RSDA REGISTER.

003660 012777 177777 175222      MOV    #177777,RSDA      ;SET ALL BITS
003666 022777 177777 175214      CMP    #177777,RSDA      ;ARE THE BITS SET
003674 001401              BEQ    +4              ;YES
003676 104004              HLT    !DA              ;NO
003700 012777 125252 175202      MOV    #125252,RSDA      ;SET THESE BITS
003706 022777 125252 175174      CMP    #125252,RSDA      ;ARE THEY =
003714 001401              BEQ    +4              ;YES
003716 104004              HLT    !DA              ;SHOULD BE 125252
003720 012777 052525 175162      MOV    #52525,RSDA      ;SET THESE BITS
003726 022777 052525 175154      CMP    #52525,RSDA      ;ARE THEY =
003734 001401              BEQ    +4              ;YES
003736 104004              HLT    !DA              ;SHOULD BE 52525
003740 104400          TST15: SCOPE

          ;FLOAT A 1 THROUGH RSDA

003742 012701 000001          FLOTDA: MOV    #1,GOOD      ;GET A 1
003746 000241              CLC                      ;CLEAR CARRY
003750 010177 175134          1$:  MOV    GOOD,RSDA      ;FLOAT NUMBER
003754 017700 175130          MOV    RSDA,BAD          ;GET DA
003760 020100              CMP    GOOD,BAD          ;COMPARE DA
003762 001401              BEQ    +4              ;DA CORRECT
003764 104000              HLT                      ;BAD=DA GOOD=CORRECT ANS
003766 006101              ROL     GOOD            ;ROTATE NUMBER
003770 103367              BCC    1$              ;LOOP TILL DONE

```

003772 104400

:CAN WE CLEAR THE RSDA REG.
TST16: SCOPE

003774 012777 177777 175106
 004002 005077 175102
 004006 005777 175076
 004012 001401
 004014 104004
 004016 104400

MOV #177777, @RSDA ;SET RSDA TO ALL ONES
 CLR @RSDA
 TST @RSDA ;TEST FOR ZERO RSDA
 BEQ +4 ;YES
 HLT !DA ;ANS SHOULD BE 0
 TST17: SCOPE

;SET AND CLEAR THE RSER REG.

004020 012777 177017 175066
 004026 022777 177017 175060
 004034 001401
 004036 104002
 004040 112777 000001 175046
 004046 022777 000001 175040
 004054 001401
 004056 104002

MOV #177017, @RSER ;SET THESE BITS
 CMP #177017, @RSER ;DID THEY SET
 BEQ +4 ;YES
 HLT !ER ;RSER SHOULD = 157017
 MOVB #1, @RSER ;A MOVB INST
 CMP #1, @RSER ;SHOULD MODIFY COMPLETE WD
 BEQ +4 ;OK
 HLT !ER

004060 104400

TST20: SCOPE

004062 012777 052005 175024
 004070 022777 052005 175016
 004076 001401
 004100 104002
 004102 104400

MOV #52005, @RSER ;SET THESE BITS
 CMP #52005, @RSER ;DID THEY SET
 BEQ +4 ;YES
 HLT !ER ;ER SHOULD = 52005
 TST21: SCOPE

004104 012777 125012 175002
 004112 022777 125012 174774
 004120 001401
 004122 104002

MOV #125012, @RSER ;SET THESE BITS
 CMP #125012, @RSER ;DID THEY SET
 BEQ +4 ;YES
 HLT !ER ;ER SHOULD = 105012

004124 104400

TST22: SCOPE

004126 012777 177017 174760
004134 005077 174754
004140 005777 174750
004144 001401
004146 104002
004150 104400

MOV #177017, @R5ER
CLR @R5ER
TST @R5ER
BEQ +4
HLT !ER

:SET THESE BITS
:CLEAR THEM
:DID THEY CLEAR
:YES
:SHOULD = 0

TST23: SCOPE

;SET AND CLEAR RSMR

004152 012777 000070 174744
004160 017767 174740 175026
004166 042767 177700 175020
004174 022767 000070 175012
004202 001401
004204 104220
004206 104400

MOV #70, @RSMR
MOV @RSMR, WORK
BIC #177700, WORK
CMP #70, WORK
BEQ +4
HLT !MR

:SET THESE BITS
:PUT INTO WORKABLE REG
:CLEAR JUNK
:DID THEY SET
:YES
:SHOULD = 70

TST24: SCOPE

004210 012777 000070 174706
004216 005077 174702
004222 032777 000077 174674
004230 001401
004232 104220
004234 104400

MOV #70, @RSMR
CLR @RSMR
BIT #77, @RSMR
BEQ +4
HLT !MR

:SET BITS
:CLEAR THEM
:DID THEY CLEAR
:YES
:BITS (77) SHOULD = 0

TST25: SCOPE

004236 012777 000050 174660
004244 017767 174654 174742
004252 042767 177700 174734
004260 022767 000050 174726
004266 001401
004270 104220
004272 104400

MOV #50, @RSMR
MOV @RSMR, WORK
BIC #177700, WORK
CMP #50, WORK
BEQ +4
HLT !MR

:SET BITS
:PUT IN WORKABLE REG
:CLEAR JUNK
:DID THESE BITS SET
:YES
:BITS (50) SHOULD BE SET

TST26: SCOPE

004274 012777 000020 174622
004302 017767 174616 174704
004310 042767 177700 174676
004316 022767 000020 174670
004324 001401
004326 104220

MOV #20, @RSMR
MOV @RSMR, WORK
BIC #177700, WORK
CMP #20, WORK
BEQ +4
HLT !MR

:SET BITS
:PUT INTO WORKABLE REG
:CLEAR JUNK
:DID THEY SET
:YES
:MR SHOULD AT LEAST HAVE A (21)

F03

:TEST 27 TEST ODD BYTE INSTRUCTIONS ON CS1, CS2, WC AND BA

004330 104400

TST27: SCOPE

004332 104414
004334 012777 003566 174536
004342 112777 000005 174564
004350 022777 004766 174522
004356 001401
004360 104001
004362 112777 000032 174510
004370 022777 004632 174502
004376 001401
004400 104001

BITST: CLRDK
MOV #3566, @RSCS1
MOVB #5, @RSCS1B
CMP #4766, @RSCS1
BEQ +4
HLT !CS1
MOVB #32, @RSCS1
CMP #4632, @RSCS1
BEQ +4
HLT !CS1
;CLEAR ALL RS REG
;LOAD CS1
;LOAD BIT
;DID IT LOAD?
;YES
;CS1 SHOULD = 4632

004402 104400

TST30: SCOPE

004404 016777 174550 174470
004412 052777 177400 174462
004420 105077 174512
004424 016701 174530
004430 052701 000100
004434 017700 174442
004440 020001
004442 001401
004444 104000

BITCS2: MOV UNNUM, @RSCS2
BIS #177400, @RSCS2
CLRB @RSCS2B
MOV UNNUM, GOOD
BIS #100, GOOD
MOV @RSCS2, BAD
CMP BAD, GOOD
BEQ +4
HLT
;LOAD UNIT NUMBER
;LOAD ALL BITS
;CLR UPPER BYTE
;GET UNIT NO.
;SET OR BIT
;GET CS2
;IS CS2 CORRECT?
;YES
;LOAD BYTE DID NOT WORK

004446 104400

TST31: SCOPE

004450 012777 025252 174426
004456 112777 000377 174454
004464 022777 177652 174412
004472 001401
004474 104010
004476 112777 000123 174400
004504 022777 177523 174372
004512 001401
004514 104010

BITWC: MOV #25252, @RSWC
MOVB #377, @RSWCB
CMP #177652, @RSWC
BEQ +4
HLT !WC
MOVB #123, @RSWC
CMP #177523, @RSWC
BEQ +4
HLT !WC
;LOAD WC
;LOAD BIT
;DID IT LOAD?
;YES
;NO WC SHOULD =177652
;WC SHOULD = 177523

004516 104400

TST32: SCOPE

004520 012777 025252 174360
004526 112777 000377 174406
004534 022777 177652 174344
004542 001401
004544 104020
004546 112777 000125 174332
004554 022777 177524 174324
004562 001401
004564 104020
004566 104414

BITBA: MOV #25252, @RSBA
MOVB #377, @RSBAB
CMP #177652, @RSBA
BEQ +4
HLT !BA
MOVB #125, @RSBA
CMP #177524, @RSBA
BEQ +4
HLT !BA
CLRDK
;LOAD BA
;LOAD BIT
;DID IT LOAD?
;YES
;BA SHOULD =177652
;BA SHOULD = 177525
;CLEAR ALL RS REG

```
004570 104400
004572 104414
004574 005077 174322
004600 012777 177777 174314
004606 012767 002000 174400
004614 012701 000300
004620 056701 174334
004624 017700 174252
004630 020100
004632 001404
004634 005367 174354
004640 001371
004642 104200
004644 005001
004646 017700 174250
004652 020100
004654 001401
004656 104000
004660 012701 177777
004664 017700 174232
004670 020100
004672 001401
004674 104000
```

```
*****
:TEST 33 LOAD RSDB WITH ALL ONES AND ALL ZEROS
*****
TST33: SCOPE
ZERONE: CLRDK

CLR @RSDB
MOV #177777,@RSDB
MOV #2000,WORK
MOV #300,GOOD
BIS UNNUM,GOOD
2$: MOV @RSCS2,BAD
CMP GOOD,BAD
BEQ 3$
DEC WORK
BNE 2$
HLT !CS2
3$: CLR GOOD
MOV @RSDB,BAD
CMP GOOD,BAD
BEQ .+4
HLT

#-1,GOOD
MOV @RSDB,BAD
CMP GOOD,BAD
BEQ .+4
HLT

: CLEAR ALL RS REG
: LOAD DB WITH ALL 0
: LOAD DB WITH ALL ONES
: TIME OUT ROUTINE
: GET CORRECT FOR CS2
: GET CS2
: IS IT CORRECT?
: YES
: TO WAIT FOR OR
: TO SET
: OR SHOULD BE SET
: LOAD BAD WITH DB
: IS BAD CORRECT
: YES
: COULD NOT FLOAT 0 THROUGH DB
: LOAD GOOD WITH ANS
: GET DATA FROM DB
: IS DB CORRECT
: YES
: BAD SHOULD = 177777
```

```

:TEST INTERRUPT IN THE RH11
:BY MOVING 300 INTO RHCS1
:*****
:TEST 34 TEST INTERRUPT IN RH11
:*****
004676 104400 TST34: SCOPE
004700 104414 INT: CLADK
004702 012777 004754 174220 MOV #PGTRAP,ARSVEC
004710 012777 000340 174214 MOV #340,ARSVCPS
004716 012737 000200 177776 MOV #200,APS
004724 012777 000300 174146 MOV #300,ARSCS1
004732 012767 000500 174254 MOV #500,WORK
004740 005367 174250 1$: DEC WORK
004744 001375 BNE 1$
004746 104001 HLT !CS1
004750 000167 000014 JMP INTDON
004754 022626 PGTRAP: CMP (6)+,(6)+
004756 022777 004200 174114 CMP #4200,ARSCS1
004764 001401 BEQ +4
004766 104001 HLT !CS1
004770 INTDON:

: CLEAR ALL ERRORS
: SET UP VECTOR
: SET TRAP PS
: SET PS AT PRIORITY 4
: THIS SHOULD CAUSE A TRAP
: SETUP LOOP
: DEC LOOP SHOULD
: INTERRUPT BEFORE LOOP IS DONE
: SHOULD NEVER GET HERE
: GET OUT
: TRAP OK
: DID IE CLEAR?
: YES
: IE SHOULD BE CLEARED

```


004770 104400

:TEST 35 MAINTENANCE TIMING TEST

†ST35: SCOPE

:MODULE TESTED G092
:THE FOLLOWING TEST ON THE R504 DISK IS A SINGLE-STEPPED
:MAINTENANCE MODE TEST ON THE R504 TIMING LOGIC. THE ACTUAL
:DISK SURFACE IS SUBSTITUTED BY THE MAINTENANCE RESISTER--I.E.
:THE PROGRAM WILL SUPPLY ALL "DISK CLOCK" PULSES TO DRIVE THE
:TIMING LOGIC. WE ARE TESTING THE ENTIRE TIMING TRACK LOGIC, INCLUDING, INDEX
:PULSE FUNCTION, RESYNC AREA, SECTOR COUNTERS, ETC.

004772	104414			:PUT DRIVE IN MAINTENANCE MODE	
004774	052767	001040	174164	MRTIME: CLRDK	:CLEAR DRIVE REGISTERS
005002	104430			BIS #1040,ONCEE	:SET CLK CNT
005004	104420			MRIND	:SEND INDEX PULSE TO MR REG
005006	022701			MRCK	:CHECK MAINTENANCE REG FOR
005010	104424			22701	:22701
				MRINT	:INIT MAINT MODE (CLEAR MRSP)
005012	104430			MRIND	:BY SENDING 2 CLOCK PULSES
					:SEND MAINT INDEX PULSE
005014	104420			MRCK	:CHECK MAINT REG TO
005016	022701			22701	:EQUAL 22701
005020	104000			HLT	:MR=BAD GOOD=CORRECTIONS
					:COULD NOT INITIALIZE MR REG

:INDEX PULSE SHOULD CLEAR LOOK-AHEAD REG

005022	005777	174072		TST	DRSLA	:IS RSLA CLEARED
005026	001401			BEG	+4	:YES
005030	104224			HLT	MR!LA	:RSLA SHOULD BE CLEARED
						:WITH THE INDEX PULSE

:PERFORM MAINTENANCE CLOCK OPERATION 1024 TIMES TO
:PROVIDE CLOCK TO STEP TIMING THRU RESYNC PERIOD.
:IF SECTOR PULSE IS ASSERTED DURING THIS LOOP
:CHECK SECTOR BOUNDARY COUNTER AND E12

005032	012767	001000	174142	MRTIM1: MOV	#512.,REPT	
005040	104422			MRCLK		:CLOCK MAINT REG WITH AN 11 AND A 1
005042	104420			MRCK		:CHECK MR REG TO
005044	072701			72701		:EQUAL 72701
005046	104000			HLT		:MR = BAD, GOOD = CORRECT ANS
005050	104422			MRCLK		:CLOCK MR
005052	104420			MRCK		:CHECK MR TO
005054	022701			22701		:EQUAL 22701
005056	104000			HLT		:BAD=MR REG GOOD=CORRECTIONS
005060	005367	174116		DEC	REPT	:IS THE LOOP DONE YET?
005064	001365			BNE	MRTIM1	:NO-LOOP

; AFTER ONE MORE CLOCK SECTOR PULSE SHOULD BE ASSERTED
; IF NOT, CHECK SECTOR BOUNDARY COUNTER, SECTOR BOUNDARY FF (E21) AND E12

005066 104422
005070 104420
005072 072301
005074 104000
005076 104422
005100 104420
005102 022301
005104 104000
005106 005777 174006
005112 001401
005114 104224

MRCLK
MRCK
72301
HLT
MRCLK
MRCK
22301
HLT
TST JRSLA
BEQ MRT2
HLT !MR!LA

; CLOCK MAINT REG WITH A 11 AND A 1
; CHECK MR REG TO
; EQUAL 72301
; MR=BAD GOOD=CORRECTIONS
; CLOCK MR WITH 11 AND A 1
; CHECK MAINT REG
; TO EQUAL 22301
; MR=BAD GOOD-CORRECT ANS
; DOES LOOK AHEAD REG=0
; YES-CONT
; LOOK AHEAD REG SHOULD=0

; PERFORM MAINTENANCE CLOCK OPERATION 80 TIMES TO PROVIDE
; CLOCK PULSES TO STEP THRU 1ST SECTOR PRE-AMBLE AREA

005116 005002
005120 012767 000050 174054
005126 104422
005130 104420
005132 073701
005134 104000
005136 104422
005140 104420
005142 023701
005144 104000
005146 005367 174030
005152 001365

MRT2: CLR R2
MOV #40., REPT
MRT2A: MRCLK
MRCK
73701
HLT
MRCLK
MRCK
23701
HLT
DEC REPT
BNE MRT2A

; CLEAR R2 FOR SECTOR COMPARE WITH LA REG
; 80 CLOCKS TO STEP THRU PRE-AMBLE
; CLOCK MR WITH A 11 AND A 1
; CHECK MAINT REG
; EQUAL 73701
; MR = BAD GOOD = CORRECT ANS
; CLOCK MR REG
; CHECK MR REG
; TO EQUAL 23701
; MR = BAD GOOD = CORRECTANS
; REPEAT
; LOOP 40 TIMES

005154 012767 002200 174020
005162 104422
005164 104420
005166 073701
005170 104000
005172 104422
005174 104420
005176 023701
005200 104000
005202 005367 173774
005206 001365

; SUPPLY CLOCKS TO STEP THROUGH THE DATA AREA IN THE SECTOR
MOV #9.*128., REPT

MRT2B: MRCLK
MRCK
73701
HLT
MRCLK
MRCK
23701
HLT
DEC REPT
BNE MRT2B

; 18 CLOCKS PER DATA WORD
; CLOCK MR WITH A 11 AND A 1
; CHECK MAINT REG
; TO EQUAL 73701
; MR = BAD GOOD = CORRECT ANS
; CLOCK MR REG
; CHECK MR REG
; TO EQUAL 23701
; MR=BAD GOOD=CORRECTANS
; REPEAT
; LOOP

;SUPPLY ENOUGH MAINT CLOCKS TO STEP THROUGH THE CRC AREA
;AND THE DEAD BAND ON THE SECTOR

005210	012767	000214	173764	MRT2C:	MOV	#140.,REPT	;AMOUNT OF CLOCKS TO END OF SECTOR
005216	104422				MRCLK		;CLOCK MR WITH A 11 AND A 1
005220	104420				MRCK		;CHECK MAINT REG
005222	073701				73701		;TO EQUAL 73701
005224	104000				HLT		;MR = BAD GOOD = CORRECT ANS
005226	104422				MRCLK		;CLOCK MR REG
005230	104420				MRCK		;CHECK MAINT REG
005232	023701				23701		;TO EQUAL 23701
005234	104000				HLT		;MR=BAD GOOD=CORRECT ANS
005236	005367	173740			DEC	REPT	;REPEAT
005242	001365				BNE	MRT2C	;LOOP
005244	104422				MRCLK		;CLOCK MR REG
005246	104420				MRCK		;CHECK MR REG
005250	073701				73701		;TO EQUAL 73701
005252	104000				HLT		;MR = BAD GOOD = CORRECT ANS

;ONE MORE CLOCK SHOULD CAUSE SECTOR PULSE
;IF NOT, CHECK E16-6

005254	104422				MRCLK		;CLOCK MR WITH A 11 AND A 1
005256	104420				MRCK		;MAINT REG SHOULD
005260	023701				23701		;EQUAL 22301
005262	104000				HLT		;MR=BAD GOOD=CORRECT ANS
005264	104422				MRCLK		;CLOCK MR WITH A 11 AND A 1
005266	104420				MRCK		;MAINT REG
005270	072301				72301		;SHOULD EQUAL 72301
005272	104000				HLT		;MR=BAD GOOD=CORRECT ANS

;LOOK-AHEAD REGISTER SHOULD NOW POINT TO SECTOR 1 (OR 4000 IF INTERLEAVED)

005274	022777	000002	173624		CMP	#2,ARSDT	;INTERLEAVED?
005302	001403				BEQ	3\$	;NO
005304	062702	004000			ADD	#4000,R2	;YES
005310	000402				BR	2\$	;CONT
005312	062702	000100		3\$:	ADD	#100,R2	;INCREMENT SECTOR COMPARE
005316	020277	173576		2\$:	CMP	R2,ARSLA	;LA REG SHOULD=100
005322	001401				BEQ	1\$	;LA IS CORRECT
005324	104224				HLT	!MR!LA	;LA SHOULD=100

:REPEAT NEXT STEPS 62 TIMES. LOOK-AHEAD REGISTER SHOULD INCREMENT
:TO SHOW NEXT SECTOR. CHECKS FOR ALL SECTORS. IF DRIVE IS NOT
:INTERLEAVED, LA = 200, 300, ETC. IF DRIVE IS INTERLEAVED,
:LA = 100, 4100, 200, 4200 ETC. SEE SERVICE MANUAL FOR DETAILS.

005326	012767	000076	173650	1\$:	MOV	#62., REPT1	
005334	012767	005152	173640	MRT3:	MOV	#2666., REPT	
005342	104422			3\$:	MRCLK		:CLOCK MR WITH A 11 AND A 1
005344	005367	173632			DEC	REPT	:STEP THROUGH
005350	001374				BNE	3\$	:SECTOR
005352	104422				MRCLK		:CLOCK MR WITH A 11 AND A 1
005354	104420				MRCK		:MAINT REG
005356	022701				22701		:SHOULD EQUAL 22701
005360	104000				HLT		:MR=BAD GOOD=CORRECT ANS
005362	104422				MRCLK		:1 MORE CLK, SAME SECTOR PULSE
005364	104420				MRCK		:MAINT REG SHOULD
005366	072301				72301		:EQUAL 72301
005370	104000				HLT		:MR=BAD GOOD=CORRECT ANS
005372	022777	000002	173526		CMP	#2, ARSDT	:DRIVE INTERLEAVED?
005400	001420				BEQ	6\$	:NO
005402	032767	001000	173556		BIT	#BIT9, ONCEE	:DO I ADD 4000
005410	001406				BEQ	4\$	:OR SUBTRACT IT FROM WHAT I EXPECT TO
005412	042767	001000	173546		BIC	#BIT9, ONCEE	:FIND IN RSLA
005420	162702	004000			SUB	#4000, R2	
005424	000406				BR	6\$	
005426	052767	001000	173532	4\$:	BIS	#BIT9, ONCEE	
005434	062702	004000			ADD	#4000, R2	
005440	000402				BR	5\$	
005442	062702	000100		6\$:	ADD	#100, R2	:INCREMENT SECTOR COMPARE
005446	017700	173446		5\$:	MOV	ARSLA, BAD	:LA REG SHOULD HAVE INCREMENTED TO NEXT SECTOR
005452	010201				MOV	R2, GOOD	:GET CORRECT ANS FOR RSLA
005454	020100				CMP	GOOD, BAD	:COMPARE FOR CORRECT ANS
005456	001401				BEQ	1\$	:RSLA IS GOOD
005460	104000				HLT		:RSLA=BAD GOOD=CORRECT ANS
005462	005367	173516		1\$:	DEC	REPT1	:REPEAT 62
005466	001322				BNE	MRT3	:TIMES
005470	012767	005152	173504		MOV	#2666., REPT	:COUNT FOR LAST SECTOR
005476	104422			2\$:	MRCLK		:CLOCK
005500	005367	173476			DEC	REPT	:THRU
005504	001374				BNE	2\$	:LAST SECTOR
005506	017700	173406			MOV	ARSLA, BAD	:GET CONTENTS OF RSLA
005512	012701	007777			MOV	#7777, GOOD	:GET CORRECT ANS
005516	020100				CMP	GOOD, BAD	:DOES RSLA EQUAL 7777
005520	001401				BEQ	.+4	:YES
005522	104000				HLT		:BAD=RSLA GOOD=CORRECT ANS

MO3

005524 104400

```
*****
;TEST 36 SECTOR FRACTION TEST
*****
TST36: SCOPE
;MODULE TESTED G092
;CLOCK THROUGH AN ENTIRE TRACK IN MAINT MODE WHILE
;CHECKING FOR THE PROPER OPERATION OF THE LOOK-AHEAD REGISTER AND
;THE SECTOR FRACTION COUNTER. WHEN THE LAST WORD IS BEING TRANSFERRED,
;SECTOR AND FRACTION IS EQUAL TO 7777 TO INDICATE LAST WORD ON THIS TRACK --
;HANDLE END OF TRACK SPECIAL FOR THE LOOK-AHEAD REGISTER WILL CLEAR THE
;FRACTION BITS IF ANOTHER WORD IS Clocked. RSLA SHOULD INDICATE 7700 ON
;ANOTHER MAINTENANCE CLOCK.
```

005526 104414
005530 052767 000040 173430
005536 042767 003000 173422
005544 005067 173424
005550 005002
005552 104430
005554 104420
005556 022701
005560 104424
005562 104430

005564 104420
005566 022701
005570 104000

```
MRT4: CLRDK ;CLEAR DRIVE REGISTERS
      BIS #40,ONCEE ;SET FLAG BITS
      BIC #3000,ONCEE
      CLR MCCNT
      CLR R2 ;CLEAR MAINT CLOCK COUNTER
      MRIND ;CLEAR R2 FOR SECTOR COUNTER
      MRCK ;SEND INDEX PULSE TO MR REG
      22701 ;CHECK MR REG
      MRINT ;TO EQUAL 22701
      MRIND ;INIT MAINT MODE
      ;ISSUE A MAINT INDEX PULSE
      ;TO CLEAR THE DRIVE
      MRCK ;CHECK MAINT REG
      22701 ;TO EQUAL 22701
      HLT ;MR=BAD GOOD=CORRECT ANS
```

;ISSUE 1024 MAINT CLOCKS TO STEP THROUGH THE RESYNC AREA

005572 012767 001000 173402
005600 104422
005602 104420
005604 072701
005606 104000
005610 005777 173304
005614 001401
005616 104204
005620 104422
005622 104420
005624 022701
005626 104000
005630 005777 173264
005634 001401
005636 104204
005640 005367 173336
005644 001355

```
MRT4A: MOV #512.,REPT ;COUNT TO STEP THRU RESYNC AREA
      MRCLK ;CLOCK THROUGH RESYNC
      MRCK ;CHECK MAINT REG
      72701 ;TO EQUAL 72701
      HLT ;MR = BAD GOOD = CORRECT ANS
      TST @RSLA ;IS RSLA=TO 0
      BEQ .+4 ;YES
      HLT !LA ;RSLA SHOULD=0 DURING RESPONSE
      MRCLK ;CLOCK MR REG
      MRCK ;CHECK MR REG
      22701 ;TO EQUAL 22701
      HLT ;BAD=MR GOOD=CORRECT ANS
      TST @RSLA ;IS RSLA=TO 0
      BEQ .+4 ;YES
      HLT !LA ;RSLA SHOULD=0 DURING RESPONSE
      DEC REPT ;LOOP THROUGH
      BNE MRT4A ;RESYNC AREA
```

;ONE MORE PULSE SHOULD CAUSE THE FIRST SECTOR PULSE

005646 104422
005650 104420
005652 072301
005654 104000

```
MRCLK ;CLOCK MR WITH A 11 AND A 1
MRCK ;CHECK MAINT REG FOR SECTOR PULSE
72301 ;MR SHOULD=72301
HLT ;MR=BAD GOOD=CORRECT ANS
```


005656 104422
005660 104420
005662 022301
005664 104000

MRT4B: MRCLK
MRCK
22301
HLT

;CLOCK MR REG WITH A 11 AND A 1
;CHECK MAINT REG
;TO EQUAL 22301
;MR=BAD GOOD=CORRECT ANS

;SECTOR FRACTION BITS IN LOOK-AHEAD REGISTER SHOULD BE CLEARED (EQUAL TO 00)

005666 017700 173226
005672 010201
005674 020100
005676 001401
005700 104000

MOV RSLA,BAD
MOV R2,GOOD
CMP GOOD,BAD
BEQ 1\$
HLT

;GET RSLA
;GET CORRECT ANS
;IS THE RSLA REG CORRECT
;YES
;RSLA=BAD GOOD=CORRECTANS

;STEP THROUGH THE PREAMBLE AREA AND SECTOR DATA
;AREA WHILE CHECKING THE SECTOR FRACTION

005702 012767 000244 173272
005710 104422
005712 017700 173202
005716 010201
005720 020001
005722 001401
005724 104000
005726 005367 173250
005732 001366

1\$: MOV #164.,REPT
MRT4C: MRCLK
MOV RSLA,BAD
MOV R2,GOOD
CMP BAD,GOOD
BEQ 1\$
HLT
1\$: DEC REPT
BNE MRT4C

;FOR FIRST FRACTION CHANGE
;CLOCK MR REG WITH A 11 AND A 1
;GET RSLA
;GET CORRECT ANS
;IS RSLA CORRECT
;YES
;BAD=RSLA GOOD=CORRECT ANS
;LOOP ON
;PREAMBLE AREA

;ONE MORE CLOCK TO CAUSE THE SECTOR FRACTION TO CHANGE

005734 104422
005736 005202
005740 017700 173154
005744 010201
005746 020001
005750 001401
005752 104000

MRCLK
INC R2
MOV RSLA,BAD
MOV R2,GOOD
CMP BAD,GOOD
BEQ 2\$
HLT

;CLOCK MR WITH A 11 AND A 1
;COUNT THE FRACTION
;GET RSLA
;GET CORRECT ANS
;IS RSLA CORRECT?
;YES
;RSLA=BAD GOOD=CORRECT ANS

;FIRST FRACTION CHANGES AFTER 164 MAINT. CLKS, THE REST
;CHANGE AFTER 40 MAINTENANCE CLOCKS

005754 012767 000076 173220
005762 012767 000047 173214
005770 104422
005772 017700 173122
005776 010201
006000 020100
006002 001401
006004 104000
006006 005367 173172
006012 001366

2\$: MOV #62.,REPT
MRT4D: MOV #39.,REPT1
MRT4E: MRCLK
MOV RSLA,BAD
MOV R2,GOOD
CMP GOOD,BAD
BEQ 1\$
HLT
1\$: DEC REPT1
BNE MRT4E

;COUNT FOR WORDS IN A SECTOR
;COUNT FOR SECT FRACT TO CHANGE
;CLOCK MR WITH A 11 AND A 1
;GET RSLA
;GET CORRECT ANS
;IS RSLA CORRECT?
;YES
;RSLA=BAD GOOD=CORRECT ANS
;LOOP

;ONE MORE CLOCK TO CAUSE THE SECTOR FRACTION TO CHANGE

006014	104422			MRCLK		:CLOCK MR WITH A 11 AND A 1
006016	022702	007777		CMP	#7777,R2	:AT THE LAST SECTOR-LAST FRACTION?
006022	001472			BEQ	MRT4F	:YES, FINISH THE SECTOR
006024	005202			INC	R2	:NO, ADD 1 TO FRACTION
006026	017700	173066	4\$:	MOV	2RSLA,BAD	:GET RSLA
006032	022777	000002	173066	CMP	#2,2RSDT	:IS THIS DIRVE INTERLEAVED?
006040	001431			BEQ	12\$	:NO
006042	032767	002000	173116	BIT	#BIT10,ONCEE	:HAS REPT GONE TO ZERO YET FOR THIS SECTOR?
006050	001425			BEQ	12\$	:NO
;RSLA NOW POINTS TO NEXT INTERLEAVED SECTOR BIT 9 IN ONCEE						
;INDICATES WHETHER RSLA SHOULD NOW BE BETWEEN						
;0000-3700(1) OR 4000-7700(0).						
006052	032767	001000	173106	BIT	#BIT9,ONCEE	:SHOULD RSLA BE BETWEEN 0-3700?
006060	001004			BNE	9\$	:YES
006062	052767	001000	173076	BIS	#BIT9,ONCEE	:SET FOR NEXT PASS
006070	000406			BR	10\$	
006072	042767	001000	173066	BIC	#BIT9,ONCEE	:CLEAR FOR NEXT PASS
006100	042702	004000		BIC	#4000,R2	:MAKE EXPECTED RSLA LESS THAN 4000
006104	000404			BR	5\$	
006106	062702	004000		ADD	#4000,R2	:COMPENSATE FOR INTERLEAVING
006112	162702	000100		SUB	#100,R2	
006116	042767	002000	173042	BIC	#BIT10,ONCEE	:CLEAR FLAG FOR NEXT SECTOR
006124	010201			MOV	R2,GOOD	:GET CORRECT ANSWER FOR RSLA
006126	020100			CMP	GOOD,BAD	:IS RSLA CORRECT
006130	001401			BEQ	2\$	:YES
006132	104000			HLT		:RSLA=BAD GOOD=CORRECT ANS
006134	005367	173042	2\$:	DEC	REPT	:HAS SECTOR FRACTION REACHED 77?
006140	001310			BNE	MRT4D	:NO
;CHECK FOR END OF ONE SECTOR OR BEGINNING OF NEXT						
006142	010203			11\$:	MOV	R2,R3
006144	042703	177700		BIC	#177700,R3	:CHECK SECTOR FRACTION
006150	022703	000077		CMP	#77,R3	:END OF SECTOR?
006154	001402			BEQ	3\$	:YES
006156	000167	177474		JMP	MRT4B	:NO, BEGINNING OF NEXT
006162	012767	000025	173014	3\$:	MOV	#21,REPT1
006170	012767	000001	173004	MOV	#1,REPT	:SETUP LOOP TO FINISH
006176	052767	002000	172762	BIS	#BIT10,ONCEE	:THIS SECTOR
006204	000167	177560		JMP	MRT4E	:REPT HAS GONE TO ZERO FOR THIS SECTOR
:LOOP						
006210	012767	000021	172764	MRT4F:	MOV	#17,REPT
006216	104422			1\$:	MRCLK	:CLOCK MR WITH A 11 AND A 1
006220	017700	172674		MOV	2RSLA,BAD	:GET RSLA
006224	010201			MOV	R2,GOOD	:R2 SHOULD=7777
006226	020100			CMP	GOOD,BAD	:IS RSLA CORRECT-END OF DISK?
006230	001401			BEQ	2\$	:YES
006232	104000			HLT		:RSLA=BAD GOOD=CORRECT ANS (7777)
006234	005367	172742	2\$:	DEC	REPT	:FINISH
006240	001366			BNE	1\$	:LOOP

;SECTOR AND FRACTION IS = TO 7777 TO INDICATE LAST WORD ON THIS TRACK
;RSLA SHOULD EQUAL 7700 ON ANOTHER MAINT CLOCK.

006242 104422
006244 017700 172650
006250 012701 007700
006254 020100
006256 001401
006260 104000
006262 104430

006264 017700 172630
006270 005001
006272 020100
006274 001401
006276 104000
006300 104420
006302 022701
006304 104000

MRT4G: MRCLK
MOV 3RSLA,BAD
MOV 7700,GOOD
CMP GOOD,BAD
BEQ 1\$
HLT
1\$: MRIND

MOV 3RSLA,BAD
CLR GOOD
CMP GOOD,BAD
BEQ 2\$
HLT
2\$: MRCK
22701
HLT

;CLOCK MR WITH A 11 AND A 1
;GET RSLA
;GET CORRECT ANS
;IS RSLA CORRECT?
;YES
;RSLA=BAD GOOD=CORRECT ANS
;ISSUE AN INDEX PULSE TO
;CLEAR THE DRIVE
;GET RSLA
;GET CORRECT ANS
;IS RSLA CORRECT?
;YES
;RSLA=BAD GOOD=CORRECT ANS
;CHECK MR REG
;TO EQUAL 22701
;MR=BAD GOOD=CORRECT ANS

006306 104400

:TEST 37 ILLEGAL FUNCTION TEST

TST37: SCOPE:MODULE TESTED M7759, M7770
:TEST ILLEGAL FUNCTION (ILF) IN RSER. SEND AN ILLEGAL
:FUNCTION CODE TO THE DRIVE CONTROL REGISTER WITHOUT SETTING
:THE GO BIT. THE "ILF" BIT SHOULD NOT BE SET. THE "GO" BIT
:IS THEN SET. A CHECK IS THEN MADE FOR "ATA" AND "ERR" TO BE
:SET IN THE DRIVE STATUS REGISTER (RSDS) AND "ILF" IN THE
:DRIVE ERROR REGISTER (RSER). ALL ILLEGAL FUNCTION CODES ARE
:CHECKED.
:ILLEGAL FUNCTIONS ARE DETECTED ON M7759 BY E20-8006310 104414
006312 042767 000040 172646
006320 032767 000002 172640
006326 001002
006330 012702 000003MRILF: CLRDK ;CLEAR ALL THE DRIVE REGISTERS
BIC #BIT5,ONCEE ;CLEAR CLOCK CNT FLAG
BIT #BIT1,ONCEE ;WAS THERE AN ERROR
BNE MRLF1 ;YES DO NOT CHANGE "ILF" CODE
MOV #3,R2 ;SETUP FIRST "ILF" CODE
:PUT DRIVE IN MAINTENANCE MODE006334 104416
006336 104420
006340 022701
006342 104424MRLF1: MRDMD ;PUT DRIVE INTO MAINT MODE
MRCK ;CHECK MR REG TO
22701 ;EQUAL 22701
MRINT ;INIT MAINT MODE (CLEAR MRSP)

:ASSERT A MAINTENANCE MODE DISK "INDEX" PULSE

006344 104430
006346 010277 172526
006352 017700 172534
006356 012701 150600
006362 020100
006364 001440
006366 104402 006372
006442 010267 172546
006446 016746 172542
006452 104406
006454 052767 000002 172504
006462 104000
006464 104040MRLF2: MRIND
MOV R2,RSRCSI ;SEND "ILF" WITH THE "GO" BIT
MOV RSDS,BAD ;GET DRIVE STATUS REG
MOV #150600,GOOD ;GET CORRECT ANS
CMP GOOD,BAD ;IS RSDS CORRECT?
BEQ 1\$;YES
TYPE A,+2 ;ASCIZ <15><12>"ILLEGAL FUNCTION CODE SENT TO DRIVE=" "
MOV R2,WORK ;GET FUNCTION CODE
MOV WORK,-(6) ;PUT WORK ON STACK
TYPES ;TYPE STACK IN OCTAL - SUPPRESS
BIS #BIT1,ONCEE ;SET ERROR BIT SO ILLEGAL FUN DOESN'T CHANGE
HLT ;RSDS=BAD GOOD=CORRECT ANS
HLT !DS006466 042767 000002 172472 1\$:
006474 017700 172414
006500 012701 000001
006504 020100
006506 001404
006510 052767 000002 172450
006516 104000
006520 042767 000002 172440 2\$:BIC #BIT1,ONCEE ;CLEAR ERROR FLAG
MOV RSR,BAD ;GET RSER
MOV #1,GOOD ;GET CORRECT ANS
CMP GOOD,BAD ;DID "ILF" SET IN RSER
BEQ 2\$;YES
BIS #BIT1,ONCEE ;SET ERROR BIT
HLT ;RSER=BAD GOOD=CORRECT ANS
BIC #BIT1,ONCEE ;CLEAR ERROR FLAG

E04

```

;CLEAR THE DRIVE FOR THE NEXT "ILF" CODE PASS
MRCILF: CLDRK
MOV      RSDS,BAD
MOV      #10600,GOOD
CMP      GOOD,BAD
BEQ      1$
TYPE     ,.+2
BIS      #BIT1,ONCE
HLT
BIC      #BIT1,ONCE
MOV      RSER,BAD
CLR      GOOD
CMP      GOOD,BAD
BEQ      2$
BIS      #BIT1,ONCE
TYPE     ,.+2
HLT
BIC      #BIT1,ONCE
;GET NEXT ILLEGAL FUNCTION COE

;CLEAR ERRORS
;GET RSDS REG
;GET CORRECT ANS
;DID "ATA" AND "ERR" CLEAR IN RSDS?
;YES
;.ASCIZ <15><12>"ATA AND ERR IN RSDS SHOULD CLEAR WITH I
;RSDS=BAD GOOD=CORRECT ANS
;CLEAR ERROR FLAG
;GET RSER
;GET CORRECT ANS
;DID ILF CLEAR IN RSER
;YES
;SET ERROR BIT
;.ASCIZ <15><12>"ILF IN RSER SHOULD CLEAR WITH INIT"
;RSER=BAD GOOD=CORRECT ANS
;CLEAR ERROR BIT

;UPDATE ILF
;IS THIS A ILF CODE
;NO-UPDATE IT

;FINISHED ALL ILF CODES GET OUT
;START NEXT ILF FUNCTION

006526 104414 172356
006530 017700 010600
006534 012701 010600
006540 020100
006542 001435
006544 104402 006550
006626 052767 000002 172332
006634 104000
006636 042767 000002 172322 1$:
006644 017700 172244
006650 005001
006652 020100
006654 001431
006656 052767 000002 172302
006664 104402 006670
006736 104000
006740 042767 000002 172220 2$:
;GET NEXT ILLEGAL FUNCTION COE

MRLF3: ADD      #2,R2
CMP      #11,R2
BEQ      MRLF3
CMP      #21,R2
BEQ      MRLF3
CMP      #31,R2
BEQ      MRLF3
CMP      #51,R2
BEQ      MRLF3
CMP      #61,R2
BEQ      MRLF3
CMP      #71,R2
BEQ      MRLF3
CMP      #101,R2
BEQ      ILFDON
JMP      MRLF1

ILFDON:
006746 062702 000002
006752 022702 000011
006756 001773
006760 022702 000021
006764 001770
006766 022702 000031
006772 001765
006774 022702 000051
007000 001762
007002 022702 000061
007006 001757
007010 022702 000071
007014 001754
007016 022702 000101
007022 001402
007024 000167 177304
007030

```


007030 104400

:TEST 40 TEST NO-OP CODES 1 AND 21

TST40: SCOPE

007032 104414
007034 042767 000004 172124
007042 104416
007044 104420
007046 022701
007050 104424

:MODULE TESTED M7755
MROP: CLRDK
BIC #BIT2,ONCEE
MRDMD
MRCK
22701
MRINT

:CLEAR ALL DRIVE REGISTERS
:CLEAR ERROR FLAG
:PUT DRIVE INTO MAINT MODE
:CHECK MR REG TO
:EQUAL 22701
:INIT MAINT MODE (CLEAR MRSP)
:SEND INDEX PULSE
:TESTING CODE 1
:NO CODE 21
:LOAD NO-OP FUNCTION
:LOAD NO-OP FUNCTION
:ANY ERRORS
:NO
:TYPE IT
:TYPE ERROR
:IS RSDS CORRECT
:YES
:RSDS SHOULD
:EQUAL 10600
:CLEAR ERROR FLAG

007052 032767 000010 172106
007060 001031
007062 012777 000001 172010
007070 012767 000001 172116
007076 005777 172012
007102 001403
007104 004767 012254
007110 104040
007112 022777 010600 171772 1\$:
007120 001403
007122 004767 012236
007126 104040
007130 042767 000004 172030 2\$:

BIT #BIT3,ONCEE
BNE 3\$
MOV #1,DRSCS1
MOV #1,WORK
TST DRSER
BEQ 1\$
JSR PC,NOPERR
HLT !DS
CMP #10600,DRSDS
BEQ 2\$
JSR PC,NOPERR
HLT !DS
BIC #BIT2,ONCEE

;TEST NO-OP FUNCTION CODE 21

007136 052767 000010 172022
007144 012767 000021 172042 3\$:
007152 012777 000021 171720
007160 005777 171730
007164 001403
007166 004767 012172
007172 104040
007174 022777 010600 171710 4\$:
007202 001403
007204 004767 012154
007210 104040
007212 042767 000014 171746 5\$:

BIS #BIT3,ONCEE
MOV #21,WORK
MOV #21,DRSCS1
TST DRSER
BEQ 4\$
JSR PC,NOPERR
HLT !DS
CMP #10600,DRSDS
BEQ 5\$
JSR PC,NOPERR
HLT !DS
BIC #14,ONCEE

:TEST TESTING CODE 21 FLAG
:LOAD CODE 21
:LOAD FUNCTION
:ANY ERRORS?
:NO
:YES, TYPE ERROR
:ERROR DURING NO-OP FUNCTION
:IS RSDS CORRECT
:YES
:TYPE ERROR
:RSDS SHOULD=10600
:CLEAR TEST BITS

G04

007220 104400

:TEST 41 TEST NO-OP FUNCTION WITH ERROR BITS SET

TST41: SCOPE

:MODULE TESTED M7759

MRCPR: CLRDK
MRDMD
MRCK
22701
MRINT
MRIND

:CLEAR ALL REGISTERS
:PUT DRIVE INTO MAINT MODE
:CHECK MR REG
:TO EQUAL 22701
:INIT MAINT MODE (CLEAR MRSP)
:SEND INDEX PULSE

007222 104414
007224 104416
007226 104420
007230 022701
007232 104424
007234 104430

007236 012777 177777 171650
007244 116701 171714
007250 017700 171642
007254 020100
007256 001427
007260 104402 007264
007334 104000
007336 012767 000001 171650 1\$:
007344 032767 000010 171614
007352 001004
007354 012777 000001 171516
007362 000406
007364 012767 000021 171622 2\$:
007372 012777 000021 171500
007400 017700 171510 3\$:
007404 012701 177017
007410 020100
007412 001411
007414 104402 007420
007430 004767 012024
007434 104000
007436 017700 171454 4\$:
007442 116701 171516
007446 020100
007450 001411
007452 104402 007456
007466 004767 011766
007472 104000
007474 017700 171412 5\$:
007500 012701 150600
007504 020100
007506 001411
007510 104402 007514
007524 004767 011730
007530 104000
007532 032767 000010 171426 6\$:
007540 001005
007542 052767 000010 171416
007550 000167 177446
007554 042767 000010 171404 7\$:

MOV #1,RSER
MOVB UNCMP,GOOD
MOV RSAS,BAD
CMP GOOD,BAD
BEQ 1\$
TYPE .+2
HLT
MOV #1,WORK
BIT #BIT3,ONCE
BNE 2\$
MOV #1,RSRCS1
BR 3\$
MOV #21,WORK
MOV #21,RSRCS1
MOV RSER,BAD
MOV #177017,GOOD
CMP GOOD,BAD
BEQ 4\$
TYPE .+2
JSR PC,CHG
HLT
MOV RSAS,BAD
MOVB UNCMP,GOOD
CMP GOOD,BAD
BEQ 5\$
TYPE .+2
JSR PC,CHG
HLT
MOV RSDDS,BAD
MOV #150600,GOOD
CMP GOOD,BAD
BEQ 6\$
TYPE .+2
JSR PC,CHG
HLT
BIT #BIT3,ONCE
BNE 7\$
BIS #BIT3,ONCE
JMP MRCPR
BIC #BIT3,ONCE

:LOAD RSR WITH ERRORS
:GET DRIVE UNDER TEST
:GET RSAS REG
:DID ATA BIT SET CAUSED BY ERROR
:YES
:ASCIZ <15><12>"SET ERRORS IN RSR-RSAS IS INCORRECT"
:RSAS=BAD GOOD=CORRECT ANS
:SETUP FOR NO-OP CODE 1
:TESTING CODE 21?
:YES
:SEND NO-OP CODE 1
:CHECK FOR ERRORS
:SETUP FOR CODE 21
:SENT NO-OP CODE 21
:GET RSR REG
:GET CORRECT ANS
:DID RSR CHANGE WITH NO-OP
:NO
:ASCIZ <15><12>"RSER "
:RSER=BAD GOOD=CORRECT ANS
:GET RSAS
:GET CORRECT ANS
:IS RSAS CORRECT
:YES
:ASCIZ <15><12>"RSAS "
:TYPE ERROR
:RSAS=BAD GOOD=CORRECT ANS
:GET RSDS
:GET CORRECT ANS
:DID RSDS CHANGE
:NO
:ASCIZ <15><12>"RSDS "
:TYPE ERROR
:RSDS=BAD GOOD=CORRECT ANS
:TESTING CODE 21
:YES, GET OUT
:SET CODE 21 FLAG
:TEST CODE 21
:DONE CLEAR FLAG AND CONT.

007562 104400

:TEST 42 BLOCK SEARCH TEST 1

↑ST42: SCOPE

:MODULE TESTED: M7759, M7754, M7771, M7770
:A DRIVE SEARCH FUNCTION IS GIVEN TO THE DRIVE FOR SECTOR 3.
:(SECTOR 41, IF SECTOR INTERLEAVING IS ENABLED) THE
:POSITIONING IN PROGRESS BIT (PIP) AND THE DRIVE READY BIT
:(DRY) IN THE DRIVE STATUS REGISTER (RSDS) ARE CHECKED. THE
:ADDRESS CONFIRM BIT (AC) IS ALSO CHECKED.

007564	104414			MRSRCH: CLRDK		:CLEAR ALL REGISTERS
007566	052767	000040	171372	BIS	#BITS, ONCEE	:SET CLOCK FLAG
007574	104416			MRDMD		:PUT DRIVE INTO MAINTENANCE MOE
007576	104420			MRCK		:CHECK MR REG
007600	022701			227C1		:TO EQUAL 22701
007602	104424			MRINT		:INIT MR REG (CLEAR MRSP)
007604	104430			MRIND		:CLOCK INDEX PULSE IN RSMR
007606	012777	000003	171274	MOV	#3, ARSDA	:DO A SEARCH FOR SECTOR 3 OR 41
007614	022777	000002	171304	CMP	#2, ARSDT	:INTERLEAVED?
007622	001403			BEQ	4\$	:NO SECTOR 3
007624	012777	000041	171256	MOV	#41, ARSDA	:YES SECTOR 41
007632	012777	000031	171240	MOV	#31, ARSCS1	:LOAD SEARCH COMMAND (M7759)
007640	104426			DSCK		:CHECK RSDS
007642	030400			30400		:TO EQUAL 30400
007644	104000			HLT		:PIP SHOULD BE SET AND DRY SHOULD
						:BE 0 FOR A DRIVE SEARCH CMD
007646	012767	021506	171326	MOV	#21506, REPT	:STEP THROUGH 3 SECTORS
007654	104422			MRCLK		:CLOCK MR
007656	104426			DSCK		:RSDS SHOULD NOT
007660	030400			30400		:CHANGE TILL CLOCKING IS COMPLETED
007662	104000			HLT		:TO REACH SECTOR 3
007664	005367	171312		DEC	REPT	:KEEP CLOCKING TILL
007670	001371			BNE	1\$	:SECTOR 3 HAS BEEN REACHED
				:NOTE ADD ONE MORE CLOCK PULSE TO		:LOOP COUNTER
007672	104422			MRCLK		:CLOCK MR REG
007674	104426			DSCK		:CHECK FOR "ATA" AND "DRY"
007676	110600			110600		:TO BE SET IN RSDS FOR
007700	104000			HLT		:SEARCH FUNCTION SHOULD BE COMPLETED
007702	022777	104230	171170	CMP	#104230, ARSCS1	:SET RSCS1
007710	001401			BEQ	2\$	:SC IN RSCS1 SHOULD SET BECAUSE OF
007712	104140			HLT	!DS!AS	:COMPLETED SEARCH FUNCTION
007714	016777	171242	171174	MOV	UNITSV, ARSAS	:CLEAR ATA
007722	005777	171170		TST	ARSAS	:DID ATA CLEAR BY WRITING INTO IT?
007726	001401			BEQ	3\$	:YES
007730	104140			HLT	!DS!AS	:RSAS SHOULD=0
007732	022777	004230	171140	CMP	#4230, ARSCS1	:DID SC CLEAR BY CLEARING
007740	001401			BEQ	+4	: "ATA" YES
007742	104140			HLT	!DS!AS	:NO

007744 104400

```

*****
:TEST 43          BLOCK SEARCH TEST 2
*****
TST43: SCOPE

```

```

:MODULE TESTED: M7759, M7754, M7771, M7770
:THIS TEST INITIALIZES A BLOCK SEARCH FUNCTION FOR SECTOR 0. WHEN THE DRIVE
:IS CURRENTLY AT THE DESIRED SECTOR, THE BLOCK SEARCH FUNCTION
:SHOULD NOT BE COMPLETED UNTIL THE DRIVE MAKES A COMPLETE REVOLUTION
:AND REACHES THE BEGINNING OF THE DESIRED SECTOR.

```

007746	104414			MRSRC:	CLRDK		:CLEAR ALL REGISTERS
007750	052767	000040	171210		BIS	#BITS, ONCE	:SET CLOCK FLAG
007756	104416				MRDMD		:PUT DRIVE INTO MAINTENANCE MOE
007760	104420				MRCK		:CHECK MR REG
007762	022701				22701		:TO EQUAL 22701
007764	104424				MRINT		:INIT MR REG (CLEAR MRSP)
				:ASSERT	INDEX	PULSE TO INITIALIZE THE DRIVE	
007766	104430				MRIND		
007770	104420				MRCK		:CHECK MR REG TO EQUAL
007772	022701				22701		:22701
007774	104000				HLT		:
				:STEP THRU	RESYNC PERIOD		
007776	012767	001000	171176		MOV	#512, REPT	
010004	052767	000040	171154		BIS	#BITS, ONCE	:TYPE OUT CLOCK COUNT IF AN ERROR OCCURS
010012	104422			MRRT1:	MRCLK		:CLOCK MR REG
010014	104420				MRCK		:CHECK FOR
010016	072701				72701		:CORRECT DATA
010020	104000				HLT		:MR = BAD GOOD = CORRECT DATA
010022	104422				MRCLK		:CLOCK MR REG
010024	104420				MRCK		:CHECK FOR
010026	022701				22701		:CORRECT DATA
010030	104000				HLT		:ERROR WHILE CLOCKING THROUGH RESYNC PERIOD
010032	005367	171144			DEC	REPT	:FINISH LOOPING
010036	001365				BNE	MRRT1	:THROUGH RESYNC PERIOD
				:ONE MORE	CLOCK PULSE SHOULD ASSERT	SECTOR PULSE SP = 0	
010040	104422				MRCLK		:CLOCK MR REG
010042	104420				MRCK		:MR SHOULD
010044	072301				72301		:EQUALS 72301
010046	104000				HLT		:MR=BAD GOOD=CORRECT ANS
010050	104422				MRCLK		:CLOCK MR REG
010052	104420				MRCK		:CHECK MR
010054	022301				22301		:TO EQUAL 22301
010056	104000				HLT		:MR=BAD GOOD=CORRECT ANS
010060	012767	000100	171114		MOV	#100, REPT	:STEP INTO SECTOR 0
010066	104422			2\$:	MRCLK		:CLOCK MR REG
010070	005367	171106			DEC	REPT	:DO 100 TIMES
010074	001374				BNE	2\$	:DONE YET? NO BR
010076	012777	000031	170774	4\$:	MOV	#31, @RSCSI	:LOAD SEARCH COMMAND (M7759) FOR SECTOR 0
010104	104426				DSCK		:CHECK RSDS
010106	030400				30400		:TO EQUAL 30400
010110	104000				HLT		:PIP SHOULD BE SET AND DRY SHOULD
							:BE 0 FOR A DRIVE SEARCH CMD
010112	012767	021506	171062		MOV	#21506, REPT	:STEP 3 SECTORS BEYOND SECTOR 0

010120 104422
010122 104426
010124 030400
010126 104000
010130 005367 171046
010134 001371

010136 104430
010140 104420
010142 022701
010144 104000

1\$: MRCLK :CLOCK MR
DSCK :RSDS SHOULD NOT
30400 :CHANGE TILL CLOCKING IS COMPLETED
HLT :TO REACH SECTOR 3
DEC REPT :KEEP CLOCKING TILL
BNE 1\$:SECTOR 3 HAS BEEN REACHED
:ASSERT INDEX PULSE TO SIMULATE THE BEGINNING OF THE NEXT REVOLUTION
MRIND
MRCK :CHECK MR REG TO EQUAL
22701
HLT :22701

:STEP THRU RESYNC PERIOD

010146 012767 001000 171026
010154 052767 000040 171004
010162 104422
010164 104420
010166 072701
010170 104000
010172 104422
010174 104420
010176 022701
010200 104000
010202 005367 170774
010206 001365

MRWR1: MOV #512, REPT
BIS #BITS, ONCEE :TYPE OUT CLOCK COUNT IF AN ERROR OCCURS
MRCLK :CLOCK MR REG
MRCK :CHECK FOR
72701 :CORRECT DATA
HLT :MR = BAD GOOD = CORRECT DATA
MRCLK :CLOCK MR REG
MRCK :CHECK FOR
22701 :CORRECT DATA
HLT :ERROR WHILE CLOCKING THROUGH RESYNC PERIOD
DEC REPT :FINISH LOOPING
BNE MRWR1 :THROUGH RESYNC PERIOD

:ONE MORE CLOCK PULSE SHOULD ASSERT SECTOR PULSE
:SP=0 EQUALS SECTOR PULSE

010210 104422
010212 104420
010214 072301
010216 104000
010220 104422
010222 104420
010224 022301
010226 104000

MRCLK :CLOCK MR REG
MRCK :MR SHOULD
72301 :EQUAL 72301
HLT :MR=BAD GOOD=CORRECT ANS
MRCLK :CLOCK MR REG
MRCK :CHECK MR
22301 :TO EQUAL 22301
HLT :MR=BAD GOOD=CORRECT ANS

:NOTE ADD ONE MORE CLOCK PULSE TO LOOP COUNTER

010230 104422
010232 104426
010234 110600
010236 104000
010240 022777 104230 170632
010246 001401
010250 104140
010252 016777 170704 170636 2\$:
010260 005777 170632
010264 001401
010266 104140
010270 022777 004230 170602 3\$:
010276 001401
010300 104140

MRCLK :CLOCK MR REG
DSCK :CHECK FOR "ATA" AND "DRY"
110600 :TO BE SET IN RSDS FOR
HLT :SEARCH FUNCTION SHOULD BE COMPLETED
CMP #104230, @RSCS1 :SET RSCS1
BEQ 2\$:SC IN RSCS1 SHOULD SET BECAUSE OF
HLT !DS!AS :COMPLETED SEARCH FUNCTION
MOV UNITSV, @RSAS :CLEAR ATA
TST @RSAS :DID ATA CLEAR BY WRITING INTO IT?
BEQ 3\$:YES
HLT !DS!AS :RSAS SHOULD=0
CMP #4230, @RSCS1 :DID SC CLEAR BY CLEARING
BEQ +4 : "ATA" YES
HLT !DS!AS ;NO

K04

:TEST 44 DISK REGISTER MODIFIED REFUSED (RMR) ERROR TEST (RSCS1)

TST44: SCOPE

010302 104400

:MODULE TESTED M7759, M7755, M7770
:RMR ERROR IS CAUSED BY WRITTING INTO RSCS1 WHILE DOING A BLOCK SEARCH FUNCTION
:CHECK RMR DECODER, E12, M7755, IF THIS TEST FAILS

010304 104414
010306 042767 000040 170652
010314 104416
010316 104420
010320 022701
010322 104424
010324 012777 000001 170556
010332 012777 000031 170540
010340 104426
010342 030400
010344 104000

RMRC1: CLRDK
BIC #BITS, ONCEE
MRDMD
MRCK
22701
MRINT
MOV #1, ARSDA
MOV #31, ARSCS1
DSCK
30400
HLT

:CLEAR ALL DRIVE REGISTERS
:CLEAR CLK CNT FLAG
:PUT DRIVE INTO MAINT MODE
:CHECK MR REG TO
:EQUAL 22701
:INIT MAINT MODE (CLEAR MRSP)
:LOAD RSDA
:LOAD BLOCK SEARCH FUNCTION
:CHECK RSDS
:TO EQUAL 30400
:DRY IN RSDS SHOULD BE
:CLEARED FOR DRIVE WAS
:ISSURED A BLOCK SEARCH FUNCTION
:RSDS=BAD GOOD=CORRECT ANS
:LOAD A CLEAR FUNCTION
:THIS SHOULD CAUSE AN RMR
:ERROR FOR DRIVE WAS BUSY
:WHEN CLEAR COMMAND WAS GIVEN
:GET RSER REG
:GET CORRECT ANS
:DID RMR SET IN RSER?
:YES

010346 012777 000011 170524

MOV #11, ARSCS1

010354 017700 170534
010360 012701 000004
010364 020100
010366 001410
010370 104402 021531
010374 104402 010400
010406 104000
010410 104426
010412 150600
010414 104000
010416 022777 104230 170454
010424 001401
010426 104040

1\$:

MOV ARSER, BAD
MOV #4, GOOD
CMP GOOD, BAD
BEQ 1\$
TYPE , TRMR
TYPE , +2
HLT
DSCK
150600
HLT
CMP #104230, ARSCS1
BEQ 2\$
HLT !DS

:ASCIZ "RSCS1"
:RSER=BAD GOOD=CORRECT ANS
:CHECK RSDS TO
:EQUAL 150600
:RSDS=BAD GOOD=CORRECT ANS
:DID CORRECT BITS SET IN RSCS1
:YES
:RSCS1 SHOULD=104230
:RSDS SHOULD=150600
:RSER SHOULD=4
:DID CLR CLEAR RSDA
:NO
:RSDA SHOULD=1
:CLEAR ALL REGISTERS
:RSER SHOULD CLEAR
:RSER OK
:RSER SHOULD=0 FOR THE
:CLEAR BIT WAS LOADED IN RSCS2
:RSCS1 SHOULD=4200 FOR THE
:CLEAR BIT WAS LOADED IN RSCS2
:RSCS1 SHOULD=4200

010430 022777 000001 170452 2\$:
010436 001401
010440 104004
010442 104414
010444 005777 170444
010450 001401
010452 104040

4\$:

CMP #1, ARSDA
BEQ 4\$
HLT !DA
CLDK
TST ARSER
BEQ 3\$
HLT !DS

010454 022777 004200 170416 3\$:
010462 001401
010464 104040

CMP #4200, ARSCS1
BEQ +4
HLT !DS

L04

:TEST 45 DISK REGISTER MODIFIED REFUSED (RMR) ERROR TEST (RSDA)

TST45: SCOPE

:MODULE TESTED M7755 M7759 M7770
:RMR ERROR IS CAUSED BY WRITTING INTO RSDA WHILE DOING A BLOCK SEARCH FUNCTION

010466 104400

010470 104414
010472 104416
010474 104420
010476 022701
010500 104424
010502 012777 000001 170400
010510 012777 000031 170362
010516 104426
010520 030400
010522 104000

RMRC2: CLRDK
MRDMD
MRCK
22701
MRINT
MOV #1, RSDA
MOV #31, RSCS1
DSCK
30400
HLT

:CLEAR ALL DRIVE REGISTERS
:PUT DRIVE INTO MAINT MODE
:CHECK MR REG TO
:EQUAL 22701
:INIT MAINT MODE (CLEAR MRSP)
:LOAD RSDA
:LOAD BLOCK SEARCH FUNCTION
:CHECK RSDS
:TO EQUAL 30400
:DRY IN RSDS SHOULD BE
:CLEARED FOR DRIVE WAS
:ISSUED A BLOCK SEARCH FUNCTION
:RSDS=BAD GOOD=CORRECT ANS
:MODIFY RSDA
:THIS SHOULD CAUSE AN RMR
:ERROR FOR DRIVE WAS BUSY
:WHEN COMMAND WAS GIVEN
:GET RSR REG
:GET CORRECT ANS
:DID RMR SET IN RSR?
:YES

010524 005077 170360

CLR RSDA

010530 017700 170360
010534 012701 000004
010540 020100
010542 001410
010544 104402 021531
010550 104402 010554
010562 104000
010564 104426
010566 150600
010570 104000
010572 022777 104230 170300
010600 001401
010602 104040

MOV RSR, BAD
MOV #4, GOOD
CMP GOOD, BAD
BEQ 1\$
TYPE , TRMR
TYPE , +2
HLT
DSCK
150600
HLT
CMP #104230, RSCS1
BEQ 2\$
HLT
!DS

:ASCIZ "RSDA"
:RSER=BAD GOOD=CORRECT ANS
:CHECK RSDS TO
:EQUAL 150600
:RSDS=BAD GOOD=CORRECT ANS
:DID CORRECT BITS SET IN RSCS1
:YES
:RSCS1 SHOULD=104230
:RSDS SHOULD=50400
:RSER SHOULD=4
:DID CLR CLEAR RSDA
:NO
:RSDA SHOULD=1
:CLEAR ALL REGISTERS
:RSER SHOULD CLEAR
:RSER OK
:RSER SHOULD=0 FOR THE
:CLEAR BIT WAS LOADED IN RSCS2
:RSCS1 SHOULD=4200 FOR THE
:CLEAR BIT WAS LOADED IN RSCS2
:RSCS1 SHOULD=4200

010604 022777 000001 170276 2\$:
010612 001401
010614 104004
010616 104414
010620 005777 170270 4\$:
010624 001401
010626 104040

010630 022777 004200 170242 3\$:
010636 001401
010640 104040

CMP #1, RSDA
BEQ 4\$
HLT
!DA
CLRDK
TST RSR
BEQ 3\$
HLT
!DS

CMP #4200, RSCS1
BEQ +4
HLT
!DS

010642 104400

:TEST 46 DISK REGISTER MODIFIED REFUSED (RMR) ERROR TEST (RSE)

TST46: SCOPE

:MODULE TESTED M7759, M7755, M7770
:RMR ERROR IS CAUSED BY WRITTING INTO RSE WHILE DOING A BLOCK SEARCH FUNCTION
:CHECK RMR DECODER, E12-M7755, IF THIS TEST FAILS.

010644 104414
010646 042767 000040 170312
010654 104416
010656 104420
010660 022701
010662 104424
010664 012777 000001 170216
010672 012777 000031 170200
010700 104426
010702 030400
010704 104000

RMRC3: CLRDK
BIC #BITS, ONCEE
MRDMD
MRCK
22701
MRINT
MOV #1, JRSDA
MOV #31, JRSCS1
DSCK
30400
HLT

:CLEAR ALL DRIVE REGISTERS
:CLEAR CLOCK COUNT FLAG
:PUT DRIVE INTO MAINT MODE
:CHECK MR REG TO
:EQUAL 22701
:INIT MAINT MODE (CLEAR MRSP)
:LOAD RSDA
:LOAD BLOCK SEARCH FUNCTION
:CHECK RSDS
:TO EQUAL 30400
:DRY IN RSDS SHOULD BE
:CLEARED FOR DRIVE WAS
:ISSUED A BLOCK SEARCH FUNCTION
:RSDS=BAD GOOD=CORRECT ANS
:MODIFY RSE
:THIS SHOULD CAUSE AN RMR
:ERROR FOR DRIVE WAS BUSY
:WHEN COMMAND WAS GIVEN
:GET RSE REG
:GET CORRECT ANS
:DID RMR SET IN RSE?
:YES

010706 012777 177777 170200

MOV #-1, JRSE

010714 017700 170174
010720 012701 000004
010724 020100
010726 001410
010730 104402 021531
010734 104402 010740
010746 104000
010750 104426
010752 150600
010754 104000
010756 022777 104230 170114
010764 001401
010766 104040

1\$:

MOV JRSE, BAD
MOV #4, GOOD
CMP GOOD, BAD
BEQ 1\$
TYPE , TRMR
TYPE , +2
HLT
DSCK
150600
HLT
CMP #104230, JRSCS1
BEQ 4\$
HLT !DS

:ASCIZ "RSE"
:RSE=BAD GOOD=CORRECT ANS
:CHECK RSDS TO
:EQUAL 150600
:RSDS=BAD GOOD=CORRECT ANS
:DID CORRECT BITS SET IN RSCS1
:YES
:RSCS1 SHOULD=104230
:RSDS SHOULD=150600
:RSE SHOULD=4
:CLEAR ALL REGISTERS
:RSE SHOULD CLEAR
:RSE OK
:RSE SHOULD=0 FOR THE
:CLEAR BIT WAS LOADED IN RSCS2
:RSCS1 SHOULD=4200 FOR THE
:CLEAR BIT WAS LOADED IN RSCS2
:RSCS1 SHOULD=4200

010770 104414
010772 005777 170116
010776 001401
011000 104040

4\$:

CLRDK
TST JRSE
BEQ 3\$
HLT !DS

011002 022777 004200 170070 3\$:
011010 001401
011012 104040

CMP #4200, JRSCS1
BEQ +4
HLT !DS

:TEST 47 DISK REGISTER MODIFIED REFUSED (RMR) ERROR TEST (RSAS)

TST47: SCOPE

011014 104400

:MODULE TESTED: M7759, M7755, M7770
:RMR ERROR SHOULD NOT SET BY WRITTING INTO RSAS WHILE DOING A BLOCK SEARCH FUNCTION
:IF TEST FAILS, CHECK RMR DECODER E12-M7755.

011016 104414
011020 104416
011022 104420
011024 022701
011026 104424
011030 012777 000001 170052
011036 012777 000031 170034
011044 104426
011046 030400
011050 104000

RMRC4: CLRDK
MRDMD
MRCK
22701
MRINT
MOV #1, @RSDA
MOV #31, @RSCS1
DSCK
30400
HLT

:CLEAR ALL DRIVE REGISTERS
:PUT DRIVE INTO MAINT MODE
:CHECK MR REG TO
:EQUAL 22701
:INIT MAINT MODE (CLEAR MRSP)
:LOAD RSDA
:LOAD BLOCK SEARCH FUNCTION
:CHECK RSDS
:TO EQUAL 30400
:DRY IN RSDS SHOULD BE
:CLEARED FOR DRIVE WAS
:ISSUED A BLOCK SEARCH FUNCTION
:RSDS=BAD GOOD=CORRECT ANS
:WRITE INTO ATTENTION SUMMARY REGISTER.
:SHOULD BE NO RMR ERROR BECAUSE
:WRITING RSAS IS ALLOWED ANYTIME.
:GET RSR REG
:GET CORRECT ANS
:DID RMR SET IN RSR?
:NO
:ASCIZ <15><12>"RMR ERROR SHOULD NOT SET WHILE WRITING
:RSER=BAD GOOD=CORRECT ANS
:CHECK RSDS TO
:EQUAL 30400
:RSDS=BAD GOOD=CORRECT ANS
:DID CORRECT BITS SET IN RSCS1
:YES
:RSCS1 SHOULD=4231
:RSDS SHOULD=30400
:RSER SHOULD=0
:CLEAR ALL REGISTERS
:RSER SHOULD CLEAR
:RSER OK
:RSER SHOULD=0 FOR THE
:CLEAR BIT WAS LOADED IN RSCS2
:RSCS1 SHOULD=4200 FOR THE
:CLEAR BIT WAS LOADED IN RSCS2
:RSCS1 SHOULD=4200

011052 005077 170040

CLR @RSAS

011056 017700 170032
011062 012701 000000
011066 020100
011070 001435
011072 104402 011076
011162 104000
011164 104426
011166 030400
011170 104000
011172 022777 004231 167700
011200 001401
011202 104040

1\$:

MOV @RSER, BAD
MOV #0, GOOD
CMP GOOD, BAD
BEQ 1\$
TYPE
..+2
HLT
DSCK
30400
HLT
CMP #4231, @RSCS1
BEQ 4\$
HLT !DS

011204 104414
011206 005777 167702
011212 001401
011214 104040

4\$:

CLDK
TST @RSER
BEQ 3\$
HLT !DS

011216 022777 004200 167654
011224 001401
011226 104040

3\$:

CMP #4200, @RSCS1
BEQ .+4
HLT !DS

011230 104400

:TEST 50 DRIVE SELECT TEST

TST50: SCOPE:MODULE TESTED: M7755
:THE PROGRAM LOADS A DRIVE REGISTER, OF THE DRIVE UNDER TEST, TO ALL ONES.
:THE PROGRAM THEN FINDS A NON-EXISTENT DRIVE AND TRIES TO LOAD ITS
:REGISTER WITH ALL ZEROS. THIS SHOULD CAUSE "NED" TO
:SET IN RSCS2. THE PROGRAM RE-SELECTS THE DRIVE UNDER TEST AND CHECKS
:ITS REGISTER TO SEE IF IT WAS MODIFIED. IT SHOULD CONTAIN ALL ONES.
:CHECK UNIT NO. COMPARATOR, E19-M7755 IF TEST FAILS011232 104414
011234 104416
011236 104420
011240 022701
011242 104424MRDSEL: CLRDK
MRDMD
MRCK
22701
MRINT:CLEAR ALL REGISTERS
:PUT DRIVE INTO MAINT MODE
:CHECK MAINT REG
:TO EQUAL 22701
:INITIALIZE MAINT MODE (CLEAR MRSP)
:BY SENDING 2 CLOCK PULSES
:LOAD DISK ADDR REG OF DRIVE UNDER TEST

011244 012777 177777 167636

:SEARCH FOR NON EXISTENT DRIVES

011252 012767 000401 167734

011260 005001

011262 010177 167614

011266 005777 167622

011272 032777 010000 167602

011300 001005

011302 005201

011304 006167 167704

011310 103460

011312 000763

011314 012777 004000 167556

011322 010167 167672

011326 010177 167550

011332 005077 167552

1\$:

2\$:

MOV #1, ARSDA
FOR NON EXISTENT DRIVES
MOV #401, WORK
CLR GOOD
MOV GOOD, ARSCS2
TST ARSER
BIT #BIT12, ARSCS2
BNE 2\$
INC GOOD
ROL WORK
BCS NEDDON
BR 1\$
MOV #4000, ARSCS1
MOV GOOD, WORK1
MOV GOOD, ARSCS2
CLR ARSDA:LOAD UNIT NO
:IS THIS A NED?
:CHECK
:YES
:UPDATE UNIT NUMBER
:KEEP LOOKING FOR NED
:COULD NOT FIND ANY NON EXISTENT DRIVES
:LOOK FOR NED
:CLEAR NED
:SAVE NED NUMBER
:LOAD UNIT # OF NED INTO RSCS2
:WRITE INTO A NON EXISTENT DRIVE REG
:THIS SHOULD CAUSE NED TO
:SET IN RSCS2
:GET RSCS2
:PUT CORRECT ANS IN GOOD
:BY SETTING NED AND IR
:IS RSCS2 CORRECT?
:YES
:RSCS2=BAD GOOD=CORRECT ANS
:IS CS1 CORRECT
:YES
:TRE SHOULD BE SET IN CS1 BECAUSE
:OF NED ERROR IN RSCS2
:RSCS1 SHOULD=160200011336 017700 167540
011342 052701 010100MOV ARSCS2, BAD
BIS #10100, GOOD

011346 020100

011350 001401

011352 104000

CMP GOOD, BAD
BEQ .+4
HLT

011354 022777 160200 167516

011362 001401

011364 104004

CMP #160200, ARSCS1
BEQ .+4
HLT

011366	005777	167524		TST	DRSAS	:DID ANY ATTENTION BITS SET?
011372	001401			BEQ	:+4	:NO
011374	104100			HLT	:AS	:NO ATTENTION BITS SHOULD BE SET
011376	112777	000100	167530	MOVB	#100,DRSCS1B	:CLEAR TRE
011404	032777	010000	167470	BIT	#NED,DRSCS2	:DID NED CLEAR
011412	001401			BEQ	:+4	:YES
011414	104040			HLT	:DS	:NED DID NOT CLEAR IN RSCS2
011416	016777	167536	167456	MOV	UNNUM,DRSCS2	:BY CLEARING TRE BIT IN RSCS1
011424	022777	177777	167456	CMP	#-1,DRSDA	:LOAD CORRECT UNIT NUMBER
						:DID RSDA GET MODIFIED
						:WHILE WRITING INTO A NON
						:EXISTENT DRIVE?
011432	001443			BEQ	NNDD	:NO
011434	104004			HLT	:DA	:RSDA SHOULD= -1
011436	016700	167556		MOV	WORK1,BAD	:IT GOT MODIFIED WHILE WRITING
011442	016701	167512		MOV	UNNUM,GOOD	:INTO A NED
011446	104000			HLT		:GOOD=DRIVE UNDER TEST
011450	000434			BR	NNDD	:BAD=NON EXISTENT DRIVE THAT WAS
						:IN RSCS2 WHEN RSDA GOT MODIFIED
011452	032767	010000	167506	NEDDON: BIT	#BIT12,ONCEE	:WAS THIS TYPED BEFORE?
011460	001030			BNE	NNDD	:YES
011462	104402	011466		TYPE	:+2	:ASCIZ <15><12>"COULD NOT FIND A NON-EXISTENT DRIVE"
011534	052767	010000	167424	BIS	#BIT12,ONCEE	:SET TYPED MESSAGE FLAG
011542				NNDD:		

011542 104400

:TEST 51 MAINTENANCE MODE WRITE TEST

↑TST51: SCOPE

:MODULE TESTED: M7771, M7753, M7751
:THIS IS AN R504 DISK MAINTENANCE MODE (SINGLE-STEPPED) SECTOR
:WRITE TEST. WE ARE TESTING THE COMPLETE DATA PATH FOR A DATA
:TRANSFER TO THE DISK. MILLER ENCODED DATA TO BOTH SURFACES
:IS CHECKED ALONG WITH CORRECT GENERATION OF THE CRC WORD AT
:THE END OF THE SECTOR. INDEX PULSES, RESYNC, TIMING
:PREAMBLE, AND SECTOR PULSES ARE ALSO CHECKED.

011544 012767 000002 167372
011552 104414
011554 052767 000040 167404
011562 042767 000600 167376
011570 104430
011572 104420
011574 022701
011576 104424

MRWRT: MOV #2,FLAG2 ;SET TEST FLAG
CLRDK ;CLEAR DRIVE REGISTERS
BIS #BITS,ONCEE ;SET TYPE CLOCK COUNT FL.
BIC #600,ONCEE ;CLEAR FLAG BITS
MRIND ;SEND INDEX PULSE TO MR REG
MRCK ;CHECK MR REG
22701 ;TO EQUAL 22701
MRINT ;INIT MAINT MODE (CLEAR MRSP)
;BY SENDING 2 CLOCK PULSES

:FILL MEMORY DATA BUFFER (INBUF) WITH 128 WORDS (1 SECTOR)
:DATA BUFFER WORDS ARE :A WORD OF ALL 0'S
: :A WORD OF ALL 1'S
: :FLOATING 1'S PATTERN (16 WORDS)
: :A PATTERN OF 146314 (110 WORDS)
:

011600 012702 026716
011604 005022
011606 012722 177777
011612 005003
011614 000261
011616 006103
011620 103402
011622 010322
011624 000774
011626 012703 000156

011632 012704 146314
011636 010422
011640 005303
011642 001375

MOV #INBUF,R2 ;GET LOCATION OF OUTBUF
CLR (R2)+ ;CLEAR 1ST LOCATION
MOV #-1,(R2)+ ;2ND WORD OF ALL ONES
CLR R3 ;CLEAR WORK LOC TO GENERATE
SEC ;A PATTERN OF FLOATING ONES
1\$: ROL R3 ;GET PATTERN
BCS 2\$;DONE GET OUT
MOV R3,(R2)+ ;FILL BUFFER
BR 1\$;CONT
2\$: MOV #110.,R3 ;FILL REMAINING PORTION OF

3\$: MOV #146314,R4 ;BUFFER WITH A PATTERN OF 146314.
MOV R4,(R2)+ ;LOAD BUFFER
DEC R3 ;DONE YET?
BNE 3\$;NO

;SETUP CONTROLLER TO TRANSFER 128 WORDS OF DATA (1 SECTOR) TO SECTOR 0

011644 012777 026716 167234
011652 012777 177600 167224
011660 012777 000061 167212
011666 104446

MOV #INBUF,ARSBA ;LOAD BUS ADDR REG
MOV #177600,ARSWC ;LOAD WORD COUNT REG
MOV #61,ARSCS1 ;LOAD WRITE COMMAND
GETSP ;CLOCK ROUTINE TO GET SECTOR PULSE
;TO CLEAR OUT COUNTERS AND REGISTERS
;THAT OTHERWISE COULD NOT BE CLEARED.
;COULD NOT SET SECTOR PULSE (0)
;CLOCK MR 2 TIMES SP = 1

011670 104220
011672 104450

HLT !MR
SPASS

;ASSERT INDEX PULSE TO INITIALIZE THE DRIVE

011674 104430
011676 104420
011700 020501
011702 104000MRIND
MRCK
20501
HLT;CHECK MR REG TO EQUAL
;20501 FOR A
;WRITE COMD HAS BEEN ISSUED

;STEP THRU RESYNC PERIOD

011704 012767 001000 167270
011712 052767 000040 167246
011720 104422
011722 104420
011724 070501
011726 104000
011730 104422
011732 104420
011734 020501
011736 104000
011740 005367 167236
011744 001365MRWRT1: MOV #512.,REPT
BIS #BITS,ONCE
MRCLK
MRCK
70501
HLT
MRCLK
MRCK
20501
HLT
DEC REPT
BNE MRWRT1;TYPE OUT CLOCK COUNT IF ERROR OCCURS
;CLOCK MR REG
;CHECK FOR
;CORRECT DATA
;MR = BAD GOOD = CORRECT DATA
;CLOCK MR REG
;CHECK FOR
;CORRECT DATA
;ERROR WHILE CLOCKING THROUGH RESYNC PERIOD
;FINISH LOOPING
;THROUGH RESYNC PERIOD;ONE MORE CLOCK PULSE SHOULD ASSERT SECTOR PULSE
;SP=0 EQUALS SECTOR PULSE011746 104422
011750 104420
011752 070101
011754 104000
011756 104422
011760 104420
011762 020101
011764 104000MRCLK
MRCK
70101
HLT
MRCLK
MRCK
20101
HLT;CLOCK MR REG
;MR SHOULD
;EQUAL 70101
;MR=BAD GOOD=CORRECT ANS
;CLOCK MR REG
;CHECK MR
;TO EQUAL 20101
;MR=BAD GOOD=CORRECT ANS

;PERFORM 63 DOUBLE MAINT CLOCK OPERATIONS--WRITING PREAMBLE

011766 012767 000077 167206
011774 104422
011776 104420
012000 071501
012002 104000
012004 104422
012006 104420
012010 021501
012012 104000
012014 005367 167162
012020 001365MRWRT2: MOV #63.,REPT
MRCLK
MRCK
71501
HLT
MRCLK
MRCK
21501
HLT
DEC REPT
BNE MRWRT2;CLOCK MR REG
;CHECK MR REG
;TO EQUAL 71501
;MR=BAD GOOD=CORRECT ANS
;CLOCK MR REG
;CHECK MR REG
;TO EQUAL 21501
;MR=BAD GOOD=CORRECT ANS
;DONE YET
;NO LOOP

F05

ZZ-CERSD-0
CERSDD.P11RH70-R504 MAINTENANCE MODE DIAGNOSTIC MACY11 30A(1052) 23-JAN-78 13:38 PAGE 97
23-JAN-78 13:36 TST51 MAINTENANCE MODE WRITE TEST

SEG 0057

;DRIVE SHOULD NOW RECEIVE 1ST WORD TO BE WRITTEN

012022	104422		MRCLK		;CLOCK MR REG
012024	104420		MRCK		;CHECK MR REG
012026	171501		171501		;TO EQUAL 171501
012030	104000		HLT		;MR REG=BAD GOOD=CORRECT ANS
012032	104422		MRCLK		;CLOCK MR REG
012034	104420		MRCK		;MR REG SHOULD
012036	025501		25501		;EQUAL 25501
012040	104000		HLT		;MR REG=BAD GOOD=CORRECT ANS
012042	104422		MRCLK		
012044	104420		MRCK		
012046	175501		175501		
012050	104000		HLT		

;PERFORM NEXT STEP 3 TIMES TO FINISH WRITTING PREAMBLE

012052	012767	000003	167122	MOV	#3,REPT	
012060	104422			MRWRT3:	MRCLK	;CLOCK MR REG
012062	104420				MRCK	;CHECK MR REG
012064	025501				25501	;TO EQUAL 25501
012066	104000				HLT	;MR=BAD GOOD=CORRECT ANS
012070	104422				MRCLK	;CLOCK MR REG
012072	104420				MRCK	;CHECK MR REG
012074	175501				175501	;TO EQUAL 175501
012076	104000				HLT	;MR REG=BAD GOOD=CORRECT ANS
012100	005367	167076		DEC	REPT	;DONE YES?
012104	001365			BNE	MRWRT3	;NO LOOP BACK

;MOVE DATA WORD INTO R504 SHIFT REGISTER (M7753)

012106	104422		MRCLK		;CLOCK MR REG
012110	104420		MRCK		;CHECK MR REG
012112	027501		27501		;TO EQUAL 27501
012114	104000		HLT		;MR=BAD GOOD=CORRECT ANS
012116	104422		MRCLK		;CLOCK MR REG
012120	104420		MRCK		;MR REG SHOULD
012122	123501		123501		;EQUAL 123501
012124	104000		HLT		;MR=BAD GOOD=CORRECT ANS

;ENCODE SYNC 1 (M7751)

012126	104422		MRCLK		;CLOCK MR REG
012130	104420		MRCK		;MR REG SHOULD NOW
012132	073501		73501		;EQUAL 73501
012134	104000		HLT		;MR=BAD GOOD=CORRECT ANS
012136	012705	026716	MOV	#INBUF,R5	;GET STARTING ADDR FOR DATA BUFFER
012142	011504		MOV	(R5),R4	;GET DATA

012144	012767	002167	167042	MOV	#1143.,WORK	;DOING A 1 SECTOR TRANSFER 127 WORDS
						;18 BITS PER WORD-CLOCK LOOPS
						;TAKE CARE OF 2 BITS AT A TIME
						;127 TIMES 9 EQUALS 1143 LOOPS
						;TO GET THROUGH SECTOR (LAST WORD DONE SEPARATELY).
012152	042767	000200	167006	BIC	#BIT7,ONCE	;CLEAR LAST WORD FLAG
012160	052767	000100	167000	BIS	#BIT6,ONCE	;SET 1ST TRANSFER WORD FLAG
012166	104432			XBIT		;GET 2 BITS OF DATA
012170	104434			CLKD1		;SEND FIRST CLOCK PULSE
						;AND CALCULATE MR REG
						;FOR CORRECT DATA (MWD+MWD)
012172	104000			HLT		;MR REG NOT CORRECT
012174	104436			CLKD2		;SEND 2ND CLOCK PULSE TO
						;COMPLETE TRANSFER OF 2 BITS
						;CALCULATE CORRECT ANS FOR
						;MR REG (MWD+MWD)
						;MR=BAD GOOD=CORRECT ANS
012176	104000			HLT		;ON LAST WORD YET?
012200	032767	000200	166760	BIT	#BIT7,ONCE	;YES
012206	001015			BNE	2\$	;ON CRC WORD YET?
012210	032767	000400	166750	BIT	#BIT8,ONCE	;YES
012216	001040			BNE	3\$	;DONE WITH 127 WORDS?
012220	005367	166770		DEC	WORK	;NO
012224	001360			BNE	1\$	
012226	052767	000200	166732	BIS	#BIT7,ONCE	;SET LAST WORD FLAG
012234	012767	000012	166752	MOV	#10.,WORK	;SET UP TO TRANSFER LAST WORD
012242	005367	166746		DEC	WORK	;DONE YET?
012246	001347			BNE	1\$	;NO
012250	052767	000400	166710	BIS	#BIT8,ONCE	;SET TRANSFERRING CRC WORD
012256	042767	000200	166702	BIC	#BIT7,ONCE	;CLEAR LAST WORD FLAG
012264	004767	011262		JSR	PC,GENCRC	;GENERATE CRC WORD
						;AND LEAVE IN "WORK"
012270	012702	026716		MOV	#INBUF,R2	;GO TO END
012274	062702	000400		ADD	#400,R2	;OF DATA BUFFER
012300	016712	166710		MOV	WORK,R2	;LOAD CRC WORD
012304	010205			MOV	R2,R5	;RESET POINTER FOR
012306	162705	000002		SUB	#2,R5	;R5 FOR CRC WD
012312	012767	000012	166674	MOV	#10.,WORK	;SETUP TO XFER CRC
012320	005367	166670		DEC	WORK	;DONE YET
012324	001320			BNE	1\$	;NO

;EBL SHOULD NOW ASSERT

012326 104422
012330 104420
012332 153501
012334 104000

MRCLK
MRCK
153501
HLT

;CLOCK MR REG TO STOP THROUGH
;THE R504 DISK SECTOR DEAD BAND AREA
;CHECK MR REG
;TO EQUAL 103501
;MR REG=BAD GOOD=CORRECT ANS

;LOOP 6 TIMES

012336 012767 000006 166636
012344 104422
012346 104420
012350 003501
012352 104000
012354 104422
012356 104420
012360 153501
012362 104000
012364 005367 166612
012370 001365

4\$: MOV #6, REPT
MRCLK
MRCK
3501
HLT
MRCLK
MRCK
153501
HLT
DEC REPT
BNE 4\$

;CLOCK MR REG
;CHECK MR REG
;TO EQUAL 53501
;MR=BAD GOOD=CORRECT ANS
;CLOCK MR REG
;CHECK MR REG
;TO EQUAL 153501
;MR=BAD GOOD=CORRECT ANS
;DONE LOOPING YET?
;NO

;FINISH UP

012372 104422
012374 104420
012376 003501
012400 104000
012402 104422
012404 104420
012406 151501
012410 104000

MRCLK
MRCK
3501
HLT
MRCLK
MRCK
151501
HLT

;CLOCK MR REG
;CHECK MR REG
;TO EQUAL 3501
;MR REG=BAD GOOD=CORRECT ANS
;CLOCK MR REG
;CHECK MR REG
;TO EQUAL 151501
;MR=BAD GOOD=CORRECT ANS

;TRANSFER SHOULD NOW BE COMPLETE

012412 104422
012414 104420
012416 002701
012420 104000

MRCLK
MRCK
2701
HLT

;CLOCK MR REG
;CHECK MRE REG
;TO EQUAL 2701
;MR=BAD GOOD=CORRECT ANS

;NOW TEST CONTROLLER

012422 005777 166452
012426 100001
012430 104014
012432 005777 166446
012436 001401
012440 104010
012442 022777 000001 166440
012450 001401
012452 104004
012454 032767 000002 166462
012462 001002
012464 000137 021014

5\$: TST @RSCS1
BPL 5\$
HLT !DA!WC
TST @RSWC
BEQ +4
HLT !WC
CMP #1, @RSDA
BEQ +4
HLT !DA
BIT #BIT1, FLAG2
BNE +6
JMP @#MRVR2

;ANY ERRORS?
;NO
;YES
;DID WC GO TO 0
;YES
;WC SHOULD BE = TO 0
;DOES RSDA=1
;YES
;RSDA SHOULD=1
;IN MAINT VERIFY TEST
;NO
;YES, GO TO VERIFY TEST

012470 104400

```

*****
:TEST 52 MAINTENANCE READ TEST
*****
TST52: SCOPE

```

```

:MODULE TESTED: M7771, M7753, M7751
:THIS IS AN RS04 DISK MAINTENANCE MODE (SINGLE-STEPPED) SECTOR
:READ TEST. WE ARE TESTING THE COMPLETE DATA PATH FROM THE
:DISK DECODING LOGIC TO CORE MEMORY. (THE PHASE LOCK LOOP IS
:NOT TESTED IN MAINTENANCE MODE.)

```

```

012472 104414
012474 052767 000040 166464
012502 042767 147716 166456
012510 104430
012512 104420
012514 022701
012516 104424

```

```

MRRD: CLRDK ;CLEAR DRIVE REGISTERS
      BIS ;SET TYPE CLOCK COUNT FLAG
      BIC #147716,ONCE ;CLEAR ALL OTHER FLAG BITS
      MRIND ;SEND INDEX PULSE TO MR REG
      MRCK ;CHECK MR REG
      22701 ;TO EQUAL 22701
      MRINT ;INIT MAINT MODE (CLEAR MRSP)
          ;BY SENDING 2 CLOCK PULSES

```

012520 005067 166420

CLR FLAG2 ;CLEAR FLAG TEST BITS

```

:FILL MEMORY DATA BUFFER (INBUF) WITH 128 WORDS (1 SECTOR)
:DATA BUFFER WORDS ARE :A WORD OF ALL 0'S
:                          A WORD OF ALL 1'S
:                          FLOATING 1'S PATTERN (16 WORDS)
:                          A PATTERN OF 146314 (110 WORDS)

```

```

012524 012702 026716
012530 005022
012532 012722 177777
012536 005003
012540 000261
012542 006103
012544 103402
012546 010322
012550 000774
012552 012703 000156
012556 012704 146314
012562 010422
012564 005303
012566 001375

```

```

      MOV #INBUF,R2 ;GET LOCATION OF INBUF
      CLR (R2)+ ;CLEAR 1ST LOCATION
      MOV #-1,(R2)+ ;2ND WORD OF ALL ONES
      CLR R3 ;CLEAR WORK LOC TO GENERATE
      SEC ;A PATTERN OF FLOATING ONES
1$: ROL R3 ;GET PATTERN
      BCS 2$ ;DONE GET OUT
      MOV R3,(R2)+ ;FILL BUFFER
      BR 1$ ;CONT
2$: MOV #110,R3 ;FILL REMAINING PORTION OF
      MOV #146314,R4 ;BUFFER WITH A PATTERN OF 146314
3$: MOV R4,(R2)+ ;LOAD BUFFER
      DEC R3 ;DONE YET
      BNE 3$ ;NO

```

```

:NOTE:
:INBUF CONTAINS THE TABLE OF DATA WHICH IS "READ".
:VIA THE MRDB AND MRDT BITS IN RSMR
:OUTBUF IS WHERE THE DATA WORDS FROM THE
:MASSBUS ARE STORED

```

J05

;SETUP CONTROLLER TO TRANSFER 128 WORDS OF DATA (1 SECTOR) FROM SECTOR 0

012570 012777 027516 166310
012576 012777 177600 166300
012604 012777 000071 166266
012612 012702 000200
012616 012703 027516
012622 005023
012624 005302
012626 001375
012630 104446

4\$:

MOV #OUTBUF,ARSBA
MOV #177600,ARSWC
MOV #71,ARSC51
MOV #200,R2
MOV #OUTBUF,R3
CLR (R3)+
DEC R2
BNE 4\$
GETSP

;LOAD BUSS ADDR REG
;LOAD WORD COUNT REG
;LOAD READ COMMAND
;CLEAR THE OUTBUF TABLE
;SO THAT WHEN THE READ
;IS FINISHED, WE CAN COMPARE
;WHAT WE GOT (OUTBUF) WITH WHAT
;WE EXPECTED (INBUF).
;CLOCK ROUTINE TO GET SECTOR PULSE
;TO CLEAR OUT COUNTERS AND REGISTERS
;THAT OTHERWISE COULD NOT BE CLEARED.
;COULD NOT SET SECTOR PULSE (0)
;CLOCK MR REG 2 TIMES SP = 1

012632 104220
012634 104450

HLT !MR
SPASS

;ASSERT INDEX PULSE TO INITIALIZE THE DRIVE

012636 104430
012640 104420
012642 022601
012644 104000

MRIND
MRCK
22601
HLT

;CHECK MR REG TO EQUAL
;22601 FOR A
;READ COMD

;STEP THRU RESYNC PERIOD

012646 012767 001000 166326
012654 052767 000040 166304
012662 104422
012664 104420
012666 072601
012670 104000
012672 104422
012674 104420
012676 022601
012700 104000
012702 005367 166274
012706 001365

MRRD1:

MOV #512,REPT
BIS #BITS,ONCE
MRCLK
MRCK
72201
HLT
MRCLK
MRCK
22601
HLT
DEC REPT
BNE MRRD1

;TYPE OUT CLOCK COUNT IF ERRORS OCCUR
;CLOCK MR REG
;CHECK FOR
;CORRECT DATA
;MR=BAD GOOD=CORRECT DATA
;CLOCK MR REG
;CHECK FOR
;CORRECT DATA
;ERROR WHILE CLOCKING THROUGH RESYNC
;FINISH LOOPING
;THROUGH RESYNC PERIOD

;ONE MORE CLOCK PULSE SHOULD ASSERT SECTOR PULSE
;SP=0 EQUALS SECTOR PULSE

012710 104422
012712 104420
012714 072201
012716 104000
012720 104422
012722 104420
012724 022201
012726 104000

MRCLK
MRCK
72201
HLT
MRCLK
MRCK
22201
HLT

;CLOCK MR REG
;MR SHOULD
;EQUAL 72201
;MR=BAD GOOD=CORRECT ANS
;CLOCK MR REG
;CHECK MR
;TO EQUAL 22201
;MR=BAD GOOD=CORRECT ANS

;PERFORM 71 DOUBLE MAINT CLOCK OPERATIONS--

012730	012767	000107	166244	MOV	#71.,REPT	
012736	104432			MRRD2:	MRCLK	:CLOCK MR REG
012740	104420				MRCK	:CHECK MR REG
012742	073601				73601	:TO EQUAL 73601
012744	104000				HLT	:MR=BAD GOOD=CORRECT ANS
012746	104422				MRCLK	:CLOCK MR REG
012750	104420				MRCK	:CHECK MR REG
012752	023601				23601	:TO EQUAL 23601
012754	104000				HLT	:MR=BAD GOOD=CORRECT ANS
012756	005367	166220		DEC	REPT	:DONE YET
012762	001365			BNE	MRRD2	:NO LOOP
012764	104422			MRCLK		:CLOCK MR REG
012766	104420			MRCK		:CHECK MR REG
012770	073601			73601		:TO EQUAL 73601
012772	104000			HLT		:MR=BAD GOOD=CORRECT ANS

;READ SYNC"1"

012774	012777	000055	166122	MOV	#55,ARSMR	
013002	012777	000045	166114	MOV	#45,ARSMR	
013010	104420			MRCK		:CHECK MR REG
013012	023645			23645		:TO EQUAL
013014	104000			HLT		:CONTENTS OF GOOD
013016	012777	000055	166100	MOV	#55,ARSMR	
013024	012777	000045	166072	MOV	#45,ARSMR	
013032	104420			MRCK		
013034	173645			173645		
013036	104000			HLT		

;READ DATA

013040	005067	166160		MRD3:	CLR	WORK3	:CLEAR CLOCK COUNT FOR DATA WD
013044	012705	026716			MOV	#INBUF,R5	:GET STARTING ADDRESS FOR DATA BUFFER
013050	162705	000002			SUB	#2,R5	
013054	012767	000025	166122		MOV	#21.,REPT1	:SETUP COUNTER FOR 1ST SB BIT
013062	012767	002200	166112		MOV	#1152.,REPT	:SETUP COUNTER TO TRANSFER
							:128 WORDS-9X128=1152
							:2 CLOCKS PER 2 BITS OF DATA
							:GET 2 DATA BITS
013070	104444			15:	RBIT		:CLOCK MR
013072	104440				CLKR1		:MR REG NOT CORRECT
013074	104000				HLT		:CLOCK MR REG
013076	104442				CLKR2		:MR REG NOT CORRECT
013100	104000				HLT		:DONE WITH DATA BUFFER YET?
013102	005367	166074		DEC	REPT		:NO
013106	001370			BNE	15		

013110	032767	000400	166050	2\$:	BIT	#BIT8,ONCEE	:DID WE ALREADY DO CRC?
013116	001030				BNE	3\$	:YES
013120	052767	000400	166040		BIS	#BIT8,ONCEE	:NO SET CRC FLAG
013126	016767	166052	166036		MOV	REPT1,SAVEE	:SAVE REPT1
013134	004767	010412			JSR	PC,GENCRC	:GENERATE CRC WORD
							:AND LEAVE IN LOC "WORK"
013140	012702	026716			MOV	#INBUF,R2	
013144	016767	166022	166032		MOV	SAVEE,REPT1	:RESTORE REPT1
013152	062702	000400			ADD	#400,R2	:STORE CRC WORD AT END OF
013156	016712	166032			MOV	WORK,R2	:INBUF TABLE
013162	010205				MOV	R2,R5	
013164	162705	000002			SUB	#2,R5	
013170	012767	000011	166004		MOV	#9,REPT	:SETUP TO TRANSFER 1 WD
013176	000734				BR	1\$	:TRANSFER CRC WD
013200	104422			3\$:	MRCLK		:CLOCK MR REG
013202	104420				MRCK		:CHECK MR REG
013204	003601				3601		:TO EQUAL
013206	104000				HLT		:3601
013210	104422				MRCLK		:CLOCK MR REG
013212	104420				MRCK		:CHECK MR
013214	153601				153601		:TO EQUAL
013216	104000				HLT		:153601
013220	104422				MRCLK		:CLOCK MR REG
013222	104420				MRCK		:CHECK MR
013224	007601				7601		:TO EQUAL
013226	104000				HLT		:7601
013230	104422				MRCLK		:CLOCK MR REG
013232	104420				MRCK		:CHECK MR
013234	153601				153601		:TO EQUAL
013236	104000				HLT		:153601

:PERFORM 8 DOUBLE MAINTENANCE CLOCK OPERATIONS
:STEP INTO END OF SECTOR DEAD BAND
:EBL IS NOW ASSERTED

013240	012767	000010	165734	MRD4:	MOV	#8,REPT	
013246	104422			1\$:	MRCLK		:CLOCK MR REG
013250	104420				MRCK		:CHECK MR REG
013252	003601				3601		:TO EQUAL
013254	104000				HLT		:3601
013256	104422				MRCLK		:CLOCK MR REG
013260	104420				MRCK		:CHECK MR
013262	153601				153601		:REG TO
013264	104000				HLT		:EQUAL 153601
013266	005367	165710			DEC	REPT	:DONE YET?
013272	001365				BNE	1\$	:NO

:PERFORM ONE MAINTENANCE CLOCK OPERATION
:SHOULD GET STROBE BUFFER

013274	104422				MRCLK		:CLOCK MR REG
013276	104420				MRCK		:CHECK MR
013300	007601				7601		:REG TO
013302	104000				HLT		:EQUAL 7601

:PERFORM ONE MAINTENANCE CLOCK OPERATION
:SHOULD COMPLETE TRANSFER.

013304	104422		MRD5:	MRCLK		:CLOCK MR REG
013306	022777	004270		CMP	#4270, RSCS1	:ANY ERRORS?
013314	001401	165564		BEQ	1\$	:NO
013316	104054			HLT	!DA!DS!WC	
013320	005777	165560	1\$:	TST	RSCWC	:DID WC GO TO 0
013324	001401			BEQ	+4	:YES
013326	104010			HLT	!WC	:WC REG SHOULD=0
013330	022777	000001		CMP	#1, RSDA	:DOES RSDA=1
013336	001401	165552		BEQ	+4	:YES
013340	104004			HLT	!DA	:RSDA SHOULD=1

:COMPARE DATA READ WITH INPUT BUFFER
:WILL ONLY TYPEOUT 10 ERRORS --- BUT IF SW12 IS SET
:IT WILL TYPE OUT ALL ERRORS

013342	012700	026716		MRD6:	MOV	#INBUF, BAD	:GET STARTING LOC OF EXPECTED DATA
013346	012701	027516			MOV	#OUTBUF, GOOD	:GET STARTING LOC OF DATA "READ" FROM DISK
013352	012767	000012	165622		MOV	#12, REPT	:SET UP ERROR COUNTER
013360	012705	000201			MOV	#201, R5	:COMPARE 1 SECTOR
013364	005305			3\$:	DEC	R5	:DONE WITH SECTOR
013366	001433				BEQ	2\$	:YES GET OUT
013370	022021				CMP	(BAD)+, (GOOD)+	:IS DATA CORRECT?
013372	001774				BEQ	3\$	:YES
013374	032777	010000	164166		BIT	#BIT12, RSWR	:TYPE ALL ERRORS?
013402	001003				BNE	1\$	:YES
013404	005367	165572			DEC	REPT	:TYPED OUT 10 ERRORS YET?
013410	001422				BEQ	2\$	:YES GET OUT
013412	024041			1\$:	CMP	-(BAD), -(GOOD)	:GET ERROR
013414	104000				HLT		:TYPE OUT ERROR
013416	010067	165572			MOV	BAD, WORK	
013422	104402	013426			TYPE	+2	:ASCIZ "BAD ADDRESS= "
013444	016746	165544			MOV	WORK, -(6)	:PUT WORK ON STACK
013450	104406				TYPES		:TYPE STACK IN OCTAL - SUPPRESS
013452	022021				CMP	(BAD)+, (GOOD)+	
013454	000743				BR	3\$	
013456				2\$:			:DONE

013456 104400

```

*****
:TEST S3 MAINTENANCE MODE DATA WRITE CHECK TEST
*****
TST53: SCOPE

```

```

:MODULE TESTED: M7771, M7753, M7751
:A ONE SECTOR TRANSFER IS DONE WITH A WRITE CHECK FUNCTION.
:WITHIN THE R504, A WRITE CHECK FUNCTION IS IDENTICAL TO A
:READ FUNCTION.

```

```

013460 104414
013462 052767 000040 165476
013470 042767 147716 165470
013476 104430
013500 104420
013502 022701
013504 104424

```

```

MRWCK: CLRDK ;CLEAR DRIVE REGISTERS
        BIS #BITS ONCEE ;SET TYPE CLOCK COUNT FLAG
        BIC #147716,ONCEE ;CLEAR ALL OTHER FLAG BITS
        MRIND ;SEND INDEX PULSE TO MR REG
        MRCK ;CHECK MR REG
        22701 ;TO EQUAL 22701
        MRINT ;INIT MAINT MODE (CLEAR MRSP)
        ;BY SENDING 2 CLOCK PULSES

```

013506 012767 000004 165430

MOV #4,FLAG2 ;SET WC FLAG FOR CLKR1 ROUTINE

```

:FILL MEMORY DATA BUFFER (INBUF) WITH 128 WORDS (1 SECTOR)
:DATA BUFFER WORDS ARE :A WORD OF ALL 0'S
:                        :A WORD OF ALL 1'S
:                        :FLOATING 1'S PATTERN (16 WORDS)
:                        :A PATTERN OF 146314 (110 WORDS)

```

```

013514 012702 026716
013520 005022
013522 012722 177777
013526 005003
013530 000261
013532 006103
013534 103402
013536 010322
013540 000774
013542 012703 000156
013546 012704 146314
013552 010422
013554 005303
013556 001375

```

```

MOV #INBUF,R2 ;GET LOCATION OF INBUF
CLR (R2)+ ;CLEAR 1ST LOCATION
MOV #-1,(R2)+ ;2ND WORD OF ALL ONES
CLR R3 ;CLEAR WORK LOC TO GENERATE
SEC ;A PATTERN OF FLOATING ONES
1$: ROL R3 ;GET PATTERN
BCS 2$ ;DONE GET OUT
MOV R3,(R2)+ ;FILL BUFFER
BR 1$ ;CONT
2$: MOV #110,R3 ;FILL REMAINING PORTION OF
MOV #146314,R4 ;BUFFER WITH A PATTERN OF 146314
3$: MOV R4,(R2)+ ;LOAD BUFFER
DEC R3 ;DONE YET
BNE 3$ ;NO

```

;SETUP CONTROLLER TO TRANSFER 128 WORDS OF DATA (1 SECTOR) FROM SECTOR 0

```

013560 012777 026716 165320
013566 012777 177600 165310
013574 012777 000051 165276
013602 104446

```

```

MOV #INBUF,AR5BA ;LOAD BUS ADDR REG
MOV #177600,AR5WC ;LOAD WORD COUNT REG
MOV #51,AR5CS1 ;LOAD WRITE CHECK COMMAND
GETSP ;CLOCK ROUTINE TO GET SECTOR PULSE
;TO CLEAR OUT COUNTERS AND REGISTERS
;THAT OTHERWISE COULD NOT BE CLEARED.
;COULD NOT SET SECTOR PULSE (0)
;CLOCK MR SECTOR PULSE = 1

```

```

013604 104220
013606 104450

```

```

HLT !MR
SPASS

```


806

013610 104430
013612 104420
013614 022701
013616 104000

;ASSERT INDEX PULSE TO INITIALIZE THE DRIVE

MRIND
MRCK
22701
HLT

;CHECK MR REG TO EQUAL
22701 FOR A
;WRITE CHECK COMMAND

;STEP THRU RESYNC PERIOD

013620 012767 001000 165354
013626 052767 000040 165332
013634 104422
013636 104420
013640 072701
013642 104000
013644 104422
013646 104420
013650 022701
013652 104000
013654 005367 165322
013660 001365

MRWCK1: MOV #512.,REPT
BIS #BITS,ONCE
MRCLK
MRCK
72701
HLT
MRCLK
MRCK
22701
HLT
DEC REPT
BNE MRWCK1

;TYPE OUT CLOCK COUNT IF ERRORS OCCUR
;CLOCK MR REG
;CHECK FOR
;CORRECT DATA
;MR=BAD GOOD=CORRECT DATA
;CLOCK MR REG
;CHECK FOR
;CORRECT DATA
;ERROR WHILE CLOCKING THROUGH RESYNC
;FINISH LOOPING
;THROUGH RESYNC PERIOD

;ONE MORE CLOCK PULSE SHOULD ASSERT SECTOR PULSE
;SP=0 EQUALS SECTOR PULSE

013662 104422
013664 104420
013666 072301
013670 104000
013672 104422
013674 104420
013676 022301
013700 104000

MRCLK
MRCK
72301
HLT
MRCLK
MRCK
22301
HLT

;CLOCK MR REG
;MR SHOULD
;EQUAL 72301
;MR=BAD GOOD=CORRECT ANS
;CLOCK MR REG
;CHECK MR
;TO EQUAL 22301
;MR=BAD GOOD=CORRECT ANS

;PERFORM 71 DOUBLE MAINT CLOCK OPERATIONS--

013702 012767 000107 165272
013710 104422
013712 104420
013714 073701
013716 104000
013720 104422
013722 104420
013724 023701
013726 104000
013730 005367 165246
013734 001365
013736 104422
013740 104420
013742 073701
013744 104000

MRWCK2: MOV #71.,REPT
MRCLK
MRCK
73701
HLT
MRCLK
MRCK
23701
HLT
DEC REPT
BNE MRWCK2
MRCLK
MRCK
73701
HLT

;CLOCK MR REG
;CHECK MR REG
;TO EQUAL 73701
;MR=BAD GOOD=CORRECT ANS
;CLOCK MR REG
;CHECK MR REG
;TO EQUAL 23701
;MR=BAD GOOD=CORRECT ANS
;DONE YET
;NO LOOP
;CLOCK MR REG
;CHECK MR REG
;TO EQUAL 73701
;MR=BAD GOOD=CORRECT ANS

;READ SYNC"1"

013746	012777	000055	165150	MOV	#55,ARSMR
013754	012777	000045	165142	MOV	#45,ARSMR
013762	104420			MRCK	
013764	023745			23745	
013766	104000			HLT	
013770	012777	000055	165126	MOV	#55,ARSMR
013776	012777	000045	165120	MOV	#45,ARSMR
014004	104420			MRCK	
014006	173745			173745	
014010	104000			HLT	

;CHECK MR FOR
;FOR CORRECT
;ANS IS IN GOOD

;READ DATA

014012	005067	165206		MRWCK3: CLR	WORK3
014016	012705	026716		MOV	#INBUF,R5
014022	162705	000002		SUB	#2,R5
014026	012767	000025	165150	MOV	#21,REPT1
014034	012767	002200	165140	MOV	#1152,REPT

;CLEAR CLOCK COUNT FOR DATA WD
;GET STARTING ADDRESS FOR DATA BUFFER

014042	104444			1\$: RBIT	
014044	104440			CLKR1	
014046	104000			HLT	
014050	104442			CLKR2	
014052	104000			HLT	
014054	005367	165122		DEC	REPT
014060	001370			BNE	1\$
014062	032767	000400	165076	2\$: BIT	#BIT8,ONCE
014070	001030			BNE	3\$
014072	052767	000400	165066	BIS	#BIT8,ONCE
014100	016767	165100	165064	MOV	REPT1,SAVEE
014106	004767	007440		JSR	PC,GENCRC

;SETUP COUNTER FOR 1ST SB BIT
;SETUP COUNTER TO TRANSFER
;128 WORDS-9X128=1152
;2 CLOCKS PER 2 BITS OF DATA
;GET 2 DATA BITS
;CLOCK MR
;MR REG NOT CORRECT
;CLOCK MR REG
;MR REG NOT CORRECT
;DONE WITH DATA BUFFER YET?
;NO
;DID WE ALREADY DO CRC?
;YES
;NO SET CRC FLAG
;SAVE REPT1
;GENERATE CRC WORD
;AND LEAVE IN LOC "WORK"

014112	012702	026716		MOV	#INBUF,R2
014116	016767	165050	165060	MOV	SAVEE,REPT1
014124	062702	000400		ADD	#400,R2
014130	016712	165060		MOV	WORK,AR2
014134	010205			MOV	R2,R5
014136	162705	000002		SUB	#2,R5
014142	012767	000011	165032	MOV	#9,REPT
014150	000734			BR	1\$

;RESTORE REPT1
;STORE CRC WORD TO BE READ
;AT END OF INBUF TABLE

;SETUP TO TRANSFER 1 WD
;TRANSFER CRC WD

D06

014152	104422	3\$:	MRCLK	:CLOCK MR REG
014154	104420		MRCK	:CHECK MR REG
014156	003701		3701	:TO EQUAL
014160	104000		HLT	:3701
014162	104422		MRCLK	:CLOCK MR REG
014164	104420		MRCK	:CHECK MR
014166	153701		153701	:TO EQUAL
014170	104000		HLT	:153701
014172	104422		MRCLK	:CLOCK MR REG
014174	104420		MRCK	:CHECK MR
014176	007701		7701	:TO EQUAL
014200	104000		HLT	:7701
014202	104422		MRCLK	:CLOCK MR REG
014204	104420		MRCK	:CHECK MR
014206	153701		153701	:TO EQUAL
014210	104000		HLT	:153701

:PERFORM 8 DOUBLE MAINTENANCE CLOCK OPERATIONS
:STEP INTO END OF SECTOR DEAD BAND
:EBL IS NOW ASSERTED

014212	012767	000010	164762	MRWCK4: MOV	#8.,REPT	
014220	104422			1\$:	MRCLK	:CLOCK MR REG
014222	104420				MRCK	:CHECK MR REG
014224	003701				3701	:TO EQUAL
014226	104000				HLT	:3601
014230	104422				MRCLK	:CLOCK MR REG
014232	104420				MRCK	:CHECK MR
014234	153701				153701	:REG TO
014236	104000				HLT	:EQUAL 153601
014240	005367	164736			DEC	:DONE YET?
014244	001365				BNE	:NO

:PERFORM ONE MAINTENANCE CLOCK OPERATION
:SHOULD GET STROBE BUFFER

014246	104422			MRCLK	:CLOCK MR REG
014250	104420			MRCK	:CHECK MR
014252	007701			7701	:REG TO
014254	104000			HLT	:EQUAL 7601

:PERFORM ONE MAINTENANCE CLOCK OPERATION
:SHOULD COMPLETE TRANSFER.

014256	104422			MRWCK5: MRCLK	:CLOCK MR REG
014260	022777	004250	164612	CMP	#4250, JRSCS1
014266	001401			BEQ	1\$
014270	104054			HLT	:DA!DS!WC
014272	005777	164606		1\$:	TST
014276	001401			BEQ	:+4
014300	104010			HLT	:WC
014302	022777	000001	164600	CMP	#1, JRSDA
014310	001401			BEQ	:+4
014312	104004			HLT	:DA

:ANY ERRORS?
:NO
:DID WC GO TO 0
:YES
:WC REG SHOULD=0
:DOES RSDA=1
:YES
:RSDA SHOULD=1

E06

014314 104400

:TEST 54 MAINTENANCE MODE CRC TEST 1 (NO DCK ERRORS)

TST54: SCOPE

:MODULES TESTED: M7753)
:THE R504 DISK IS SET UP TO READ (IN MAINTENANCE MODE) ONE
:SECTOR OF A SPECIALLY CREATED DATA PATTERN WHICH LEAVES ONLY
:ONE BIT SET IN THE CRC REGISTER PRIOR TO CHECKING THE CRC
:WORD. THE CORRESPONDING CRC WORD IS THEN "READ", RESULTING
:IN NO DCK ERROR. THE DATA PATTERN IS THEN MODIFIED (BY
:SHIFTING) AND THE ENTIRE READ SEQUENCE REPEATED UNTIL ALL 16
:BITS IN THE CRC REGISTER HAVE BEEN CHECKED.

014316 012767 000040 164620
014324 104414
014326 052767 000040 164632
014334 042767 147716 164624
014342 104430
014344 104420
014346 022701
014350 104424

014352 032767 000020 164564
014360 001023
014362 012767 000001 164602

MRCRC: MOV #40,FLAG2 ;CLEAR TST FLAG
CLRDK ;CLEAR DRIVE REGISTERS
BIS #BIT5,ONCEE ;TYPE CLOCK COUNT IF ERROR OCCURS
BIC #147716,ONCEE ;CLEAR ALL OTHER FLAG BITS
MRIND ;SEND INDEX PULSE TO MR REG
MRCK ;CHECK MR REG
22701 ;TO EQUAL 22701
MRINT ;INIT MAINT MODE (CLEAR MRSP)
;BY SENDING 2 CLOCK PULSES
BIT #BIT4,FLAG2 ;FIRST TIME THROUGH
BNE 3\$;NO
MOV #1,SAVEE ;LOAD 1ST CRC WORD

:FILL MEMORY DATA BUFFER (INBUF) WITH 1 SECTOR. CREATE BUFFER
:WITH 144 WORDS OF 16 BITS WHICH EQUALS THE NO. OF BITS IN 128 18 BITS WORDS
:DATA BUFFER CONTAINS 14 WORDS OF ZEROS
:A WORD OF 12
:A WORD OF 20000
:128 WORDS OF ZEROS

:NOTE:
:IN THIS TEST, ALL 18 BITS OF THE R504 DATA
:WORD MUST BE MANIPULATED. HENCE, A TABLE
:CONTAINING 2304 BITS (128 X 18) IS REQUIRED
:INSTEAD OF A TABLE CONTAINING 128 WORDS.

014370 012702 026716
014374 012703 000016
014400 005022
014402 005303
014404 001375
014406 012722 000012
014412 012722 020000
014416 012703 000200
014422 005022
014424 005303
014426 001375

1\$: MOV #INBUF,R2 ;GET LOCATION OF INBUF
MOV #14,R3 ;SETUP COUNTER
CLR (R2)+ ;TO CLEAR THE
DEC R3 ;FIRST 14
BNE 1\$;WORDS
MOV #12,(R2)+ ;LOAD A 12
MOV #20000,(R2)+ ;LOAD A 20000
2\$: MOV #128,R3 ;SETUP COUNTER
CLR (R2)+ ;TO CLEAR THE
DEC R3 ;REMAINING WORDS
BNE 2\$;FOR THAT SECTOR

;SETUP CONTROLLER TO TRANSFER 128 WORDS OF DATA (1 SECTOR) FROM SECTOR 0

014430	012777	027516	164450	3\$:	MOV	#OUTBUF, ARSBA	:LOAD BUS ADDR REG
014436	012777	177600	164440		MOV	#177600, ARSWC	:LOAD WORD COUNT REG
014444	012777	000071	164426		MOV	#71, ARSCS1	:LOAD READ COMMAND
014452	012702	000200			MOV	#200, R2	
014456	012703	027516			MOV	#OUTBUF, R3	
014462	052767	000020	164454		BIS	#BIT4, FLAG2	:SET 1ST TIME THROUGH FLAG
014470	005023			4\$:	CLR	(R3)+	
014472	005302				DEC	R2	
014474	001375				BNE	4\$	
014476	104446				GETSP		:CLOCK ROUTINE TO GET SECTOR PULSE
							:TO CLEAR OUT COUNTERS AND REGISTERS
							:THAT OTHERWISE COULD NOT BE CLEARED.
							:COULD NOT SET SECTOR PULSE (0)
							:CLOCK MR REG SP = 1

014500 104220
014502 104450

HLT !MR
SPASS

;ASSERT INDEX PULSE TO INITIALIZE THE DRIVE

014504 104430
014506 104420
014510 022601
014512 104000

MRIND
MRCK
22601
HLT

:CHECK MR REG TO EQUAL
:22601 FOR A
:READ COMD

;STEP THRU RESYNC PERIOD

014514 012767 001000 164460
014522 052767 000040 164436
014530 104422
014532 104420
014534 072601
014536 104000
014540 104422
014542 104420
014544 022601
014546 104000
014550 005367 164426
014554 001365

MRCRC1: MOV #512, REPT
BIS #BIT5, ONCEE
MRCLK
MRCK
72601
HLT
MRCLK
MRCK
22601
HLT
DEC REPT
BNE MRCRC1

:TYPE OUT CLOCK COUNT IF ERROR OCCURS
:CLOCK MR REG
:CHECK FOR
:CORRECT DATA
:MR=BAD GOOD=CORRECT DATA
:CLOCK MR REG
:CHECK FOR
:CORRECT DATA
:ERROR WHILE CLOCKING THROUGH RESYNC
:FINISH LOOPING
:THROUGH RESYNC PERIOD

;ONE MORE CLOCK PULSE SHOULD ASSERT SECTOR PULSE
;SP=0 EQUALS SECTOR PULSE

014556 104422
014560 104420
014562 072201
014564 104000
014566 104422
014570 104420
014572 022201
014574 104000

MRCLK
MRCK
72201
HLT
MRCLK
MRCK
22201
HLT

:CLOCK MR REG
:MR SHOULD
:EQUAL 72201
:MR=BAD GOOD=CORRECT ANS
:CLOCK MR REG
:CHECK MR
:TO EQUAL 22201
:MR=BAD GOOD=CORRECT ANS

;PERFORM 71 DOUBLE MAINT CLOCK OPERATIONS--

014576	012767	000107	164376	MOV	#71.,REPT	
014604	104422			MRCRC2: MRCLK		:CLOCK MR REG
014606	104420			MRCK		:CHECK MR REG
014610	073601			73601		:TO EQUAL 73601
014612	104000			HLT		:MR=BAD GOOD=CORRECT ANS
014614	104422			MRCLK		:CLOCK MR REG
014616	104420			MRCK		:CHECK MR REG
014620	023601			23601		:TO EQUAL 23601
014622	104000			HLT		:MR=BAD GOOD=CORRECT ANS
014624	005367	164352		DEC	REPT	:DONE YET
014630	001365			BNE	MRCRC2	:NO LOOP
014632	104422			MRCLK		:CLOCK MR REG
014634	104420			MRCK		:CHECK MR REG
014636	073601			73601		:TO EQUAL 73601
014640	104000			HLT		:MR=BAD GOOD=CORRECT ANS

;READ SYNC"1"

014642	012777	000055	164254	MOV	#55, JRSMR	
014650	012777	000045	164246	MOV	#45, JRSMR	
014656	104420			MRCK		:CHECK MR REG
014660	023645			23645		:FOR CORRECT
014662	104000			HLT		:ANS IS IN GOOD
014664	012777	000055	164232	MOV	#55, JRSMR	
014672	012777	000045	164224	MOV	#45, JRSMR	
014700	104420			MRCK		
014702	173645			173645		
014704	104000			HLT		

;READ DATA

014706	005067	164312		MRCRC3: CLR	WORK3	:CLEAR CLOCK COUNT FOR DATA WD
014712	012705	026716		MOV	#INBUF, R5	:GET STARTING ADDRESS FOR DATA BUFFER
014716	162705	000002		SUB	#2, R5	
014722	012767	000025	164254	MOV	#21, REPT1	:SETUP COUNTER FOR 1ST SB BIT
014730	012767	002200	164244	MOV	#1152., REPT	:SETUP COUNTER TO TRANSFER
						:128 WORDS-9X128=1152
						:2 CLOCKS PER 2 BITS OF DATA
014736	104444			15: RBIT		:GET 2 DATA BITS
014740	104440			CLKR1		:CLOCK MR
014742	104000			HLT		:MR REG NOT CORRECT
014744	104442			CLKR2		:CLOCK MR REG
014746	104000			HLT		:MR REG NOT CORRECT
014750	005367	164226		DEC	REPT	:DONE WITH DATA BUFFER YET?
014754	001370			BNE	15	:NO

014756	032767	000400	164202	2\$:	BIT	#BIT8,ONCEE	:DID WE ALREADY DO CRC?
014764	001020				BNE	3\$	:YES
014766	052767	000400	164172		BIS	#BIT8,ONCEE	:NO SET CRC FLAG
014774	012702	026716			MOV	#INBUF,R2	:MOVE CRC
015000	062702	000440			ADD	#440,R2	:WORD TO END OF
015004	016712	164162		4\$:	MOV	SAVEE,R2	:INBUF TABLE
015010	010205			5\$:	MOV	R2,R5	:GET CRC WORD
015012	162705	000002			SUB	#2,R5	
015016	012767	000011	164156		MOV	#9,REPT	:SETUP TO TRANSFER 1 WD
015024	000744				BR	1\$	:TRANSFER CRC WD
015026	104422			3\$:	MRCLK		:CLOCK MR REG
015030	104420				MRCK		:CHECK MR REG
015032	003601				3601		:TO EQUAL
015034	104000				HLT		:3601
015036	104422				MRCLK		:CLOCK MR REG
015040	104420				MRCK		:CHECK MR
015042	153601				153601		:TO EQUAL
015044	104000				HLT		:153601
015046	104422				MRCLK		:CLOCK MR REG
015050	104420				MRCK		:CHECK MR
015052	007601				7601		:TO EQUAL
015054	104000				HLT		:7601
015056	104422				MRCLK		:CLOCK MR REG
015060	104420				MRCK		:CHECK MR
015062	153601				153601		:TO EQUAL
015064	104000				HLT		:153601

:PERFORM 8 DOUBLE MAINTENANCE CLOCK OPERATIONS
:STEP INTO END OF SECTOR DEAD BAND
:EBL IS NOW ASSERTED

015066	012767	000010	164106	MRCRC4: MOV	#8.,REPT	
015074	104422			1\$: MRCLK		:CLOCK MR REG
015076	104420			MRCK		:CHECK MR REG
015100	003601			3601		:TO EQUAL
015102	104000			HLT		:3601
015104	104422			MRCLK		:CLOCK MR REG
015106	104420			MRCK		:CHECK MR
015110	153601			153601		:REG TO
015112	104000			HLT		:EQUAL 153601
015114	005367	164062		DEC	REPT	:DONE YET?
015120	001365			BNE	1\$	:NO

:PERFORM ONE MAINTENANCE CLOCK OPERATION
:SHOULD GET STROBE BUFFER

015122	104422			MRCLK		:CLOCK MR REG
015124	104420			MRCK		:CHECK MR
015126	007601			7601		:REG TO
015130	104000			HLT		:EQUAL 7601

:PERFORM ONE MAINTENANCE CLOCK OPERATION
:SHOULD COMPLETE TRANSFER.

015132	104422			MRCRC5: MRCLK		:CLOCK MR REG
015134	022777	004270	163736	CMP	#4270,DRSCS1	:ANY ERRORS?
015142	001401			BEQ	1\$	:NO
015144	104054			HLT	!DA!DS!WC	
015146	005777	163732		1\$: TST	DRSWC	:DID WC GO TO 0
015152	001401			BEQ	+4	:YES
015154	104010			HLT	!WC	:WC REG SHOULD=0
015156	006167	164010		ROL	SAVEE	:GET NEXT CRC WORD
015162	103404			BCS	2\$	:DONE - BRANCH
015164	004767	010506		JSR	PC,MDATA	:SHIFT DATA PATTERN
015170	000167	177130		JMP	MRCRC	:RESTART TEST WITH NEW DATA PATTERN
015174				2\$:		:DONE

015174 104400

015176 012767 000040 163740
015204 104414
015206 052767 000040 163752
015214 042767 147716 163744
015222 104430
015224 104420
015226 022701
015230 104424

015232 032767 000020 163704
015240 001023
015242 012767 000001 163722

015250 012702 026716
015254 012703 000017
015260 005022
015262 005303
015264 001375
015266 012722 000001
015272 012722 042000
015276 012703 000177
015302 005022
015304 005303
015306 001375

015310 012777 027516 163570
015316 012777 177600 163560
015324 012777 000071 163546
015332 012702 000200
015336 012703 027516
015342 052767 000020 163574
015350 005023
015352 005302
015354 001375
015356 104446

015360 104220
015362 104450

```

*****
:TEST 55                                MAINTENANCE MODE CRC TEST 2 (CAUSE DCK ERRORS)
*****
TST55: SCOPE
:MODULE TESTED M7753
:THIS TEST IS SIMILAR TO CRC TEST 1 EXCEPT THAT THE DATA
:PATTERN HAS BEEN MODIFIED TO LEAVE A SINGLE BIT SET IN THE
:CRC REGISTER AFTER BOTH DATA AND CRC WORDS HAVE BEEN "READ".
:THIS CAUSES A DCK ERROR. THE READ SEQUENCE IS REPEATED 16
:TIMES TO TEST THAT EACH BIT IN THE CRC REGISTER CAN CAUSE A DCK ERROR.
MRDCK: MOV #40,FLAG2          :CLEAR TST FLAG
        CLRDK                :CLEAR DRIVE REGISTERS
        BIS #BITS,ONCE       :TYPE CLOCK COUNT IF ERROR OCCURS
        BIC #147716,ONCE     :CLEAR ALL OTHER FLAG BITS
        MRIND                :SEND INDEX PULSE TO MR REG
        MRCK                 :CHECK MR REG
        22701                :TO EQUAL 22701
        MRINT                :INIT MAINT MODE (CLEAR MRSP)
                                :BY SENDING 2 CLOCK PULSES
                                :FIRST TIME THROUGH
                                :NO
                                :LOAD 1ST CRC WORD
                                :FILL MEMORY DATA BUFFER (INBUF) WITH 128 WORDS (1 SECTOR) CREATE BUFFER
                                :WITH 144 WORDS OF 16 BITS WHICH = THE NO. OF BITS IN 128 18 BIT WORDS
                                :DATA BUFFER CONTAINS 15 WORDS OF ZEROS
                                :A WORD OF 1
                                :A WORD OF 42000
                                :127 WORDS OF ZEROS
                                :GET LOCATION OF OUTBUF
                                :SETUP COUNTER
                                :TO CLEAR THE
                                :FIRST 15
                                :WORDS
                                :LOAD A 1
                                :LOAD A 42000
                                :SETUP COUNTER
                                :TO CLEAR THE
                                :REMAINING WORDS
                                :FOR THAT SECTOR
                                :SETUP CONTROLLER TO TRANSFER 128 WORDS OF DATA (1 SECTOR) FROM SECTOR 0
3$: MOV #OUTBUF,ARSBA        :LOAD BUS ADDR REG
    MOV #177600,ARSWC        :LOAD WORD COUNT REG
    MOV #71,ARSC51           :LOAD READ COMMAND
    MOV #200,R2
    MOV #OUTBUF,R3
    BIS #BIT4,FLAG2          :SET 1ST TIME THROUGH FLAG
    CLR (R3)+
    DEC R2
    BNE 4$
    GETSP
                                :CLOCK ROUTINE TO GET SECTOR PULSE
                                :TO CLEAR OUT COUNTERS AND REGISTERS
                                :THAT OTHERWISE COULD NOT BE CLEARED.
                                :COULD NOT SET SECTOR PULSE (0)
                                :CLOCK MR REG SP = 1
4$: CLR (R3)+
    DEC R2
    BNE 4$
    GETSP
                                :CLOCK ROUTINE TO GET SECTOR PULSE
                                :TO CLEAR OUT COUNTERS AND REGISTERS
                                :THAT OTHERWISE COULD NOT BE CLEARED.
                                :COULD NOT SET SECTOR PULSE (0)
                                :CLOCK MR REG SP = 1
5$: HLT !MR
    SPASS

```

K06

ZZ-CERSD-D
CERSDD.P11RH70-R504 MAINTENANCE MODE DIAGNOSTIC
23-JAN-78 13:36MACY11 30A(1052) 23-JAN-78 13:38 PAGE 115
TST55 MAINTENANCE MODE CRC TEST 2 (CAUSE DCK ERRORS)

SEQ 0075

015364 104430
015366 104420
015370 022601
015372 104000

;ASSERT INDEX PULSE TO INITIALIZE THE DRIVE

MRIND
MRCK
22601
HLT;CHECK MR REG TO EQUAL
;22601 FOR A
;READ COMD

;STEP THRU RESYNC PERIOD

015374 012767 001000 163600
015402 052767 000040 163556
015410 104422
015412 104420
015414 072601
015416 104000
015420 104422
015422 104420
015424 022601
015426 104000
015430 005367 163546
015434 001365MRDCK1: MOV #512, REPT
BIS #BITS, ONCEE
MRCLK
MRCK
72601
HLT
MRCLK
MRCK
22601
HLT
DEC REPT
BNE MRDCK1;TYPE OUT CLOCK COUNT IF ERROR OCCURS
;CLOCK MR REG
;CHECK FOR
;CORRECT DATA
;MR=BAD GOOD=CORRECT DATA
;CLOCK MR REG
;CHECK FOR
;CORRECT DATA
;ERROR WHILE CLOCKING THROUGH RESYNC
;FINISH LOOPING
;THROUGH RESYNC PERIOD;ONE MORE CLOCK PULSE SHOULD ASSERT SECTOR PULSE
;SP=0 EQUALS SECTOR PULSE015436 104422
015440 104420
015442 072201
015444 104000
015446 104422
015450 104420
015452 022201
015454 104000MRCLK
MRCK
72201
HLT
MRCLK
MRCK
22201
HLT;CLOCK MR REG
;MR SHOULD
;EQUAL 72201
;MR=BAD GOOD=CORRECT ANS
;CLOCK MR REG
;CHECK MR
;TO EQUAL 22201
;MR=BAD GOOD=CORRECT ANS

;PERFORM 71 DOUBLE MAINT CLOCK OPERATIONS--

015456	012767	000107	163516	MOV	#71.,REPT	
015464	104422			MRDCK2:		:CLOCK MR REG
015466	104420			MRCK		:CHECK MR REG
015470	073601			73601		:TO EQUAL 73601
015472	104000			HLT		:MR=BAD GOOD=CORRECT ANS
015474	104422			MRCLK		:CLOCK MR REG
015476	104420			MRCK		:CHECK MR REG
015500	023601			23601		:TO EQUAL 23601
015502	104000			HLT		:MR=BAD GOOD=CORRECT ANS
015504	005367	163472		DEC	REPT	:DONE YET
015510	001365			BNE	MRDCK2	:NO LOOP
015512	104422			MRCLK		:CLOCK MR REG
015514	104420			MRCK		:CHECK MR REG
015516	073601			73601		:TO EQUAL 73601
015520	104000			HLT		:MR=BAD GOOD=CORRECT ANS
;READ SYNC"1"						
015522	012777	000055	163374	MOV	#55,RSMR	
015530	012777	000045	163366	MOV	#45,RSMR	
015536	104420			MRCK		:CHECK MR REG
015540	023645			23645		:TO EQUAL
015542	104000			HLT		:CORRECT ANS IN GOOD
015544	012777	000055	163352	MOV	#55,RSMR	
015552	012777	000045	163344	MOV	#45,RSMR	
015560	104420			MRCK		
015562	173645			173645		
015564	104000			HLT		
;READ DATA						
015566	005067	163432		MRDCK3:	CLR	WORK3
015572	012705	026716		MOV	#INBUF,RS	:CLEAR CLOCK COUNT FOR DATA WD
015576	162705	000002		SUB	#2,RS	:GET STARTING ADDRESS FOR DATA BUFFER
015602	012767	000025	163374	MOV	#21.,REPT1	:SETUP COUNTER FOR 1ST SB BIT
015610	012767	002200	163364	MOV	#1152.,REPT	:SETUP COUNTER TO TRANSFER
;128 WORDS-9X128=1152						
;2 CLOCKS PER 2 BITS OF DATA						
015616	104444			1\$:	RBIT	:GET 2 DATA BITS
015620	104440			CLKR1		:CLOCK MR
015622	104000			HLT		:MR REG NOT CORRECT
015624	104442			CLKR2		:CLOCK MR REG
015626	104000			HLT		:MR REG NOT CORRECT
015630	005367	163346		DEC	REPT	:DONE WITH DATA BUFFER YET?
015634	001370			BNE	1\$	:NO

015636	032767	000400	163322	2\$:	BIT	#BIT8,ONCEE	:DID WE ALREADY DO CRC?
015644	001020				BNE	3\$	:YES
015646	052767	000400	163312		BIS	#BIT8,ONCEE	:NO SET CRC FLAG
015654	012702	026716			MOV	#INBUF,R2	:MOVE CRC
015660	062702	000440			ADD	#440,R2	:WORD TO END OF
015664	012712	000000		4\$:	MOV	#0,R2	:INBUF TABLE
015670	010205			5\$:	MOV	R2,R5	:GET CRC WORD
015672	162705	000002			SUB	#2,R5	
015676	012767	000011	163276		MOV	#9.,REPT	:SETUP TO TRANSFER 1 WD
015704	000744				BR	1\$	:TRANSFER CRC WD
015706	104422			3\$:	MRCLK		:CLOCK MR REG
015710	104420				MRCK		:CHECK MR REG
015712	003601				3601		:TO EQUAL
015714	104000				HLT		:3601
015716	104422				MRCLK		:CLOCK MR REG
015720	104420				MRCK		:CHECK MR
015722	153601				153601		:TO EQUAL
015724	104000				HLT		:153601
015726	104422				MRCLK		:CLOCK MR REG
015730	104420				MRCK		:CHECK MR
015732	007601				7601		:TO EQUAL
015734	104000				HLT		:7601
015736	104422				MRCLK		:CLOCK MR REG
015740	104420				MRCK		:CHECK MR
015742	153601				153601		:TO EQUAL
015744	104000				HLT		:153601

;PERFORM 8 DOUBLE MAINTENANCE CLOCK OPERATIONS
;STEP INTO END OF SECTOR DEAD BAND
;EBL IS NOW ASSERTED

015746	012767	000010	163226	MRDCK4: MOV	#8.,REPT	
015754	104422			1\$: MRCLK		;CLOCK MR REG
015756	104420			MRCK		;CHECK MR REG
015760	003601			3601		;TO EQUAL
015762	104000			HLT		;3601
015764	104422			MRCLK		;CLOCK MR REG
015766	104420			MRCK		;CHECK MR
015770	153601			153601		;REG TO
015772	104000			HLT		;EQUAL 153601
015774	005367	163202		DEC	REPT	;DONE YET?
016000	001365			BNE	1\$	;NO

;PERFORM ONE MAINTENANCE CLOCK OPERATION
;SHOULD GET STROBE BUFFER

016002	104422			MRCLK		;CLOCK MR REG
016004	104420			MRCK		;CHECK MR
016006	007601			7601		;REG TO
016010	104000			HLT		;EQUAL 7601

;PERFORM ONE MAINTENANCE CLOCK OPERATION
;SHOULD COMPLETE TRANSFER.

016012	104422			MRDCK5: MRCLK		;CLOCK MR REG
016014	022777	144270	163056	CMP	#144270, @RSCS1	;IS RSCS1 CORRECT?
016022	001401			BEQ	1\$	;YES
016024	104054			HLT	!DA!DS!WC	
016026	005777	163052		1\$: TST	@RSWC	;DID WC GO TO 0
016032	001401			BEQ	+4	;YES
016034	104010			HLT	!WC	;WC REG SHOULD=0
016036	022777	100000	163050	CMP	#100000, @RSER	;DID DCK SET?
016044	001417			BEQ	3\$	;YES
016046	104050			HLT	!DS!WC	
016050	104402	016054		TYPE	+2	;ASCIZ <15><12>"DCK DID NOT SET "
016100	004767	004210		JSR	PC, CRCTYP	;GET IC THAT FAILED AND TYPE IT
016104	000241			3\$: CLC		
016106	006167	163060		ROL	SAVEE	;GET NEXT CRC WORD
016112	103404			BCS	2\$	;DONE - BRANCH
016114	004767	007556		JSR	PC, MDATA	;SHIFT DATA PATTERN
016120	000167	177060		JMP	MRDCK	;RESTART TEST WITH NEW DATA PATTERN
016124				2\$:		;DONE

B07

016124 104400

:TEST 56 IGNORE FUNCTION TEST

TST56: SCOPE

:MODULE TESTED: M7759, M7770
:PUT THE DISK MAINTENANCE MODE AND SET ERROR CONDITIONS IN
:THE DRIVE ERROR REGISTER (RSE). TRY TO START A READ
:TRANSFER. THE "GO" BIT IN RSCS1 SHOULD NOT SET. MISSED
:TRANSFER ERROR (MXF) SHOULD SET IN RSCS2 WHICH IS TURN SHOULD
:CAUSE "TRE" AND "SC" TO SET IN RSCS1.

016126	104414				MRIFT: CLRDK	:CLEAR ALL REGISTERS
016130	104430				MRIND	:SEND INDEX PULSE TO MR REG
016132	104420				MRCK	:CHECK MR REG
016134	022701				22701	:TO EQUAL 22701
016136	104424				MRINT	:INIT MAINT MODE (CLEAR MRSP)
016140	012777	177777	162746		MOV	:SET ERRORS
016146	016777	163010	162742		MOV	:CLEAR ATA BIT IN RSAS
					#-1,RSER	:AND ERROR BITS IN RSCS1
					UNITSV,RSAS	:LOAD RSBA
016154	012777	027516	162724		MOV	:LOAD RSWC
016162	012777	177777	162714		MOV	:LOAD READ FUNCTION
016170	012777	000071	162702		MOV	:IS "GO" BIT ZERO?
016176	032777	000001	162674		BIT	:YES
016204	001401				BEQ	: "GO" BIT IN RSCS1 SHOULD NOT
016206	104140				HLT	:LOAD IF ERRORS ARE PRESENT IN THE DRIVE
					!DS!AS	:SETUP TIMEOUT FOR MXF ERROR
016210	012767	177777	162776	1\$:	MOV	
016216	005367	162772		5\$:	DEC	
016222	000240				NOP	
016224	000240				NOP	
016226	001373				BNE	5\$
016230	017700	162646			MOV	:CHECK RSCS2 FOR MXF
016234	012701	001100			MOV	:GET CORRECT ANS
016240	056701	162714			BIS	:FOR RSCS2
016244	020001				CMP	:IS RSCS2 CORRECT
016246	001401				BEQ	:YES
016250	104000				HLT	:BAD=RSCS2 GOOD=CORRECT ANS
						:MXF SHOULD BE SET IN RSCS2
						:FOR A READ WAS ISSUED
016252	022777	144270	162620	2\$:	CMP	:WITH ERROR BITS SET IN RSE.
016260	001401				BEQ	:IS RSCS1 CORRECT?
016262	104042				HLT	:YES
					!DS!ER	:SC AND TRE SHOULD BE SET FOR
						:MXF SHOULD BE SET IN RSCS2

016264 022777 177777 162612 35:
016272 001401
016274 104010

CMP #-1,RSWC
BEQ .+4
HLT !WC

:DID RSWC CHANGE?
:NO
:WC SHOULD NOT HAVE CHANGED
:FOR READ SHOULD HAVE NEVER
:BEEN EXECUTED

016276 022777 027516 162602
016304 001405
016306 017700 162574
016312 012701 027516
016316 104000

CMP #OUTBUF,RSBA
BEQ 45
MOV RSBA,BAD
MOV #OUTBUF,GOOD
HLT

:DID RSBA MOVE
:NO
:GET RSBA
:GET CORRECT ANS
:RSBA=BAD GOOD=CORRECT ANS
:RSBA MOVED WHEN THE READ
:COND WAS LOADED WITH ERROR
:BITS SET IN RSER, READ CMD SHOULD
:NEVER HAVE BEEN EXECUTED AND RSBA
:SHOULD NOT HAVE MOVED.
:CLEAR ALL REGISTERS

016320 104414
016322 017700 162554
016326 012701 000100
016332 056701 162622
016336 020100
016340 001401
016342 104000

45:

CLDK
MOV RSCS2,BAD
MOV #100,GOOD
BIS UNNUM,GOOD
CMP GOOD,BAD
BEQ .+4
HLT

:GET RSCS2
:SETUP CORRECT
:ANS
:IS CS2 CORRECT?
:YES
:MXF SHOULD HAVE CLEARED IN
:RSCS2 WITH THE "CLR" BIT BEING
:SET IN RSCS2.

D07

016344 104400

:TEST 57 INVALID ADDRESS ERROR (IAE) TEST

↑TST57: SCOPE:MODULE TESTED M7754, M7770
:FLOAT A 1 THROUGH THE FOUR SPARE ADDRESS BITS IN THE DISK
:ADDRESS REGISTER (RSDA). THIS SHOULD CAUSE "IAE" TO SET IN
:THE ERROR REGISTER (RSE) WHEN A READ FUNCTION IS LOADED INTO
:RSCS1 WHICH IN TURN SHOULD CAUSE ATTENTION TO SET IN THE
:DRIVE STATUS REGISTER (RSDS) AND "TRE" AND "SC" TO SET IN THE
:CONTROL REGISTER (RSCS1).

016346	042767	000040	162612		BIC	#BIT5,ONCEE	:CLEAR CLK CNT FLAG
016354	012702	004000			MOV	#4000,R2	:LOAD R2 WITH INVALID ADDR
016360	012767	016366	162422		MOV	#4\$,LAD	:LOOP TO HERE ON ERROR
016366	104416			4\$:	MRODM		:PUT DRIVE IN MAINT MODE
016370	104420				MRCK		:CHECK MAINT REG
016372	022701				22701		
016374	104424				MRINT		:INIT MAINT MODE (CLEAR MRSP)
016376	032767	000004	162562		BIT	#BIT2,ONCEE	:LOOPING ON ERRORS)
016404	001002				1\$		:YES
016406	006102				BNE		:GET INVALID ADDRESS
016410	103454				ROL	R2	:DONE FLOATING A ONE YET?
016412	010277	162472		1\$:	BCS	IADONE	:LOAD RSDA WITH INVALID ADDRESS
016416	012777	000071	162454		MOV	R2,RSDA	:DO A READ TO INVALID ADDR
016424	022777	002000	162462		MOV	#71,RSCS1	:IS RSE CORRECT?
016432	001404				CMP	#2000,RSE	:YES
016434	052767	000004	162524		BEQ	2\$	:SET ERROR BIT
016442	104044				BIS	#BIT2,ONCEE	:RSE SHOULD=2000 FOR
					HLT	!DS!DA	:A READ COMMAND WAS GIVEN
							:TO AN ILLEGAL ADDRESS
016444	042767	000004	162514	2\$:	BIC	#BIT2,ONCEE	:CLEAR ERROR FLAG
016452	022777	150600	162432		CMP	#150600,RSDS	:DID IAE SET?
016460	001404				BEQ	3\$	:YES
016462	052767	000004	162476		BIS	#BIT2,ONCEE	:SET ERROR BIT
016470	104044				HLT	!DS!DA	:RSDS SHOULD=150600 FOR
							:IAE SHOULD BE SET IN RSE
016472	042767	000004	162466	3\$:	BIC	#BIT2,ONCEE	:CLEAR ERROR FLAG
016500	032777	100000	162372		BIT	#BIT15,RSCS1	:DID SC SET?
016506	001004				BNE	5\$	:YES
016510	052767	000004	162450		BIS	#BIT2,ONCEE	:SET ERROR BIT
016516	104044				HLT	!DA!DS	:SC SHOULD BE SET IN RSCS1
							:FOR IAE SHOULD BE SET IN RSE
016520	042767	000004	162440	5\$:	BIC	#BIT2,ONCEE	:CLEAR ERROR BIT
016526	104414				CLDRK		:CLEAR ALL ERRORS
016530	005777	162360			TST	RSE	:DID IAE CLEAR?
016534	001401				BEQ	+4	:YES
016536	104040				HLT	!DS	:IAE DID NOT CLEAR
016540	000712				BR	4\$	:CONTINUE
016542				IADONE:			:DONE

E07

016542 104400

:*****
:TEST 60 OPERATION INCOMPLETE ERROR TEST
:*****
TST60: SCOPE:MODULE TESTED M7770
:PUT THE DISK IN MAINTENANCE MODE AND START A READ COMMAND
:THEN ISSUE THREE DISK "INDEX" PULSES TO SIMULATE A COMPLETE
:ROTATION OF THE DISK SURFACE. THE THIRD INDEX PULSE SHOULD
:CAUSE OPERATION IN COMPLETE "OPI" TO APPEAR IN THE DRIVE ERROR
:REGISTER (RSER) AND "ATA" AND "ERR" IN THE DRIVE STATUS REGISTER (RSDS)016544 104414
016546 013777 027516 162332
016554 012777 177777 162322MROPI: CLRDK :CLEAR ALL DRIVE REGISTERS
MOV @#OUTBUF, @RSBA :SETUP RSBA
MOV #-1, @RSWC :SETUP RSWC016562 104430
016564 104420
016566 022701
016570 104424MRIND :SEND INDEX PULSE TO MR REG
MRCK :CHECK MAINT REG
22701 :TO EQUAL 22701
MRINT :INIT MAINT MODE (CLEAR MRSP)

016572 012777 000071 162300

MOV #71, @RSCS1 :LOAD A READ COMMAND

016600 104430
016602 104430
016604 104430MRIND :ISSUE THREE INDEX
MRIND :PULSES TO
MRIND :CAUSE OPI016606 017700 162302
016612 012701 020000
016616 020100
016620 001434
016622 104402 016626
016710 104000

;NOW CHECK FOR CORRECT ERRORS IN RSER AND RSDS

MOV @RSER, BAD :GET RSER
MOV #20000, GOOD :GET CORRECT ANS
CMP GOOD, BAD :DID OPI SET IN RSER?
BEQ 1\$:YES
TYPE ., +2 :ASCIZ <15><12>"OPI IN RSER SHOULD SET-3 INDEX PULSES W
HLT :RSER=BAD GOOD=CORRECT ANS016712 022777 150600 162172 1\$:
016720 001401
016722 104040CMP #150600, @RSDS :DID CORRECT ERRORS SET?
BEQ 2\$:YES
HLT !DS :RSDS SHOULD=150600 BECAUSE016724 022777 144270 162146 2\$:
016732 001401
016734 104050CMP #144270, @RSCS1 :DID SC AND TRE SET IN RSCS1?
BEQ MROPIA :YES
HLT !DS!WC :SC AND TRE SHOULD SET IN RSCS1016736 104414
016740 005777 162150
016744 001437
016746 104402 016752
017042 104040
017044 022777 010600 162040 1\$:MROPIA: CLRDK
TST @RSER
BEQ 1\$
TYPE ., +2
HLT !DS
CMP #10600, @RSDS:BECAUSE OF ERROR IN RSER
:CLEAR ALL ERRORS
:DID OPI CLEAR IN RSER
:YES
:ASCIZ <15><12>"OPI IN RSER DID NOT CLEAR BY SETTING CL
:RSER SHOULD=0
:DID ERROR BITS CLEAR IN RSDS
:BY SETTING CLR BIT IN RSCS2
:YES
:RSDS SHOULD=10600017052 001401
017054 104040BEQ .+4
HLT !DS

F07

017056 104400

:TEST 61 PARITY ERROR TEST

TST61: SCOPE

:MODULES TESTED: M7754, M7770
:SET "PAT" BIT IN RSCS2. WRITE A DRIVE REGISTER. "PAR" SHOULD SET IN
:THE DRIVE ERROR REGISTER (RSE) WHICH SHOULD CAUSE "ATA" TO SET IN RSAS
:AND "SC" TO SET IN RSCS1.

017060	104414			MRPAR: CLRDK		:CLEAR ALL REGISTERS
017062	042767	000040	162076	BIC	#BITS,ONCE	:CLEAR CLK CNT FLAG
017070	104430			MRIND		:SEND INDEX PULSE TO MR REG
017072	104420			MRCK		:CHECK MAINT TO
017074	022701			22701		:EQUAL 22701
017076	104424			MRINT		:INIT MAINT MODE (CLEAR I...SP)
017100	052777	000020	161774	BIS	#BIT4,RSCS2	:SET THE "PAT" BIT
017106	012777	000077	161774	MOV	#77,RSDA	:BY WRITING INTO THIS REGISTER,
						:PAR SHOULD SET IN RSE
017114	022777	000010	161772	CMP	#10,RSE	:DID PAR SET?
017122	001401			BEQ	:+4	:YES
017124	104040			HLT	:DS	: "PAR" IN RSE SHOULD BE SET FOR
						:THE "PAT" BIT WAS SET IN RSCS2
017126	022777	104200	161744	CMP	#104200,RSCS1	:WHEN PROGRAM TRIED TO WRITE INTO RSDA
017134	001401			BEQ	:+4	:DID PAR CAUSE SC TO SET?
017136	104044			HLT	:DS!DA	:YES
						:SC SHOULD BE SET IN RSCS1 FOR
017140	022777	000077	161742	CMP	#77,RSDA	:PAR SHOULD BE SET IN RSE
017146	001401			BEQ	:+4	:DID RSDA GET LOADED?
017150	104004			HLT	:DA	:YES
						:RSDA SHOULD=77 FOR PAT
017152	104414			CLRDK		:BIT WAS SET WHEN PROGRAM
017154	022777	004200	161716	CMP	#4200,RSCS1	:TRIED TO WRITE INTO RSDA
017162	001401			BEQ	:+4	:CLEAR ALL ERRORS
017164	104044			HLT	:DS!DA	:DID ERRORS CLEAR?
						:YES
017166	005777	161722		TST	RSE	:SC DID NOT CLEAR BY USING
017172	001401			BEQ	:+4	:THE "CLR" BIT IN RSCS2
017174	104044			HLT	:DS!DA	:DID PAR CLEAR?
						:YES
						:PAR DID NOT CLEAR BY USING
						:THE CLR BIT IN RSCS2

G07

017176 104400

:TEST 62 MAINTENANCE MODE INTERRUPT TEST

†ST62: SCOPE

:MODULE TESTED M7771
:IN THIS TEST THE INTERRUPT ENABLE BIT IS SET (I.E.).
:A TWO SECTOR WRITE COMMAND IS GIVEN. AN "RMR"
:ERROR IS CREATED WHILE THE FIRST SECTOR IS BEING WRITTEN
:THIS SHOULD CAUSE THE DRIVE TO INTERRUPT AFTER THE FIRST
:SECTOR IS WRITTEN AND THE TRANSFER TO TERMINATE.

017200 012767 000002 161736
017206 104414
017210 012737 000200 177776
017216 012706 000500
017222 052767 000040 161736
017230 042767 000600 161730
017236 104430
017240 104420
017242 022701
017244 104424

MREX: MOV #2,FLAG2 ;CLEAR DRIVE REGISTERS
CLRDK ;SETUP FOR INTERRUPT
MOV #200,0#PS
MOV #500,SP
BIS #BITS,ONCE ;SET TYPE CLOCK COUNT FLAG
BIC #600,ONCE ;CLEAR FLAG BITS
MRIND ;SEND INDEX PULSE TO MR REG
MRCK ;CHECK MR REG
22701 ;TO EQUAL 22701
MRINT ;INIT MAINT MODE (CLEAR MRSP)
;BY SENDING 2 CLOCK PULSES

:FILL MEMORY DATA BUFFER (INBUF) WITH 128 WORDS (1 SECTOR)
:DATA BUFFER WORDS ARE :A WORD OF ALL 0'S - ALL 1'S
: :FLOATING 1'S PATTERN (16 WORDS)
: :A PATTERN OF 146314 (110 WORDS)

017246 012702 026716
017252 005022
017254 012722 177777
017260 005003
017262 000261
017264 006103
017266 103402
017270 010322
017272 000774
017274 012703 000156
017300 012704 146314
017304 010422
017306 005303
017310 001375

: :MOV #INBUF,R2 ;GET LOCATION OF OUTBUF
: :CLR (R2)+ ;CLEAR 1ST LOCATION
: :MOV #-1,(R2)+ ;2ND WORD OF ALL ONES
: :CLR R3 ;CLEAR WORK LOC TO GENERATE
: :SEC ;A PATTERN OF FLOATING ONES
1\$: ROL R3 ;GET PATTERN
BCS 2\$;DONE GET OUT
MOV R3,(R2)+ ;FILL BUFFER
BR 1\$;CONT
2\$: MOV #110,R3 ;FILL REMAINING PORTION OF
MOV #146314,R4 ;BUFFER WITH A PATTERN OF 146314
3\$: MOV R4,(R2)+ ;LOAD BUFFER
DEC R3 ;DONE YET?
BNE 3\$;NO

017312 012777 020120 161610
017320 012777 000340 161604
017326 012777 026716 161552
017334 012777 177400 161542
017342 012777 000161 161530
017350 104446

:SETUP CONTROLLER TO TRANSFER 256 WORDS OF DATA (2 SECTORS)
MOV #INTMR,0RSVEC ;SETUP INTERRUPT VECTOR
MOV #340,0RSVCPS
MOV #INBUF,0RSBA ;LOAD BUS ADDR REG
MOV #177400,0RSWC ;LOAD WORD COUNT REG
MOV #161,0RSCSI ;LOAD WRITE COMMAND I/E
GETSP ;CLOCK ROUTINE TO GET SECTOR PULSE

017352 104220
017354 104450

HLT !MR
SPASS

;TO CLEAR OUT COUNTERS AND REGISTERS
;THAT OTHERWISE COULD NOT BE CLEARED.
;COULD NOT SET SECTOR PULSE (0)
;CLOCK MR REG SP = 1

:ASSERT INDEX PULSE TO INITIALIZE THE DRIVE

017356 104430
017360 104420
017362 020501
017364 104000MRIND
MRCK
20501
HLT:CHECK MR REG TO EQUAL
:20501 FOR A
:WRITE COMD

:STEP THRU RESYNC PERIOD

017366 012767 001000 161606
017374 052767 000040 161564
017402 104422
017404 104420
017406 070501
017410 104000
017412 104422
017414 104420
017416 020501
017420 104000
017422 005367 161554
017426 001365MREX1: MOV #512, REPT
BIS #BIT5, ONCEE
MRCLK
MRCK
70501
HLT
MRCLK
MRCK
20501
HLT
DEC REPT
BNE MREX1:TYPE OUT CLOCK COUNT IF ERROR OCCURS
:CLOCK MR REG
:CHECK FOR
:CORRECT DATA
:MR = BAD GOOD = CORRECT DATA
:CLOCK MR REG
:CHECK FOR
:CORRECT DATA
:ERROR WHILE CLOCKING THROUGH RESYNC PERIOD
:FINISH LOOPING
:THROUGH RESYNC PERIOD:ONE MORE CLOCK PULSE SHOULD ASSERT SECTOR PULSE
:SP=0 EQUALS SECTOR PULSE017430 104422
017432 104420
017434 070101
017436 104000
017440 104422
017442 104420
017444 020101
017446 104000MRCLK
MRCK
70101
HLT
MRCLK
MRCK
20101
HLT:CLOCK MR REG
:MR SHOULD
:EQUAL 70101
:MR=BAD GOOD=CORRECT ANS
:CLOCK MR REG
:CHECK MR
:TO EQUAL 20101
:MR=BAD GOOD=CORRECT ANS

:PERFORM 63 DOUBLE MAINT CLOCK OPERATIONS--WRITING PREAMBLE

017450 012767 000077 161524
017456 104422
017460 104420
017462 071501
017464 104000
017466 104422
017470 104420
017472 021501
017474 104000
017476 005367 161500
017502 001365MREX2: MOV #63, REPT
MRCLK
MRCK
71501
HLT
MRCLK
MRCK
21501
HLT
DEC REPT
BNE MREX2:CLOCK MR REG
:CHECK MR REG
:TO EQUAL 71501
:MR=BAD GOOD=CORRECT ANS
:CLOCK MR REG
:CHECK MR REG
:TO EQUAL 21501
:MR=BAD GOOD=CORRECT ANS
:DONE YET
:NO LOOP

;DRIVE SHOULD NOW RECEIVE 1ST WORD TO BE WRITTEN

017504	104422		MRCLK		;CLOCK MR REG
017506	104420		MRCK		;CHECK MR REG
017510	171501		171501		;TO EQUAL 171501
017512	104000		HLT		;MR REG=BAD GOOD=CORRECT ANS
017514	104422		MRCLK		;CLOCK MR REG
017516	104420		MRCK		;MR REG SHOULD
017520	025501		25501		;EQUAL 25501
017522	104000		HLT		;MR REG=BAD GOOD=CORRECT ANS
017524	104422		MRCLK		
017526	104420		MRCK		
017530	175501		175501		
017532	104000		HLT		

;PERFORM NEXT STEP 3 TIMES TO FINISH WRITTING PREAMBLE

017534	012767	000003	161440		MOV	#3,REPT	
017542	104422			MREX3:	MRCLK		;CLOCK MR REG
017544	104420				MRCK		;CHECK MR REG
017546	025501				25501		;TO EQUAL 25501
017550	104000				HLT		;MR=BAD GOOD=CORRECT ANS
017552	104422				MRCLK		;CLOCK MR REG
017554	104420				MRCK		;CHECK MR REG
017556	175501				175501		;TO EQUAL 175501
017560	104000				HLT		;MR REG=BAD GOOD=CORRECT ANS
017562	005367	161414			DEC	REPT	;DONE YES?
017566	001365				BNE	MREX3	;NO LOOP BACK

;MOVE DATA WORD INTO RS04 SHIFT REGISTER

017570	104422		MRCLK		;CLOCK MR REG
017572	104420		MRCK		;CHECK MR REG
017574	027501		27501		;TO EQUAL 27501
017576	104000		HLT		;MR=BAD GOOD=CORRECT ANS
017600	104422		MRCLK		;CLOCK MR REG
017602	104420		MRCK		;MR REG SHOULD
017604	123501		123501		;EQUAL 123501
017606	104000		HLT		;MR=BAD GOOD=CORRECT ANS

;ENCODE SYNC 1

017610	104422		MRCLK		;CLOCK MR REG
017612	104420		MRCK		;MR REG SHOULD NOW
017614	073501		73501		;EQUAL 73501
017616	104000		HLT		;MR=BAD GOOD=CORRECT ANS
017620	012705	026716	MOV	#INBUF,R5	;GET STARTING ADDR FOR DATA BUFFER
017624	011504		MOV	(R5),R4	;GET DATA

017626	012767	002167	161360	MOV	#1143.,WORK	:DOING A 1 SECTOR TRANSFER 127 WORDS
						:18 BITS PER WORD-CLOCK LOOPS
						:TAKE CARE OF 2 BITS AT A TIME
						:127 TIMES 9 EQUALS 1143 LOOPS
						:TO GET THROUGH SECTOR (LAST WORD DONE SEPARATELY).
017634	042767	000200	161324	BIC	#BIT7,ONCEE	:CLEAR LAST WORD FLAG
017642	052767	000100	161316	BIS	#BIT6,ONCEE	:SET 1ST TRANSFER WORD FLAG
017650	104432			1\$: XBIT		:GET 2 BITS OF DATA
017652	104434			CLKD1		:SEND FIRST CLOCK PULSE
						:AND CALCULATE MR REG
						:FOR CORRECT DATA (MWDI+MWDB)
017654	104000			HLT		:MR REG NOT CORRECT
017656	104436			CLKD2		:SEND 2ND CLOCK PULSE TO
						:COMPLETE TRANSFER OF 2 BITS
						:CALCULATE CORRECT ANS FOR
						:MR REG (MWDI+MWDB)
017660	104000			HLT		:MR=BAD GOOD=CORRECT ANS
017662	032767	000200	161276	BIT	#BIT7,ONCEE	:ON LAST WORD YET
017670	001015			BNE	2\$	:YES
017672	032767	000400	161266	BIT	#BIT8,ONCEE	:ON CRC WORD YET?
017700	001043			BNE	3\$	:YES
017702	005367	161306		DEC	WORK	:DONE WITH 127 WORDS?
017706	001360			BNE	1\$	:NO
017710	052767	000200	161250	BIS	#BIT7,ONCEE	:SET LAST WORD FLAG
017716	012767	000012	161270	MOV	#10.,WORK	:SET UP TO TRANSFER LAST WORD
017724	005367	161264		2\$: DEC	WORK	:DONE YET
017730	001347			BNE	1\$	
017732	052767	000400	161226	BIS	#BIT8,ONCEE	:SET TRANSFERRING CRC WORD
017740	042767	000200	161220	BIC	#BIT7,ONCEE	:CLEAR LAST WORD FLAG
						:GENERATE RMR ERROR BY ATTEMPTING TO WRITE RSR
						:EXC SHOULD THEN BE ASSERTED
017746	012777	177777	161140	MOV	#-1,RSER	
017754	004767	003572		JSR	PC,GENCRC	:GENERATE CRC WORD
						:AND LEAVE IN "WORK"
017760	012702	026716		MOV	#INBUF,R2	:GO TO END
017764	062702	000400		ADD	#400,R2	:OF DATA BUFFER
017770	016712	161220		MOV	WORK,R2	:LOAD CRC WORD
017774	010205			MOV	R2,R5	:RESET POINTER FOR
017776	162705	000002		SUB	#2,R5	:R5 FOR CRC WD
020002	012767	000012	161204	MOV	#10.,WORK	:SETUP TO XFER CRC
020010	005367	161200		3\$: DEC	WORK	:DONE YET?
020014	001315			BNE	1\$	:NO


```

020016 104422      ;EBL SHOULD NOW ASSERT AND CRC BE WRITTEN
020020 104420      MRCLK      ;CLOCK MR REG TO STEP THROUGH DEAD BAND AREA
020022 153501      MRCK      ;CHECK MR REG
020024 104000      153501    ;TO EQUAL 103501
                                ;MR REG=BAD GOOD=CORRECT ANS
                                HLT

```

```

020026 012767 000006 161146 ;LOOP 6 TIMES
020034 104422 4$:      MOV      #6, REPT
020036 104420      MRCLK
020040 003501      MRCK
020042 104000      3501
020044 104422      HLT
020046 104420      MRCLK
020050 153501      MRCK
020052 104000      153501
020054 005367 161122    HLT
020060 001365      DEC      REPT
                                BNE      4$
                                ;CLOCK MR REG
                                ;CHECK MR REG
                                ;TO EQUAL 53501
                                ;MR=BAD GOOD=CORRECT ANS
                                ;CLOCK MR REG
                                ;CHECK MR REG
                                ;TO EQUAL 103501
                                ;MR=BAD GOOD=CORRECT ANS
                                ;DONE LOOPING YET?
                                ;NO

```

```

020062 104422      ;FINISH UP
020064 104420      MRCLK
020066 003501      MRCK
020070 104000      3501
020072 104422      HLT
020074 104420      MRCLK
020076 151501      MRCK
020100 104000      151501
                                HLT
                                ;CLOCK MR REG
                                ;CHECK MR REG
                                ;TO EQUAL 3501
                                ;MR REG=BAD GOOD=CORRECT ANS
                                ;CLOCK MR REG
                                ;CHECK MR REG
                                ;TO EQUAL 151501
                                ;MR=BAD GOOD=CORRECT ANS

```

```

020102 104422      ;TRANSFER SHOULD NOW BE COMPLETE
020104 104420      MRCLK
020106 002701      MRCK
020110 104000      2701
020112 000240      HLT
020114 104050      NOP
020116 000424      HLT      !WC!DS
                                BR      INTMR1
                                ;CLOCK MR REG
                                ;CHECK MR REG
                                ;TO EQUAL 2701
                                ;MR=BAD GOOD=CORRECT ANS
                                ;STALL FOR TIME
                                ;SHOULD NEVER GET HERE
                                ;BECAUSE DRIVE SHOULD HAVE INTERRUPTED.
                                ;CAUSING JUMP TO INTMR.
                                ;CHECK FOR ASSERTION OF FT5 ATTN L

```

```

                                ;NOW TEST CONTROLLER
020120 022777 144260 160752 INTMR: CMP      #144260, @RSCS1 ;IS CS1 CORRECT?
020126 001401      BEQ      #4      ;YES
020130 104014      HLT      !DA!WC ;YES
020132 022777 177610 160744 5$: CMP      #177610, @RSWC ;IS WC REG CORRECT?
020140 001401      BEQ      +4      ;YES
020142 104010      HLT      !WC    ;WC SHOULD BE = TO 177610
020144 022777 000004 160742    CMP      #4, @RSER ;DID RMR SET IN RSER
020152 001401      BEQ      +4      ;YES
020154 104050      HLT      !DS!WC ;RSER SHOULD = 4
020156 022777 000001 160724    CMP      #1, @RSDA ;DOES RSDA=1
020164 001401      BEQ      +4      ;YES
020166 104004      HLT      !DA    ;RSDA SHOULD=1
020170 000240      INTMR1: NOP      ;DONE

```

020172 104400

```

*****
:TEST 63 DISK ADDRESS OVERFLOW TEST
*****
TST63: SCOPE

```

```

:MODULES TESTED: M7754, M7771, M7770
:SET UP TO TRANSFER 2 SECTORS TO THE DISK, STARTING AT TRACK 77 SECTOR 77
:TO CAUSE A DISK ADDRESS OVERFLOW CONDITION. ALSO CHECK LAST BLOCK TRANSFER
:(LBT) BIT TO SET IN THE RSDS REGISTER.

```

020174 104414
020176 012706 000500
020202 104430
020204 104420
020206 022701
020210 104424

```

MRAOE: CLRDK      ;CLEAR ALL REGISTERS
        MOV        ;SETUP STACK POINTER
        MRIND      ;SEND INDEX PULSE TO MR REG
        MRCK       ;CHECK MAINT REG
        22701      ;TO EQUAL 22701
        MRINT      ;INITIALIZE MAINT REG BY JENDING
                2 CLOCK PULSES (CLEAR MRSP)
        MOV        ;SETUP DISK ADDRESS
        MOV        ;SETUP FOR A 2 SECTOR TRANSFER
        MOV        ;GET OUTPUT BUFFER
        #500,SP

```

020212 012777 007777 160670
020220 012777 177400 160656
020226 012777 027516 160652

```

;SETUP BUFFER WITH ALL ONES
        MOV        ;GET STARTING ADDRESS OF OUTBUF
        MOV        ;LOAD 2 SECTORS
        MOV        ;WITH WORDS
        DEC        ;OF ALL ONES
        REPT
        BNE
        1$

```

020260 012777 000061 160612
020266 104430

```

        MOV        ;LOAD WRITE COMMAND
        MRIND      ;SET INDEX PULSE
        #61,ARSCS1

```

```

;SUPPLY CLOCKS TO STEP THROUGH A TRACK

```

020270 012767 000003 160704
020276 012704 160000
020302 012702 000011
020306 012703 000001
020312 010277 160606
020316 010377 160602
020322 005304
020324 001372
020326 005367 160650
020332 001361

```

5$: MOV        #3,REPT
    MOV        #57344.,R4
    MOV        #11,R2
    MOV        #1,R3
    MOV        R2,ARSMR
    MOV        R3,ARSMR
    DEC        R4
    BNE        2$
    DEC        REPT
    BNE        5$

```

```

;CAUSE "LBT IN RSDS TO SET

```

020334 104422
020336 104426
020340 012400
020342 104000

```

MRCLK
DSCK
12400
HLT
;CLOCK AN 11 AND A 1 INTO RSMR
;CHECK MR
;TO EQUAL 12400
;LBT SHOULD BE SET IN RSDS

```


;ASSERT MAINTENANCE INDEX PULSE TO RESET DRIVE
;FOR THE SECOND REVOLUTION

020344 104430
020346 005067 160622
020352 104420
020354 002501
020356 104000

MRIND
CLR MCCNT
MRCK
2501
HLT

;ASSERT MAINT INDEX PULSE
;CLEAR THE CLOCK COUNTER
;CHECK MR REG
;TO EQUAL 2501. SHOULD STILL BE WRITING

020360 012767 001000 160614
020366 104422
020370 104420
020372 052501
020374 104000
020376 104422
020400 104420
020402 002501
020404 104000
020406 005367 160570
020412 001365

;SUPPLY ENOUGH CLOCKS TO STEP THROUGH THE R504 RESYNC PERIOD
;CLOCK COUNT TO STEP THRU RESYNC
4\$: MOV #512.,REPT

MRCLK
MRCK
52501
HLT
MRCLK
MRCK
2501
HLT
DEC REPT
BNE 4\$

;2ND REVOLUTION
;CHECK MR
;TO EQUAL 52501
;MR=BAD GOOD=CORRECT ANS
;CLOCK MR REG
;CHECK MR
;REG TO
;EQUAL 2501
;LOOP TILL DONE

;SUPPLY 2 CLOCKS TO CAUSE THE SECTOR PULSE TO APPEAR IN
;THE MR REGISTER AND THE "AOE" ERROR TO APPEAR IN
;THE RSER REGISTER

020414 104422
020416 104422
020420 104420
020422 022301
020424 104000
020426 022777 001000 160460
020434 001401
020436 104040
020440 022777 152600 160444 1\$:
020446 001401
020450 104040

AOECK:

MRCLK
MRCLK
MRCK
22301
HLT
CMP
BEQ
HLT
CMP
BEQ
HLT

#1000,RSER
1\$
!DS
#152600,RSDS
2\$
!DS

;CAUSE SECTOR PULSE AND AOE ERROR
;CHECK FOR SECTOR PULSE
;IN RSMR
;MR=BAD GOOD=CORRECT ANS
;DID AOE SET IN RSER?
;AOE SHOULD BE SET IN RSER
;RSER SHOULD EQUAL 1000
;IS RSDS CORRECT
;YES
;ERR & ATA SHOULD BE SET IN RSDS
;BECAUSE OF AOE ERROR IN RSER
;CLEAR ERROR
;DID ERROR CLEAR?
;YES
;AOE DID NOT CLEAR BY SETTING CLR IN RSCS2
;DID ERRORS CLEAR
;YES
;ERR AND ATA & LBT SHOULD ALL BE CLEARED
;FOR CLR WAS SET IN RSCS2

020452 104414
020454 005777 160434 2\$:
020460 001401
020462 104040
020464 022777 010600 160420 3\$:
020472 001401
020474 104040

CLRDK
TST
BEQ
HLT
CMP
BEQ
HLT

RSER
3\$
!DS
#10600,RSDS
+4
!DS

NO7

;MAINTENANCE MODE VERIFY TEST
;-----DANGER---THIS TEST DESTROYS DATA ON DISKS--DANGER
;THIS TEST WILL ONLY RUN IF SWITCH 11 IS SET IN THE "SWITCH
;REGISTER" FOR IT WILL ACTUALLY WRITE DATA INTO THE DISK. IT
;WILL WRITE ONE TRACK OF ALL ONES. THE PROGRAM THEN GOES BACK
;TO THE MAINT WRITE TEST AND WRITES ONE SECTOR OF DATA (ZERO'S, ONES, FLOATING
;ONES AND FILLS THE REMAINDER OF SECTOR WITH A PATTERN OF 146314)
;THE DRIVE IS THEN TAKEN OUT OF
;"MAINTENANCE MODE" AND THE TRACK IS THEN READ. THE TRACK
;SHOULD CONTAIN ALL ONES.

;*****
;TEST 64 MAINTENANCE MODE VERIFY TEST
;*****
TST64: SCOPE

020476 104400

;MODULE TESTED G182

020500	032767	004000	157062	MRVR:	BIT	#BIT11,SWR	;DO THIS TEST?
020506	001002				BNE	3\$	;YES
020510	000137	021224			JMP	#INFTST	;NO
020514	005067	160424		3\$:	CLR	FLAG2	;SET VERIFY TEST FLAG
020520	104414				CLRDK		;CLEAR ALL DRIVES
020522	012767	177777	160500		MOV	#177777,WORK5	;STALL TO
020530	005367	160474		4\$:	DEC	WORK5	;RESYNC DRIVE
020534	001375				BNE	4\$	;TIMING LOGIC
020536	042767	000040	160422		BIC	#BIT5,ONCE	;CLEAR CLK CNT
020544	012777	160000	160332		MOV	#-20000,DRSWC	;WRITE ONE TRACK - 8K WDS
020552	012767	177777	006136		MOV	#177777,INBUF	;WRITE A PATTERN 12525
020560	052777	000010	160314		BIS	#BIT3,DRSCS2	;SET BAI BIT
020566	012777	026716	160312		MOV	#INBUF,DRSBA	;SET DATA WD
020574	012767	177777	160400		MOV	#177777,REPT	;SETUP WAIT LOOP
020602	012777	000061	160270		MOV	#61,DRSCS1	;GO WRITE
020610	105777	160264		1\$:	TSTB	DRSCS1	;DONE YET?
020614	100404				BMI	2\$	;YES
020616	005367	160360			DEC	REPT	;DECREMENT COUNTER WAITING
020622	001372				BNE	1\$	;FOR READY
020624	104000				HLT		;READY NEVER CAME UP
020626	005777	160246		2\$:	TST	DRSCS1	;ANY ERRORS?
020632	100002				BPL	MRVR1	;NO
020634	104050				HLT	!DS!WC	;STOP HERE TILL THIS PROBLEM IS FIXED TRY DZRSB DIAG
020636	000433				BR	TBDIA	;TYPE MESSAGE

020640	104414		MRVR1:	CLDK		:CLEAR ALL REGISTERS
020642	012777	160000		MOV	#-20000, ARSWC	:SETUP WC
020650	052777	000010		BIS	#BIT3, ARSCS2	:SET BAI
020656	012777	026716		MOV	#INBUF, ARSBA	:SETUP RSBA
020664	012767	177777		MOV	#177777, REPT	:SETUP WAIT LOOP
020672	012777	000051		MOV	#51, ARSCS1	:DO A WRITE CHECK TO VERIFY DISK
020700	105777	160174	1\$:	TSTB	ARSCS1	:TEST
020704	100404			BMI	2\$	:FOR READY TO COME BACK
020706	005367	160270		DEC	REPT	:WAIT
020712	001372			BNE	1\$	
020714	104000			HLT		:READY NEVER CAME BACK
020716	005777	160156	2\$:	TST	ARSCS1	:ANY ERRORS?
020722	100032			BPL	MRVR	:NO
020724	104050			HLT	!DS!WC	:STOP HERE WC FAILED
						:GO TO DZRSB DIAG
						:BEFORE TRYING TO DEBUG TILIS TEST
020726			TBDIA:			
020726	104402	020732		TYPE	.+2	:ASCIZ <15><12>"FAILED VERIFY TEST --- RUN DZRSB DIAGNO
021010	000137	011552	MRVR:	JMP	MRWRT	:GO WRITE IN MAINTENANCE MODE
						:NOW CHECK TO SEE IF DRIVE WAS WRITTEN ON IN MAINTENANCE MODE
021014	104414		MRVR2:	CLDK		:CLEAR ALL REGISTERS
021016	012767	177777		MOV	#177777, WORK	:STALL - TO RESPONSE
021024	005367	160164	3\$:	DEC	WORK	:INDEX PULSE
021030	001375			BNE	3\$	:ON DRIVE
021032	012777	160000		MOV	#-20000, ARSWC	:SETUP WC FOR 1 TRACK
021040	052777	000010		BIS	#BAI, ARSCS2	:SET BAI
021046	012777	026716		MOV	#INBUF, ARSBA	:SETUP RSBA
021054	012767	177777		MOV	#177777, INBUF	:SETUP FOR COMPARE
021062	012777	000051		MOV	#51, ARSCS1	:DO A WRITE CHECK
021070	105777	160004	1\$:	TSTB	ARSCS1	:TEST FOR
021074	100375			BPL	1\$	:READY TO COME BACK
021076	032777	040000		BIT	#WCE, ARSCS2	:DID WCE SET?
021104	001442			BEQ	2\$	:NO
021106	104402	021112		TYPE	.+2	:ASCIZ <15><12>"WRITE AMPLIFIER DID NOT GET DISABLED B
021206	104040			HLT	!DS	
021210	000404			BR	4\$	:GET OUT
021212	005777	157662	2\$:	TST	ARSCS1	:ANY ERRORS?
021216	100001			BPL	4\$	:NO
021220	104040			HLT	!DS	:SHOULD NOT BE ANY ERRORS
						:TRY THE DZRSB DIAGNOSTIC
021222	000240		4\$:	NOP		
021224	000137	002212	INFTST:	JMP	TRYNX	:GET NEXT DRIVE

```
.SBTTL          SDONE - BELL AND SCOPE ROUTINE

021230 104400    000001 157546  DONE:  SCOPE          : TERMINATING SCOPE FOR LOOPING
021232 062767    000001 157546          ADD          #1,PCNT+2      : ADD 1 TO THE PASS COUNT
021240 005567    157540          ADC          PCNT          : MAKE IT DOUBLE PREC.
021244 032737    002000 177570          BIT          #SW10,2#SWR    : RING THE BELL?
021252 001004          4S          BNE          4S          : NO!
021254 104402    021260          TYPE          +2          : .ASCIZ <BELL><177>
021264 013700    000042 4S:          MOV          2#42,RO      : GET MONITOR ADDRESS
021270 001405          BEQ          SEND1          : IF NONE
021272 000005          RESET
021274 004710          SENDAD: JSR          7,(0)          : GO TO MONITOR
021276 000240    000240 000240 SEND1: JMP          240,240,240 : SAVE ROOM FOR ACT11
021304 000167    000002          MULSYS          : RETURN

021310 000000          .TBIT: 0          : T BIT FLAG

          ;MULTI DRIVE SYSTEM?

021312          MULSYS:          TYPE          +2          : .ASCIZ <15><12>"END OF PASS"
021312 104402    021316          CLR          LAD          :
021334 005067    157450          CLR          ICNT          :
021340 005067    157434          BIT          #BIT4,ONCE     : MULTI DRVIE?
021344 032767    000020 157614          BNE          1S          : NO
021352 001002          1S:          JMP          2#MULTI      : YES
021354 000137    001766          JMP          2#NOWGO        : TEST ONLY ONE DRIVE
021360 000137    002416          ;ERROR TYPEOUT ROUTINE FOR NO-OP TEST

021364 032767    000004 157574 NOPERR: BIT          #BIT2,ONCE     : WERE WE HERE BEFORE?
021372 001031          1S          BNE          1S          : YES
021374 052767    000004 157564          BIS          #BIT2,ONCE : SET BEEN HERE BEFORE FLAG
021402 104402    021406          TYPE          +2          : .ASCIZ <15><12>"ERROR CAUSED BY NO-OP FUNCTION "
021450 016746    157540          MOV          WORK,-(6)      : PUT WORK ON STACK
021454 104406          TYPES          : TYPE STACK IN OCTAL - SUPRESS
021456 000207          1S:          RTS          PC
```


008

```
021460 104402 021474 CHG: TYPE ,REGCHG ;TYPE MESSAGE
021464 016746 157524 MOV WORK,-(6) ;PUT WORK ON STACK
021470 104406 TYPES ;TYPE STACK IN OCTAL - SUPRESS
021472 000207 RTS PC

021474 044103 047101 042507 REGCHG: .ASCIZ "CHANGED WITH NO-OP FUNCTION "
021502 020104 044527 044124
021510 047040 026517 050117
021516 043040 047125 052103
021524 047511 020116 000

021531 015 051012 051115 TRMR: .ASCIZ <15><12>"RMR DID NOT SET BY WRITING INTO "
021536 020040 044504 020104
021544 047516 020124 042523
021552 020124 054502 053440
021560 044522 044524 043516
021566 044440 052116 020117
021574 000
021576 .EVEN

021576 104422 .MRINT: .MRCLK ;CLOCK THE MAINT REG WITH AN 11 AND A 1
021600 104422 .MRCLK ;SAME
021602 000002 RTI ;RETURN

021604 012777 000011 157312 .MRCLK: MOV #11,RSMR ;CLOCK THE
021612 012777 000001 157304 MOV #1,RSMR ;MAINT REG
021620 062767 000001 157350 ADD #1,MCCNT+2 ;ADD 1 TO CLOCK COUNT
021626 005567 157342 ADC MCCNT ;MAKE DOUBLE PRECISION
021632 000002 RTI

021634 017700 157264 .MRCK: MOV RSMR,BAD ;GET THE CONTENTS OF RSMR
021640 017601 000000 MOV @ (SP),GOOD ;GET THE CORRECT ANSWER
021644 062716 000002 ADD #2,(SP) ;UPDATE THE RETURN ADDRESS FOR AN ERROR
021650 020100 CMP GOOD,BAD ;IS THE MR REG CORRECT?
021652 001002 BNE IS ;NO EXIT
021654 062716 000002 IS: ADD #2,(SP) ;UPDATE RETURN ADDRESS TO SKIP THE HLT FOR CORRECT ANS
021660 000002 RTI ;RETURN

021662 012777 000021 157234 ;SEND INDEX PULSE TO THE MAINTENANCE REGISTER
021670 012777 000001 157226 .MRIND: MOV #21,RSMR ;SEND INDEX
021676 000002 MOV #1,RSMR ;PULSE TO MR REG
021700 017700 157206 .DSCK: MOV RSDS,BAD ;GET THE CONTENTS OF RSDS
021704 017601 000000 MOV @ (SP),GOOD ;GET THE CORRECT ANS
021710 062716 000002 ADD #2,(SP) ;UPDATE THE RETURN ADDR FOR AN ERROR
021714 020100 CMP GOOD,BAD ;IS RSDS CORRECT
021716 001002 BNE IS ;NO EXIT
021720 062716 000002 IS: ADD #2,(SP) ;UPDATE RETURN ADDR TO SKIP THE HLT FOR CORRECT ANS
021724 000002 RTI
```

:GET 2 BITS OF DATA FROM BUFFER
:SET NOWOD (ODD DATA BIT NOW BEING WRITTEN) TO A 1 IF DATA TRANSFER BIT IS A 1
:CLEAR NOWOD IF DATA BIT IS A 0
:SET NOWEV (EVEN DATA BIT NOW BEING WRITTEN) TO A 1 IF DATA TRANSFER BIT IS A 1
:CLEAR NOWEV IF DATA BIT IS A 0

021726	032767	000100	157232	.XBIT:	BIT	#BIT6,ONCE	:1ST 2 BITS OF 1ST WORD?
021734	001427				BEG	2\$	:NO
021736	012767	000001	157202		MOV	#1,LSTEV	:SET LAST EVEN BIT TRANSFERRED TO A 1
021744	012767	000001	157176		MOV	#1,LSTOD	:SET LAST ODD BIT TRANSFERRED TO A 1
							:THIS SETS UP THE SYNC 1 BITS AT END OF PREAMBLE
							:FOR THE TOP AND BOTTOM
							:BITS IN THE MR REGISTER
							:CLEAR 1ST WORD TRANSFER FLAG
							:CLEAR CLOCK COUNTER AT START OF EACH WORD
							:CRC WORD BEING WRITTEN?
							:YES
							:NO, LOAD EVEN
							:AND ODD WITH 0 FOR BITS 16 & 17 IN R504 DATA WORD.
							:8 LOOPS FOR REMAINING 16 BITS OF WORD
021752	042767	000100	157206		BIC	#BIT6,ONCE	:SAVE LAST 2 BITS TRANSFERRED
021760	005067	157222		4\$:	CLR	CLKCNT	:DONE WITH WORD YET?
021764	032767	000400	157174		BIT	#BIT8,ONCE	:NO
021772	001042				BNE	1\$	:UPDATE BUFFER WD
021774	005067	157154			CLR	NOWOD	:GET DATA WD
022000	005067	157146			CLR	NOWEV	:GET BITS 16 & 17
022004	012767	000010	157212	6\$:	MOV	#8,WORK3	:CLEAR PRESENT ODD BIT
022012	000002				RTI		:GET NEXT ODD DATA BIT
022014	016767	157134	157126	2\$:	MOV	NOWOD,LSTOD	:SAVE IT IN ODD BIT
022022	016767	157124	157116		MOV	NOWEV,LSTEV	:CLEAR PRESENT EVEN BIT
022030	005767	157170			TST	WORK3	:GET NEXT EVEN BIT
022034	001004				BNE	3\$	:SAVE IT IN EVEN BIT
022036	062705	000002			ADD	#2,R5	:KEEP COUNT OF BITS IN THE WORD
022042	011504				MOV	(R5),R4	:RETURN
022044	000745				BR	4\$	
022046	005067	157102		3\$:	CLR	NOWOD	
022052	006104				ROL	R4	
022054	006167	157074			ROL	NOWOD	
022060	005067	157066			CLR	NOWEV	
022064	006104				ROL	R4	
022066	006167	157060			ROL	NOWEV	
022072	005367	157126			DEC	WORK3	
022076	000002				RTI		
022100	005067	157050					:CRC WORD IS BEING WRITTEN BIT 17 & 16 ARE DATA BITS. 0 & 1 ARE ALWAYS 0
022104	006104			1\$:	CLR	NOWOD	:GET BITS 17
022106	006167	157042			ROL	R4	:AND 16
022112	005067	157034			ROL	NOWOD	:FOR CRC WORD
022116	006104				CLR	NOWEV	
022120	006167	157026			ROL	R4	
022124	000727				ROL	NOWEV	
					BR	6\$	:CONTINUE

F08

22-CERSD-D
CERSDD.F11

RH70-R504 MAINTENANCE MODE DIAGNOSTIC
23-JAN-78 13:36

MACY11 30A(1052) 23-JAN-78 13:38 PAGE 136
SDONE - BELL AND SCOPE ROUTINE

SEQ 0096

;CLOCK ROUTINE (1ST OF TWO) WHICH IS USED TO CLOCK TWO BITS OF
;DATA TO THE DRIVE AT A TIME. THIS ROUTINE ALSO CHECKS THE PREVIOUS
;BITS THAT HAVE BEEN TRANSFERRED AND CALCULATES WHICH STATE
;THE MWDT BIT (BIT 14 IN THE MR REG) AND MWDB BIT (BIT 12 IN THE MR REG) SHOULD BE IN

022126 104422
022130 005003
022132 005767 157016
022136 001005

022140 005767 157004
022144 001002

022146 052703 010000

022152 005767 156774

022156 001005
022160 005767 156762
022164 001002
022166 052703 040000
022172 012701 123501
022176 050301
022200 004767 001200
022204 017700 156714
022210 020100
022212 001002
022214 062716 000002
022220 000002

.CLKD1: MRCLK
CLR R3
TST NOWOD
BNE TSTVB

1\$: TST LSTOD
BNE TSTVB

2\$: BIS #BIT12,R3

;NOW TEST FOR EVEN BITS BEING SENT

TSTVB: TST NOWEV
BNE 1\$
TST LSTEV
BNE 1\$
BIS #BIT14,R3
1\$: MOV #123501,GOOD
BIS R3,GOOD
JSR PC,MRCAL
MOV JRSMB,BAD
CMP GOOD,BAD
BNE 2\$
ADD #2,(SP)
2\$: RTI

;CLOCK MR REG WITH AN 11 AND A 1
;CLEAR WORK LOCATION
;TEST ODD BIT NOW BEING SENT FOR A 1 OR A 0
;NOW TEST EVEN DATA BIT ON 1ST CLOCK
;NOW BIT IS A 1 MWDB IS 0
;TEST THE LAST ODD DATA BIT THAT WAS SENT
;LAST ODD DATA BIT WAS A 1
;MWDB IS A 0

;SET MWDB FOR LATER COMPARE WITH MR REG

;TEST EVEN BIT NOW BEING TRANSFERRED
;FOR EITHER A 1 OR A 0
;NOW BIT IS A 1
;WAS LAST EVEN DATA BIT A 0?
;NO LAST EVEN DATA BIT WAS A 1
;MWDT SHOULD BE SET
;GET CORRECT ANS
;FOR MR REG
;DETERMINE STATE OF SB & LSR BITS
;GET CONTENTS OF MR REG
;IS MR REG CORRECT?
;NO TYPE OUT MR REG
;UPDATE RETURN ADDR FOR CORRECT ANS
;RETURN

;SECOND CLOCK ROUTINE WHICH WILL FINISH TRANSFERRING THE TWO DATA BITS
;THIS ROUTINE WILL CALCULATE WHAT MWDT AND MWDB SHOULD EQUAL IN THE
;MAINTENANCE REGISTER

022222	104422		.CLKD2: MRCLK		;CLOCK MR REG
022224	005767	156724	TST	NOWOD	;IS THE PRESENT DATA BIT A 1?
022230	001403		BEQ	1\$	;NO IT IS A 0
022232	052703	010000	BIS	#BIT12,R3	;SET MWDB FOR BIT BEING SENT IS A 1
022236	000402		BR	2\$	
022240	042703	010000	1\$: BIC	#BIT12,R3	;CLEAR MWDB FOR PRESENT BIT IS A 0
022244	005767	156702	2\$: TST	NOWEV	;IS PRESENT EVEN BIT A 1
022250	001403		BEQ	3\$	;NO IT IS A 0
022252	052703	040000	BIS	#BIT14,R3	;IT IS A 1 SET MWDT
022256	000402		BR	4\$	
022260	042703	040000	3\$: BIC	#BIT14,R3	;PRESENT BIT IS A 0 CLEAR MWDT
022264	012701	023501	4\$: MOV	#23501,GOOD	;GET CORRECT ANS
022270	050301		BIS	R3,GOOD	;FOR MR REG
022272	004767	001106	JSR	PC,MRCAL	;DETERMINE STATE OF SB & LSR BITS
022276	017700	156622	MOV	QASMR,BAD	;GET CONTENTS OF MR REG
022302	020100		CMP	GOOD,BAD	;IS MR REG CORRECT?
022304	001002		BNE	5\$	;NO TIMEOUT ERROR
022306	062716	000002	ADD	#2,(SP)	;UPDATE RETURN ADDR FOR CORRECT ANS
022312	000002		5\$: RTI		;RETURN

;TYPEOUT ROUTINE TO DETERMINE WHICH IC FAILED IN CRC TEST2
;AND TO TYPE IT OUT

022314	012767	022424	156672	CRCTYP: MOV	#CRCTAB,WORK	;GET STARTING LOC OF IC TABLE
022322	012767	000001	156670	MOV	#1,WORK1	;SETUP TO TEST FIRST CHIP
022330	036767	156664	156634	1\$: BIT	WORK1,SAVEE	;WAS IT THIS BIT?
022336	001006			BNE	2\$	;YES TYPE IT
022340	062767	000006	156646	ADD	#6,WORK	;NO INDEX TABLE POINTER
022346	006167	156646		ROL	WORK1	;SETUP TO TEST NEXT CHIP
022352	000766			BR	1\$	;NOW TEST IT
022354	004777	156634		2\$: JSR	PC,QWORK	;TYPE OUT CHIP
022360	104402	022364		TYPE	PC+2	;ASCIZ " IN THE CRC REG SHOULD BE SET"
022422	000207			RTS	PC	

;TABLE FOR CRC TEST 2 TYPEOUT ROUTINE

022424	104402	022564	CRC TAB: TYPE	E302
022430	000207		RTS	PC
022432	104402	022572	TYPE	E305
022436	000207		RTS	PC
022440	104402	022600	TYPE	E307
022444	000207		RTS	PC
022446	104402	022606	TYPE	E3010
022452	000207		RTS	PC
022454	104402	022615	TYPE	E3012
022460	000207		RTS	PC
022462	104402	022624	TYPE	E3015
022466	000207		RTS	PC
022470	104402	022633	TYPE	E242
022474	000207		RTS	PC
022476	104402	022641	TYPE	E245
022502	000207		RTS	PC
022504	104402	022647	TYPE	E247
022510	000207		RTS	PC
022512	104402	022655	TYPE	E2410
022516	000207		RTS	PC
022520	104402	022664	TYPE	E2412
022524	000207		RTS	PC
022526	104402	022673	TYPE	E2415
022532	000207		RTS	PC
022534	104402	022702	TYPE	E192
022540	000207		RTS	PC
022542	104402	022710	TYPE	E197
022546	000207		RTS	PC
022550	104402	022716	TYPE	E1910
022554	000207		RTS	PC
022556	104402	022725	TYPE	E1915
022562	000207		RTS	PC

022564	031505	026460	000062	E302:	.ASCIZ	"E30-2"
022572	031505	026460	000065	E305:	.ASCIZ	"E30-5"
022600	031505	026460	000067	E307:	.ASCIZ	"E30-7"
022606	031505	026460	030061	E3010:	.ASCIZ	"E30-10"
022614	000					
022615	105	030063	030455	E3012:	.ASCIZ	"E30-12"
022622	000062					
022624	031505	026460	032461	E3015:	.ASCIZ	"E30-15"
022632	000					
022633	105	032062	031055	E242:	.ASCIZ	"E24-2"
022640	000					
022641	105	032062	032455	E245:	.ASCIZ	"E24-5"
022646	000					
022647	105	032062	033455	E247:	.ASCIZ	"E24-7"
022654	000					
022655	105	032062	030455	E2410:	.ASCIZ	"E24-10"
022662	000060					
022664	031105	026464	031061	E2412:	.ASCIZ	"E24-12"
022672	000					
022673	105	032062	030455	E2415:	.ASCIZ	"E24-15"
022700	000065					
022702	030505	026471	000062	E192:	.ASCIZ	"E19-2"
022710	030505	026471	000067	E197:	.ASCIZ	"E19-7"
022716	030505	026471	030061	E1910:	.ASCIZ	"E19-10"
022724	000					
022725	105	034461	030455	E1915:	.ASCIZ	"E19-15"
022732	000065					

:GET TWO BITS OF DATA FROM INBUF
:FOR READING FROM DRIVE TO DETERMINE THE
:STATE OF MRDT AND MRDB IN THE MR REG.

022734	005767	156264	.RBIT:	TST	WORK3	:STARTING NEW WD?
022740	001031			BNE	3\$	:NO
022742	062705	000002		ADD	#2,R5	:UPDATE BUFFER WD
022746	011504			MOV	(R5),R4	:GET DATA WD
022750	005067	156232		CLR	CLKCNT	:CLEAR CLOCK COUNTER AT START OF EACH WD
022754	032767	000400	156204	BIT	#BIT8,ONCE	:ON CRC WD?
022762	001035			BNE	1\$	:YES
022764	032767	000040	156152	BIT	#BIT5,FLAG2	:IN CRC TEST ????
022772	001404			BEG	7\$	:NO
022774	012767	000010	156222	MOV	#8.,WORK3	
023002	000410			BR	3\$	
023004	005067	156144	7\$:	CLR	NOWOD	:LOAD EVEN & ODD BITS
023010	005067	156136		CLR	NOWEV	:WITH 0 FOR BITS 16 & 17 IN R504 DATA WORD
023014	012767	000010	156202	6\$:	MOV	#8.,WORK3
023022	000002			RTI		:8 LOOPS FOR REMAINING 16 BITS OF WORD
023024	005067	156124	3\$:	CLR	NOWOD	:CLEAR PRESENT ODD BIT
023030	006104			ROL	R4	:GET NEXT ODD DATA BIT
023032	006167	156116		ROL	NOWOD	:SAVE IT IN ODD BIT
023036	005067	156110		CLR	NOWEV	:CLEAR PRESENT EVEN BIT
023042	006104			ROL	R4	:GET NEXT EVEN BIT
023044	006167	156102		ROL	NOWEV	:SAVE IT IN EVEN BIT
023050	005367	156150		DEC	WORK3	:KEEP COUNT OF BITS IN THE WORD
023054	000002			RTI		:RETURN
023056	005067	156072	:CRC WORD IS BEING WRITTEN BIT 17	1\$:	CLR	NOWOD
023062	006104			ROL	R4	:GET BITS 17
023064	006167	156064		ROL	NOWOD	:AND 16
023070	005067	156056		CLR	NOWEV	:FOR CRC WORD
023074	006104			ROL	R4	
023076	006167	156050		ROL	NOWEV	
023102	000744			BR	6\$	:CONTINUE

023104	004767	000236		.CLKR1: JSR	PC, CALRTB	: CALCULATE TOP AND BOTTOM BITS FOR MR REG
023110	012703	000011		MOV	#11, R3	: SETUP CLOCK BITS
023114	056703	156074		CLOCK: BIS	WORK, R3	: SET TOP & BOTTOM BITS
023120	010377	156000		MOV	R3, ARSMR	: SEND
023124	042703	000010		BIC	#BIT3, R3	: CLOCK
023130	010377	155770		MOV	R3, ARSMR	: PULSE
023134	062767	000001	156034	ADD	#1, MCCNT+2	: INCREMENT
023142	005567	156026		ADC	MCCNT	: CLOCK COUNT
023146	012701	023601		MOV	#23601, GOOD	: CALCULATE CORRECT ANS FOR MR REG
023152	032767	000004	155764	BIT	#BIT2, FLAG2	: WRITE CK TEST?
023160	001402			BEQ	7\$	: NO
023162	052701	000100		BIS	#BIT6, GOOD	: YES SET RD IN MR REG
023166	050301			7\$: BIS	R3, GOOD	
023170	042701	000010		BIC	#BIT3, GOOD	: CLEAR MCLK
023174	032767	000400	155764	BIT	#BIT9, ONCEE	: ON CRC WD?
023202	001406			BEQ	5\$	: NO
023204	022767	000011	155770	CMP	#11, REPT	: SHOULD CRCW BE SET?
023212	001402			BEQ	5\$	: YES
023214	042701	020000		BIC	#20000, GOOD	: CLEAR CRCW
023220	032767	000001	155716	5\$: BIT	#BIT0, FLAG2	: SHOULD SDCLK BE SET?
023226	001004			BNE	1\$	: YES
023230	052767	000001	155706	BIS	#BIT0, FLAG2	: NO
023236	000405			BR	2\$	: CONTINUE
023240	052701	100000		1\$: BIS	#BIT15, GOOD	: SET IT
023244	042767	000001	155672	BIC	#BIT0, FLAG2	: CLEAR FLAG FOR SDCLK FOR NEXT CLOCK PULSE
023252	005367	155726		2\$: DEC	REPT1	: SHOULD SB SET?
023256	001017			BNE	6\$	: NO
023260	012767	000022	155716	MOV	#18, REPT1	: RESET SB COUNTER
023266	052701	004000		BIS	#BIT11, GOOD	: SET SB
023272	032767	000400	155666	3\$: BIT	#BIT8, ONCEE	: ON CRC WD?
023300	001406			BEQ	6\$	: NO
023302	022767	000022	155674	CMP	#22, REPT1	: SHOULD SB AND CRCW BE SET ?
023310	001002			BNE	6\$	: NO
023312	052701	020000		BIS	#20000, GOOD	: SET SB AND CRCW
023316	017700	155602		6\$: MOV	ARSMR, BAD	: GET MR REG
023322	020100			CMP	GOOD, BAD	: IS RSMR CORRECT?
023324	001002			BNE	4\$	: NO
023326	062716	000002		ADD	#2, (SP)	: YES
023332	000002			4\$: RTI		: RETURN
023334	004767	000006		.CLKR2: JSR	PC, CALRTB	
023340	012703	050011		MOV	#50011, R3	
023344	000663			BR	CLOCK	


```

;CALCULATE THE STATE OF MRDT AND MRDB FROM CURRENT INPUT BITS
;LOCATION WORK CONTAINS CORRECT DATA FOR MRDT AND MRDB
023346 005067 155642 CALRTB: CLR WORK ;CLEAR WORK LOCATION
023352 005767 155576 TST NOWOD ;IS CURRENT ODD BIT A 0?
023356 001403 BEQ 1$ ;YES
023360 052767 000004 155626 BIS #BIT2,WORK ;NO SET MRDB
023366 005767 155560 1$: TST NOWEV ;IS CURRENT EVEN BIT A 0?
023372 001403 BEQ 2$ ;YES
023374 052767 000040 155612 BIS #BIT5,WORK ;NO SET MRDT
023402 000207 RTS PC ;RETURN

;CALCULATE MR REG TO DETERMINE THE STATE OF THE CRC-SB AND LSR BITS
;ON THE DIFFERENT CLOCKS ON THE DIFFERENT WORDS THROUGHOUT THE SECTOR

023404 005267 155576 MRCAL: INC CLKCNT ;ADD ONE TO CLOCK COUNT OF WORD
023410 032767 000200 155550 BIT #BIT7,ONCEE ;TRANSFERRING LAST WORD?
023416 001026 BNE LSTWD ;YES
023420 032767 000400 155540 BIT #BIT8,ONCEE ;TRANSFERRING CRC WORD?
023426 001040 BNE CRCWD ;YES
023430 022767 000010 155550 CMP #8.,CLKCNT ;CLOCK COUNT 8 OR GREATER?
023436 101401 BLOS 1$ ;YES
023440 000414 BR 2$ ;GET OUT
023442 022767 000021 155536 1$: CMP #17.,CLKCNT ;CLOCK COUNT 17 OR GREATER?
023450 101410 BLOS 2$ ;YES GET OUT

023452 052701 004000 BIS #BIT11,GOOD ;SET SB BIT
023456 022767 000017 155522 CMP #15.,CLKCNT ;SHOULD LSR BE CLEARED
023464 001002 BNE 2$ ;NO
023466 042701 002000 BIC #BIT10,GOOD ;CLEAR LSR
023472 000207 RTS PC ;RETURN

;CALCULATE MR FOR LAST DATA WORD
023474 022767 000016 155504 LSTWD: CMP #14.,CLKCNT ;IS THIS CLOCK 14 OR LESS?
023502 103011 BHIS 2$ ;YES GETOUT
023504 022767 000017 155474 CMP #15.,CLKCNT ;IS THIS CLOCK 15?
023512 001003 BNE 1$ ;NO
023514 042701 002000 BIC #BIT10,GOOD ;YES CLEAR LSR
023520 000402 BR 2$ ;GET OUT
023522 042701 020000 1$: BIC #BIT13,GOOD ;CLEAR CRCW BIT
023526 000207 RTS PC

;CALCULATE MR FOR CRC WORD
023530 042701 020000 CRCWD: BIC #BIT13,GOOD ;CLEAR CRCW BIT
023534 022767 000017 155444 CMP #17,CLKCNT ;IS THIS CLOCK 17?
023542 001002 BNE 1$ ;NO
023544 042701 002000 BIC #BIT10,GOOD ;CLEAR LSR BIT
023550 000207 RTS PC ;RETURN
```

; GENERATE A CRC WORD FROM THE DATA BUFFER
; AND LEAVE THE CRC WORD IN "WORK" LOCATION
; EXIT ROUTINE WITH RTS PC

023552	012767	000200	155422	GENCRC:	MOV	#128.,REPT	:128 WORDS PER SECTOR
023560	032767	000040	155356		BIT	#BITS,FLAG2	:IN CRC TEST?
023566	001403				BEQ	13\$	:NO
023570	012767	000220	155404		MOV	#144.,REPT	:YES
023576	012705	026716		13\$:	MOV	#INBUF,R5	:GET STARTING ADDR OF OUTPUT BUFFER
023602	011504				MOV	(R5),R4	:GET DATA WD
023604	005067	155406			CLR	WORK0	:CLEAR WORK LOCATION

; INBIT CONTAINS PRESENT INPUT BIT
; WK15 = BIT15 OF CRC AT TIME T
; WORK0 = CRC AT TIME T + DURING FINAL MANIPULATION
; WORK = BITS FROM SAVED CRC WORD (WCRC)

023610	012767	000022	155366	1\$:	MOV	#18.,REPT1	:GET 18 BITS PER WD
023616	032767	000040	155320		BIT	#BITS,FLAG2	:IN CRC TEST?
023624	001403				BEQ	2\$	:NO
023626	012767	000020	155350		MOV	#16.,REPT1	:YES
023634	016767	155356	155336	2\$:	MOV	WORK0,WCRC	:SAVE CURRENT CRC WD
023642	005067	155344			CLR	WK15	:CLEAR BIT 15 FROM CRC AT T 1
023646	000241				CLC		:CLEAR CARY
023650	006167	155342			ROL	WORK0	:SHIFT CRC WD LEFT
023654	006167	155332			ROL	WK15	:CONTAINS BIT 15 OF CRC
023660	032767	000040	155256		BIT	#BITS,FLAG2	:IN CRC TEST?
023666	001004				BNE	12\$	:YES
023670	022767	000021	155306		CMP	#17.,REPT1	:DONE BITS 16 AND 17 YET?
023676	101406				BLOS	3\$	:NO
023700	005067	155304		12\$:	CLR	INBIT	:CLEAR WORK LOC
023704	006104				ROL	R4	:PUT DATA BIT FROM BUFFER
023706	006167	155276			ROL	INBIT	:IN WORK1 LOC
023712	000402				BR	4\$	
023714	005067	155270		3\$:	CLR	INBIT	:FOR BITS 16 AND 17
023720	016767	155266	155266	4\$:	MOV	WK15,WORK	:GET BIT 15 OF CRC
023726	004767	000220		5\$:	JSR	PC,XOR	:XOR BIT15 WITH INPUT BIT
023732	042767	000001	155256		BIC	#BIT0,WORK0	
023740	005767	155244			TST	INBIT	:TEST RESULT OF XOR
023744	001403				BEQ	6\$	
023746	052767	000001	155242		BIS	#BIT0,WORK0	
023754	016767	155230	155174	6\$:	MOV	INBIT,R50	:SAVE XOR SESULT OF BIT 0 AND INPUT

;FROM B0 IN WORK0 AND B1 IN SAVED CRC (WCRC) CALCULATE
;NEW B2 FOR WORK0

023762	005067	155226		CLR	WORK	
023766	032767	000002	155204	BIT	#BIT1,WCRC	
023774	001403			BEQ	7\$	
023776	052767	000001	155210	BIS	#BIT0,WORK	
024004	016767	155146	155176	MOV	RS0,INBIT	
024012	004767	000134		JSR	PC,XXOR	
024016	042767	000004	155172	BIC	#BIT2,WORK0	
024024	005767	155160		TST	INBIT	;TEST RESULT OF XOR
024030	001403			BEQ	8\$	
024032	052767	000004	155156	BIS	#BIT2,WORK0	

;FROM B0 IN WORK0 AND B14 IN WCRC CLACULATE BIT15 IN WORK0

024040	005067	155150		8\$: CLR	WORK	
024044	032767	040000	155126	BIT	#BIT14,WCRC	
024052	001403			BEQ	9\$	
024054	052767	000001	155132	BIS	#BIT0,WORK	
024062	016767	155070	155120	MOV	RS0,INBIT	
024070	004767	000056		JSR	PC,XXOR	
024074	042767	100000	155114	BIC	#BIT15,WORK0	
024102	005767	155102		TST	INBIT	;TEST RESULT OF XOR
024106	001403			BEQ	10\$	
024110	052767	100000	155100	BIS	#BIT15,WORK0	
024116	005367	155062		10\$: DEC	REPT1	;DONE WITH WD
024122	001244			BNE	2\$	;NO
024124	005367	155052		DEC	REPT	;DONE WITH SECTOR?
024130	001404			BEQ	11\$	;YES
024132	062705	000002		ADD	#2,R5	;GET NEXT WD
024136	011504			MOV	(R5),R4	;GET DATA WD
024140	000623			BR	1\$	
024142	016767	155050	155044	11\$: MOV	WORK0,WORK	;SAVE CRC WORD IN WORK
024150	000207			RTS	PC	;EXIT

;XOR SUBROUTINE

024152	016703	155036		XXOR: MOV	WORK,R3	
024156	046703	155026		BIC	INBIT,R3	
024162	046767	155026	155020	BIC	WORK,INBIT	
024170	050367	155014		BIS	R3,INBIT	
024174	000207			RTS	PC	

809

.SBTTL

STYPER - TTY TYPEOUT ROUTINE

; THIS ROUTINE IS USE TO TYPE ASCII MESSAGES ON THE TTY. THE
; CALL CAN BE IN ONE OF 3 FORMS: 1) "TYPE ADR" - TYPES THE
; MESSAGE STARTING IN LOCATION "ADR:" 2) "TYPE CHAR" - TYPES
; THE ASCII "CHAR", AND 3) "PRINT <<15><12>"MESSAGE">" - TYPES
; THE MESSAGE WHICH IS INLINE ASCII. THE FILLER CHARACTER WHICH IS
; TYPED AFTER A LINE FEED IS IN FILCHR AND THE NUMBER OF FILLERS
; IS IN FILCHR+1.

024176	010446			.TYPE:	MOV	R4,-(6)	:SAVE R4
024200	010546				MOV	R5,-(6)	:SAVE R5
024202	017605	000004			MOV	24(6),R5	:GET ADDRESS TO BE TYPED
024206	032705	177400			BIT	#177400,R5	:IS IT A TYPEN?
024212	001002				BNE	1\$	:NO
024214	016605	000004			MOV	4(6),R5	:GET ADDRESS OF CHARACTER
024220	105715			1\$:	TSTB	(R5)	:TERMINATOR?
024222	001423				BEQ	2\$	:GET OUT IF SO
024224	122715	000012			CMPB	#12,(R5)	:IS THE CHAR A LINE FEED
024230	001012				BNE	4\$	:NO - GET OUT
024232	116704	154557			MOVB	FILCHR+1,R4	:GET THE FILL COUNT
024236	116777	154552	154554	5\$:	MOVB	FILCHR,2TPB	:TYPE A FILLER
024244	105777	154546			TSTB	2TPS	:DONE YET?
024250	100375				BPL	.-4	:NO - WAIT
024252	005304				DEC	R4	:DEC COUNT
024254	001370				BNE	5\$	:LOOP UNTIL 0
024256	112577	154536		4\$:	MOVB	(R5)+,2TPB	:LOAD AND TYPE THE CHARACTER
024262	105777	154530			TSTB	2TPS	:IS THE PRINTER READY
024266	100375				BPL	.-4	:WAIT UNTIL IT IS
024270	000753				BR	1\$	:GET THE NEXT CHARACTER
024272	017646	000004		2\$:	MOV	24(6),-(6)	:GET ADDRESS TO BE TYPED
024276	062766	000002	000006		ADD	#2,6(6)	:ADD 2 TO THE ADDRESS
024304	022666	000004			CMP	(6)+,4(6)	:IS IT .+2?
024310	001006				BNE	3\$	:NO
024312	062705	000002			ADD	#2,R5	:ADD 2 TO THE ADDRESS
024316	042705	000001			BIC	#1,R5	:BACK UP TO AN EVEN BYTE
024322	010566	000004			MOV	R5,4(6)	:RESTORE ADDRESS
024326	012605			3\$:	MOV	(6)+,R5	:RESTORE R5
024330	012604				MOV	(6)+,R4	:RESTORE R4
024332	000002				RTI		:RETURN

.SBTTL

SSCOPE - SCOPE LOOP HANDLER

:THIS ROUTINE HANDLES THE ITERATIONS, LOOPING, ERROR
:LOOPING, AND THE DISPLAYING OF THE TEST NUMBER.
:"SCOPE" IS PLACED BETWEEN EACH SUBTEST IN THE TEST AND
:RECORDS THE STARTING ADDRESS OF THE SUBTEST IN "LAD:"

024334	032737	000400	177570	.SCOPE:	BIT	#SW8,2#SWR	:LOOP ON SPEC. TEST?
024342	001404				BEQ	15	:NO LOOP ON SPEC. TEST
024344	123767	177570	154426		CMPB	2#SWR,ICNT	:ON RIGHT TEST? *SW7-0*
024352	001453				BEQ	.OVER	:NOT RIGHT TEST
024354	032737	040000	177570	1\$:	BIT	#SW14,2#SWR	:LOOP ON TEST?
024362	001045				BNE	.KIT	:LOOP ON TEST IS SET
024364	000416				BR	3\$	:SKIP - NOP FOR XOR TESTER
024366	013746	000004			MOV	2#4,-(6)	:PUSH 2#4 ON STACK
024372	012737	024412	000004		MOV	2#4,2#4	:SET FOR TIMEOUT
024400	005737	177060			TST	2#177060	:ERROR ON XOR?
024404	012637	000004			MOV	(6)+,2#4	:POP STACK INTO 2#4
024410	000422				BR	.SVLAD	:NO ERROR - GO TO NEXT TEST
024412	022626			4\$:	CMP	(6)+,(6)+	:CLEAR STACK
024414	012637	000004			MOV	(6)+,2#4	:POP STACK INTO 2#4
024420	000426				BR	.KIT	:ERROR - LOOP ON TEST
024422	032737	004000	177570	3\$:	BIT	#SW11,2#SWR	:KILL ITERATIONS
024430	001012				BNE	.SVLAD	:YES - KILL ITERATIONS
024432	105767	154343			TSTB	ICNT+1	:FIRST ONE?
024436	001404				BEQ	2\$	:BRANCH IF FIRST
024440	126767	000060	154333		CMPB	TIMES,ICNT+1	:DONE?
024446	003013				BGT	.KIT	:BRANCH IF NOT
024450	112767	000001	154323	2\$:	MOVB	#1,ICNT+1	:FIRST ITERATION
024456	105267	154316		.SVLAD:	INCB	ICNT	:COUNT TEST NUMBERS
024462	011667	154322			MOV	(6),LAD	:SAVE LOOP ADDRESS
024466	016737	154306	177570		MOV	ICNT,2#DISPLAY	:DISPLAY TEST NO. AND ITERATION COUNT
024474	000002				RTI		:RETURN
024476	105267	154277		.KIT:	INCB	ICNT+1	:INC THE ITERATION COUNT
024502	016737	154272	177570	.OVER:	MOV	ICNT,2#DISPLAY	:SET UP DISPLAY
024510	005767	154274			TST	LAD	:FIRST ONE?
024514	001760				BEQ	.SVLAD	:YES
024516	016716	154266			MOV	LAD,(6)	:FUDGE RETURN ADDRESS
024522	000002				RTI		:FIXES PS
024524	000001			TIMES:	1		:RUN 1 TIMES

.SBTTL \$HLT - HLT ROUTINE (ERROR TYPEOUT)

;THIS ROUTINE PRINTS OUT ERROR MESSAGES STARTING WITH THE
;ADDRESS OF THE "HLT". IT ALSO COUNTS THE NUMBER OF ERRORS
;AND HAS THE CAPABILITY OF LOOPING ON ERROR, BELL ON ERROR,
;"HALT" ON ERROR, AND INHIBIT TYPEOUTS. AN OPTIONAL ARGUMENT
; (HLT+3) WILL BE PLACED IN ".HLTCT:" FOR ADDITIONAL TYPEOUTS.

024526	032737	002000	177570	.HLT:	BIT	#SW10,2#SWR	:BELL ON ERROR?
024534	001402				BEG	1\$	:NO - SKIP
024536	104402	000007			TYPE	BELL	:RING BELL
024542	005267	154234		1\$:	INC	ERRORS	:COUNT THE NUMBER OF ERRORS
024546	032737	020000	177570		BIT	#SW13,2#SWR	:SKIP TYPEOUT IF SET
024554	001025				BNE	2\$	:SKIP TYPEOUTS
024556	104402	024562			TYPE	.+2	:ASCIZ <15><12>
024566	011667	154220			MOV	{6},HLTADR	:PUT ADDRESS OF INSTRUCT:JN ON STACK
024572	162767	000002	154212		SUB	#2,HLTADR	:FUDGE ADDRESS
024600	117767	154206	000066		MOVB	2HLTADR,.HLTCT	:GET HLT ARGUMENT
024606	016746	154200			MOV	HLTADR,-(5)	:PUT HLTADR ON STACK
024612	104404				TYPE0		:TYPE STACK IN OCTAL
024614	104402	024620			TYPE	.+2	:ASCIZ " "
024624	004767	001152			JSR	PC,RSREG	:GO TO USER ERROR ROUTINE
024630	005737	177570		2\$:	TST	2#SWR	:HALT ON ERROR
024634	100001				BPL	.+4	:SKIP IF CONTINUE
024636	000000				HALT		:HALT ON ERROR!
024640	032737	001000	177570		BIT	#SW9,2#SWR	:CHECK FOR INHIBIT LOOP ON ERROR
024646	001010				BNE	4\$	:SKIP IF LOOP ON ERROR
024650	105067	154125			CLRB	ICNT+1	:CLEAR ITERATION COUNT
024654	022737	021274	000042		CMP	#SENDAD,2#42	:ACT11 AUTO ACCEPT?
024662	001001				BNE	3\$	:NO BR
024664	000000				HALT		
024666	000002			3\$:	RTI		:RETURN
024670	000167	177602		4\$:	JMP	.KIT	:LOOP ON TEST UNTIL NO ERRORS
024674	000000			.HLTCT: 0			:HLT ARGUMENT

E09

.SBTTL SOCTAL - OCTAL TYPEOUT ROUTINE

;THIS ROUTINE IS USED TO TYPE AN OCTAL NUMBER ON THE TTY. IT WILL TYPE
;ALL 6 CHARACTERS, SUPPRESS LEADING ZEROES, OR TYPE THE
;16 BITS. IT IS CALLED VIA THE TYPOCT, TYPBIT, OR TYPOCS MACRO'S.

024676	012767	170101	000160	.TYPEB: MOV	#170101,.PR	:SET BIT FLAG AND 16. CHARACTER COUNT
024704	000411			BR	.PTIT	:NOW TYPE IT IN BIT FORM
024706	112767	000001	000150	.TYPEO: MOV	#1,.PR	:SET ZERO FILL SWITCH
024714	000402			BR	+.6	:SKIP
024716	005067	000142		.TYPES: CLR	.PR	:SUPPRESS LEADING ZERO'S
024722	112767	177772	000135	MOV	#-6,.PR+1	:SET COUNT
024730				.PTIT:		
024730	010446			MOV	R4,-(6)	:PUSH R4 ON STACK
024732	010546			MOV	R5,-(6)	:PUSH R5 ON STACK
024734	016605	000010		MOV	10(6),R5	:GET THE DATA
024740	012704	025066		MOV	#.PR+2,R4	:SET POINTER TO FIRST ASCII CHAR.
024744	105014			CLRB	(4)	:CLEAR FIRST BYTE
024746	000411			BR	.PRF	:ROTATE FIRST BIT
024750	105014			.PRL: CLRB	(4)	:CLEAR BYTE OF CHARACTER
024752	032767	000100	000104	BIT	#100,.PR	:BIT TYPING MODE?
024760	001004			BNE	.PRF	:YES - SKIP 2 ROTATES
024762	006105			ROL	R5	:ROTATE BIT INTO C
024764	106114			ROLB	(4)	:PACK IT
024766	006105			ROL	R5	:ROTATE BIT INTO C
024770	106114			ROLB	(4)	:PACK IT
024772	006105			.PRF: ROL	R5	:ROTATE BIT INTO C
024774	106114			ROLB	(4)	:PACK IT
024776	105714			TSTB	(4)	:IS IT ZERO?
025000	001402			BEQ	+.6	:SKIP INC
025002	105267	000056		INCB	.PR	:SET FILL SWITCH
025006	105767	000052		TSTB	.PR	:CHECK FILL SWITCH
025012	001402			BEQ	+.6	:SKIP BITSET
025014	152724	000060		BISB	#'0,(4)+	:MAKE INTO ASCII CHAR
025020	105267	000041		INCB	.PR+1	:INC COUNT
025024	001351			BNE	.PRL	:REPEAT
025026	022704	025066		CMP	#.PR+2,R4	:EMPTY BUFFER?
025032	001002			BNE	+.6	:SKIP IF NOT
025034	112724	000060		MOV	#'0,(4)+	:LOAD 1 ZERO
025040	105014			CLRB	(4)	:NULL TERMINATOR
025042	104402	025066		TYPE	.PR+2	:TYPE IT
025046	012605			MOV	(6)+,R5	:POP STACK INTO R5
025050	012604			MOV	(6)+,R4	:POP STACK INTO R4
025052	016666	000002	000004	MOV	2(6),4(6)	:GET RID OF
025060	012616			MOV	(6)+,(6)	:DATA WORD
025062	000002			RTI		:RETURN
025064	000012			.PR: .BLKW	12	:COUNT, SWITCH, AND OUTPUT BUFFER

F09

.SBTTL

\$POWER - POWER DOWN AND UP ROUTINES

; THIS IS THE POWER FAIL ROUTINE WHICH WILL SAVE ALL
; THE GENERAL REGISTERS AND USER DEFINED REGISTERS THEN
; WAIT FOR POWER TO GO DOWN AND BE RESTORED.
; IF THERE ISN'T ENOUGH TIME FOR SAVING ALL THE REGISTERS,
; THE PROGRAM WILL HALT AT '.ILLUP'.

025110	012777	025236	000126	.POWER:	MOV	#.ILLUP,2.PUVEC	;SET FOR FAST UP
025116	012777	000340	000122		MOV	#340,2.PUVECS+2	;PRIO:7
025124	010046				MOV	R0,-(6)	;PUSH R0 ON STACK
025126	010146				MOV	R1,-(6)	;PUSH R1 ON STACK
025130	010246				MOV	R2,-(6)	;PUSH R2 ON STACK
025132	010346				MOV	R3,-(6)	;PUSH R3 ON STACK
025134	010446				MOV	R4,-(6)	;PUSH R4 ON STACK
025136	010546				MOV	R5,-(6)	;PUSH R5 ON STACK
025140	010667	000076			MOV	SP,SAVR6	;SAVE SP
025144	012777	025154	000072		MOV	#.POWUP,2.PUVEC	;SET UP VECTOR
025152	000000				HALT		;WAIT FOR PF
025154	016706	000062		.POWUP:	MOV	.SAVR6,SP	;GET SP
025160	005001				CLR	R1	;WAIT LOOP FOR THE TTY
025162	005201			1\$:	INC	R1	;WAIT FOR THE INC
025164	001376				BNE	1\$	;OF WORD
025166	012605				MOV	(6)+,R5	;POP STACK INTO R5
025170	012604				MOV	(6)+,R4	;POP STACK INTO R4
025172	012603				MOV	(6)+,R3	;POP STACK INTO R3
025174	012602				MOV	(6)+,R2	;POP STACK INTO R2
025176	012601				MOV	(6)+,R1	;POP STACK INTO R1
025200	012600				MOV	(6)+,R0	;POP STACK INTO R0
025202	012737	025110	000024		MOV	#.POWER,2#24	;SET UP THE POWER DOWN VECTOR
025210	012737	000340	000026		MOV	#340,2#26	;PRIO:7
025216	104402	025222			TYPE	.+2	;ASCIZ <15><12>"POWER"
025232	000167	174054			JMP	MULSYS	;JMP TO USER ADDRESS
025236	000000			.ILLUP:	HALT		;THE POWER UP SEQUENCE WAS STARTED
025240	000776				BR	.-2	;BEFORE THE POWER DOWN WAS COMPLETE
025242	000000			.SAVR6:	0		;PUT THE SP HERE
025244	000024	000026		.PUVEC:	24,26		;POWER UP VECTOR

.SBTTL

SRDOCT - OCTAL INPUT ROUTINE

:THIS ROUTINE CALLS RDLIN, INPUTS A LINE FROM THE TTY AND CONVERTS
:IT INTO AN OCTAL NUMBER WHICH IS THE FIRST WORD ON THE STACK.

025250	011646		.RDOCT:	MOV	(6),-(6)	:MOVE THE PC
025252	016666	000004 000002		MOV	4(6),2(6)	:MOVE THE PS
025260	010146			MOV	R1,-(6)	:PUSH R1 ON STACK
025262	010246			MOV	R2,-(6)	:PUSH R2 ON STACK
025264	010346			MOV	R3,-(6)	:PUSH R3 ON STACK
025266	104412		4\$:	RDLIN		:READ A LINE INTO INPUT
025270	005001			CLR	R1	:INIT DATA WORD
025272	012703	025472		MOV	#INPUT,R3	:INIT POINTER
025276	112302		1\$:	MOVB	(3)+,R2	:GET A BYTE
025300	001417			BEQ	2\$	:GET OUT IF ZERO
025302	122702	000060		CMPB	#'0,R2	:CHECK FOR 0 OR GREATER
025306	003022			BGT	3\$	:ERROR - LESS THAN 0
025310	122702	000067		CMPB	#'7,R2	:CHECK FOR 7 OR LESS
025314	002417			BLT	3\$	:ERROR - GREATER THAN 7
025316	006002			ROR	R2	:GET
025320	006002			ROR	R2	:INTO
025322	006002			ROR	R2	:POSITION
025324	006101			ROL	R1	:FIRST BIT
025326	006102			ROL	R2	:GET
025330	006101			ROL	R1	:SECOND BIT
025332	006102			ROL	R2	:GET
025334	006101			ROL	R1	:THIRD BIT
025336	000757			BR	1\$	:LOOP
025340	010166	000012	2\$:	MOV	R1,12(6)	:SAVE THE RESULT
025344	012603			MOV	(6)+,R3	:POP STACK INTO R3
025346	012602			MOV	(6)+,R2	:POP STACK INTO R2
025350	012601			MOV	(6)+,R1	:POP STACK INTO R1
025352	000002			RTI		:RETURN
025354			3\$:			
025354	104402	025360		TYPE	4\$+2	:ASCIZ "?"(15)<12>
025364	000740			BR	4\$	:TRY AGAIN

.SBTTL

SRDLIN - TTY INPUT ROUTINE

; THIS ROUTINE INPUTS A LINE TERMINATED BY A RETURN INTO ADDRESS
; INPUT AND RETURNS A LINE FEED. THE BUFFER HAS A NULL TERMINATOR
; INSTEAD OF THE RETURN. RUBOUTS ARE HANDLED BY RETYPING
; THE LINE. BUFFER OVERFLOW ERRORS LIKE A RUBOUT.

025366	010546	
025370	012705	025472
025374	022705	025512
025400	001412	
025402	105737	177560
025406	100375	
025410	113715	177562
025414	142715	000200
025420	122715	000177
025424	001005	
025426		
025426	104402	025432
025436	000754	
025440	111527	000000
025444	104402	025442
025450	122725	000015
025454	001347	
025456	105065	177777
025462	104402	000012
025466	012605	
025470	000002	
025472	000020	

RD LIN:	MOV	R5, -(6)	:SAVE R5
1\$:	MOV	#INPUT, R5	:GET ADDRESS
2\$:	CMP	#INPUT+16., R5	:BUFFER FULL?
	BEQ	4\$	:YES - TYPE "?"
	TSTB	3#177560	:WAIT FOR
	BPL	-4	:A CHARACTER
	MOVB	3#177562, (5)	:GET CHARACTER
	BICB	#200, (5)	:GET RID OF JUNK
	CMPB	#177, (5)	:IS IT A RUBOUT
	BNE	3\$	:SKIP IF NOT
4\$:	TYPE	.+2	:ASCIZ "?"(15)(12)
	BR	1\$	:ZAP THE BUFFER AND LOOP
3\$:	MOVB	(5), #0	:SET UP FOR TYPING
	TYPE	3\$+2	:ECHO IT
	CMPB	#15, (5)+	:CHECK FOR RETURN
	BNE	2\$	:LOOP IF NOT RETURN
	CLRB	-1(5)	:ZAP RETURN (THE 15)
	TYPE	12	:TYPE A LINE FEED
	MOV	(6)+, R5	:RESTORE R5
	RTI		:RETURN
INPUT:	.BLKB	16.	:TTY INPUT AREA

.SBTTL \$TRAP - TRAP HANDLER

:THIS ROUTINE DECODES A TRAP CALL AND JUMPS TO THE APROPRATE
:SUBROUTINE. THE CALL IS A "TRAP+N" WHERE N IS A MULTIPLE OF 2.
:THE "SET" MACRO WILL CREATE THE TABLE NEEDED. IT HAS TO
:FOLLOW THIS MACRO.

025512 011646
025514 162716 000002
025520 017616 000000
025524 062716 121132
025530 013607

.TRAP: MOV (6),-(6) :GET ADDRESS OF TRAP +2
SUB #2,(6) :MAKE IT ADDRESS OF TRAP
MOV #2,(6) :GET TRAP INSTRUCTION
ADD #.TRAP+2-TRAP,(6) :GET DATA AND MAKE IT AN OFFSET
.TRP: MOV #2,(6)+,PC :GO TO PROPER SUBROUTINE

025532	024334	.SCOPE	:SCOPE	=	TRAP+0	(104400)
025534	024176	.TYPE	:TYPE	=	TRAP+2	(104402)
025536	024706	.TYPE0	:TYPE0	=	TRAP+4	(104404)
025540	024716	.TYPES	:TYPES	=	TRAP+6	(104406)
025542	025250	.RDOCT	:RDOCT	=	TRAP+10	(104410)
025544	025366	.RDLIN	:RDLIN	=	TRAP+12	(104412)
025546	025604	.CLRDK	:CLRDK	=	TRAP+14	(104414)
025550	025632	.MRDMD	:MRDMD	=	TRAP+16	(104416)
025552	021634	.MRCK	:MRCK	=	TRAP+20	(104420)
025554	021604	.MRCLK	:MRCLK	=	TRAP+22	(104422)
025556	021576	.MRINT	:MRINT	=	TRAP+24	(104424)
025560	021700	.DSCK	:DSCK	=	TRAP+26	(104426)
025562	021662	.MRIND	:MRIND	=	TRAP+30	(104430)
025564	021726	.XBIT	:XBIT	=	TRAP+32	(104432)
025566	022126	.CLKD1	:CLKD1	=	TRAP+34	(104434)
025570	022222	.CLKD2	:CLKD2	=	TRAP+36	(104436)
025572	023104	.CLKR1	:CLKR1	=	TRAP+40	(104440)
025574	023334	.CLKR2	:CLKR2	=	TRAP+42	(104442)
025576	022734	.RBIT	:RBIT	=	TRAP+44	(104444)
025600	025724	.GETSP	:GETSP	=	TRAP+46	(104446)
025602	025764	.SPASS	:SPASS	=	TRAP+50	(104450)

J09

```
025604 012777 000040 153270 :CLEAR ALL DISK REGISTERS
025612 016777 153342 153262 .CLRDK: MOV #40,DRSCS2 :CLEAR ALL DSK REG
025620 005067 153350 :MOV UNNUM,DRSCS2 :GET UNIT NUMBER
025624 005067 153346 :CLR MCCNT :CLEAR MAINT CLOCK COUNT
025630 000002 :CLR MCCNT+2

025632 012777 000001 153264 .MRDMD: MOV #1,DRSMR :PUT DRIVE INTO MAINT MODE
025640 000002 :RTI

025642 005067 153346 WAITRY: CLR WORK :CLEAR COUNTER
025646 105777 153226 1$: TSTB DRSCS1 :TEST READY
025652 100406 :BMI 2$ :OK CONT
025654 005267 153334 :INC WORK :UPDATE COUNTER
025660 005767 153330 :TST WORK :DONE YET?
025664 001403 :BEQ 3$ :READY DID NOT COME UP
025666 000767 :BR 1$ :CONTINUE WAITING
025670 062716 000002 2$: ADD #2,(SP) :UPDATE RETURN PC
025674 000207 3$: RTS PC :RETURN

:ROUTINE TO SHIFT COMPLETE DATA TABLE ONE BIT
:TO THE LEFT. CARRIES BIT 15 OF ONE WORD TO BIT 0 OF THE NEXT WORD

025676 012702 026716 MDATA: MOV #INBUF,R2 :GET LEFT ADDRESS OF
025702 062702 000442 :ADD #442,R2 :DATA TABLE
025706 012703 000220 :MOV #220,R3 :SETUP COUNTER FOR 200 WORDS
025712 000241 :CLC :CLEAR CARRY
025714 006142 1$: ROL -(R2) :SHIFT DATA PATTERN
025716 005303 :DEC R3 :DO ALL
025720 001375 :BNE 1$ :WORDS
025722 000207 :RTS PC

:THIS ROUTINE CLOCKS MR REG TO GET A SECTOR PULSE WHICH
:CLEARS OUT REGS. AND COUNTERS

025724 012767 002001 153250 .GETSP: MOV #1025.,REPT :SETUP COUNTER
025732 104430 :MRIND :SEND INDEX PULSE TO MR REG
025734 104422 1$: MRCLK :CLOCK MR
025736 005367 153240 :DEC REPT :TO REACH
025742 001374 :BNE 1$ :SECTOR PULSE
025744 032777 000400 153152 :BIT #400,DRSMR :DID SECTOR PULSE SET????
025752 001401 :BEQ 2$ :YES
025754 000002 :RTI :NO REPORT ERROR
025756 062716 000002 2$: ADD #2,(SP) :UPDATE RETURN ADDR
025762 000002 :RTI

025764 104422 .SPASS: MRCLK :CLOCK PAST SECTOR PULSE
025766 104422 :MRCLK :RESET MAINT CLOCK COUNTERS
025770 005067 153200 :CLR MCCNT
025774 005067 153176 :CLR MCCNT+2
026000 000002 :RTI
```


: SHOULD WE TYPTXT GOOD AND BAD

Address	Offset	Value	Register	Operation	Comment
026002	005767	176666	RSREG:	TST	..HLTCT
026006	001022			BNE	8\$
026010	104402	026014		TYPE	..+2
026022	010046			MOV	BAD,-(6)
026024	104404			TYPE0	
026026	104402	026032		TYPE	..+2
026042	010146			MOV	GOOD,-(6)
026044	104404			TYPE0	
026046	000402			BR	8\$
026050	000167	000432		JMP	PTDONE
026054			8\$:		
026054	104402	026060		TYPE	..+2
026066	017746	153006		MOV	ARSCS1,-(6)
026072	104404			TYPE0	
026074			1\$:		
026074	104402	026100		TYPE	..+2
026106	017746	153002		MOV	ARSER,-(6)
026112	104404			TYPE0	
026114			2\$:		
026114	104402	026120		TYPE	..+2
026126	017746	152750		MOV	ARSCS2,-(6)
026132	104404			TYPE0	
026134	032767	000200	176532	BIT	#200, .HLTCT
026142	001076			BNE	SEEC
026144	032767	000100	176522	BIT	#AS, .HLTCT
026152	001410			BEQ	3\$
026154	104402	026160		TYPE	..+2
026166	017746	152724		MOV	ARSAS,-(6)
026172	104404			TYPE0	
026174	032767	000020	176472	3\$:	BIT
026202	001410			BEQ	4\$
026204	104402	026210		TYPE	..+2
026216	017746	152664		MOV	ARSDA,-(6)
026222	104404			TYPE0	
026224	032767	000004	176442	4\$:	BIT
026232	001410			BEQ	5\$
026234	104402	026240		TYPE	..+2
026246	017746	152636		MOV	ARSDA,-(6)
026252	104404			TYPE0	
026254	032767	000010	176412	5\$:	BIT
026262	001410			BEQ	6\$
026264	104402	026270		TYPE	..+2
026276	017746	152602		MOV	ARSWC,-(6)
026302	104404			TYPE0	
026304	032767	000040	176362	6\$:	BIT
026312	001475			BEQ	PTDONE
026314	104402	026320		TYPE	..+2

```
026326 017746 152560      MOV      ARSDS,-(6)      ;PUT ARSDS ON STACK
026332 104404              TYPE0      ;TYPE STACK IN OCTAL
026334 000167 000146      JMP      PTDONE      ;GET OUT
026340 042767 000200 176326 SEEC:    BIC      #200,.HLTCT      ;CLEAR COMMON BIT
026346 032767 000240 176320        BIT      #DT,.HLTCT      ;TYPTXT DRIVE TYPE?
026354 001410              BEQ      9$      ;NO
026356 104402 026362      TYPE      +2      ;ASCIZ "DT="
026370 017746 152532      MOV      ARSDT,-(6)      ;PUT ARSDT ON STACK
026374 104404              TYPE0      ;TYPE STACK IN OCTAL
026376 032767 000210 176270 9$:    BIT      #DB,.HLTCT      ;TYPTXT DATA BUFFER
026404 001410              BEQ      10$      ;NO
026406 104402 026412      TYPE      +2      ;ASCIZ "DB="
026420 017746 152476      MOV      ARSDB,-(6)      ;PUT ARSDB ON STACK
026424 104404              TYPE0      ;TYPE STACK IN OCTAL
026426 032767 000220 176240 10$:   BIT      #MR,.HLTCT      ;TYPTXT MN?
026434 001410              BEQ      11$      ;NO
026436 104402 026442      TYPE      +2      ;ASCIZ "MR="
026450 017746 152450      MOV      ARSMR,-(6)      ;PUT ARSMR ON STACK
026454 104404              TYPE0      ;TYPE STACK IN OCTAL
026456 032767 000204 176210 11$:   BIT      #LA,.HLTCT      ;TYPTXT LA?
026464 001410              BEQ      PTDONE      ;NO
026466 104402 026472      TYPE      +2      ;ASCIZ "LA="
026500 017746 152414      MOV      ARSLA,-(6)      ;PUT ARSLA ON STACK
026504 104404              TYPE0      ;TYPE STACK IN OCTAL
026506 052767 100000 152452 PTDONE: BIS      #BIT15,ONCEE      ;SET FORND ERROR FLAG
026514 032767 000040 152444        BIT      #BIT5,ONCEE
026522 001466              BEQ      1$      ;
026524 104402 026530      TYPE      +2      ;ASCIZ <15><12>"MAINT CLOCK COUNT "
026556 016767 152412 152442      MOV      MCCNT,WORK4      ;GET MAINT CLOCK COUNT
026564 016767 152406 152430      MOV      MCCNT+2,WORK2      ;CAL NUMBERS FOR DOUBLE PRECISION
026572 006167 152424      ROL      WORK2
026576 006167 152424      ROL      WORK4
026602 000241              CLC
026604 016746 152416      MOV      WORK4,-(6)      ;PUT WORK4 ON STACK
026610 104406              TYPES      ;TYPE STACK IN OCTAL - SUPRESS
026612 012767 000005 152410      MOV      #5,WORK5
026620 005067 152406      2$:      CLR      WORK6
026624 006167 152372      ROL      WORK2
026630 006167 152376      ROL      WORK6
026634 006167 152362      ROL      WORK2
026640 006167 152366      ROL      WORK6
026644 006167 152352      ROL      WORK2
026650 006167 152356      ROL      WORK6
026654 016746 152352      MOV      WORK6,-(6)      ;PUT WORK6 ON STACK
026660 104406              TYPES      ;TYPE STACK IN OCTAL - SUPRESS
026662 005367 152342      DEC      WORK5
026666 001354              BNE      2$
026670 104402 026674      TYPE      +2      ;ASCIZ <15><12>
026700 023737 000042 000046 1$:    CMP      #42,#46      ;ACT11?
026706 001001              BNE      3$      ;BR IF NO
026710 000000              HALT
026712 000207      3$:      RTS      PC
026714 000000      FUDGE:    .WORD    0      ;THIS WORD IS INSERTED TO INSURE INBUF
                                           ;STARTS ON BOUNDARY OF 4,10 SEE AIDS PR#2507.

026716 000300      INBUF:    .BLKW    300
```


ZZ-CERSD-D
CERSDD.P11

RH70-R504 MAINTENANCE MODE DIAGNOSTIC
23-JAN-78 13:36

MACY11 30A(1052)
STRAP - TRAP HANDLER

23-JAN-78 13:38 PAGE 148-1

SEQ 0116

027516 000300
000001

OUTBUF: .BLKW 300
.END

M09

AOECK 020414
AS = 000100
ATA = 100000
BA = 000020
BAD = %000000
BAI = 000010
BEGIN 001234
BEGIN1 000232
BEGIN2 000226
BELL = 000007
BITBA 004520
BITCS2 004404
BITST 004332
BITWC 004450
BIT0 = 000001
BIT1 = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2 = 000004
BIT3 = 000010
BIT4 = 000020
BIT5 = 000040
BIT6 = 000100
BIT7 = 000200
BIT8 = 000400
BIT9 = 001000
CALRTB 023346
CHKCDV 002236
CHG 021460
CLKCNT 001206
CLKD1 = 104434
CLKD2 = 104436
CLKR1 = 104440
CLKR2 = 104442
CLODK 023114
CLDK = 104414
CRCTAB 022424
CRCTYP 022314
CRCWD 023530
CS1 = 000001
CS2 = 000200
DA = 000004
DAO = 001000
DB = 000210
DCK = 100000
DISPLA = 177570
DLT = 100000
DONE 021230
DNEE 002412

DRY = 000200
DS = 000040
DSCK = 104426
DT = 000240
DVNUM 002016
ER = 000002
ERR = 040000
ERRORS 001002
E1910 022716
E1915 022725
E192 022702
E197 022710
E2410 022655
E2412 022664
E2415 022673
E242 022633
E245 022641
E247 022647
E3010 022606
E3012 022615
E3015 022624
E302 022564
E305 022572
E307 022600
FILCHR 001014
FLAG2 001144
FLOTBA 003442
FLOTDA 003742
FLOTWC 003602
FUDGE 026714
GENCRC 023552
GETSP = 104446
GOOD = %000001
HLT = 104000
HLTADR 001012
IADONE 016542
ICNT 001000
IE = 000100
ILFDOH 007030
INBIT 001210
INBUF 026716
INFTST 021224
INPUT 025472
INT 004700
INTDON 004770
INTMR 020120
INTMR1 020170
IR = 000100
LA = 000204
LAD 001010
LBT = 002000
LSTEV 001146
LSTOD 001150

LSTWD 023474
MCCNT 001174
MDATA 025676
MPRO = 172100
MR = 000220
MRAOE 020174
MRCAL 023404
MRCILF 006526
MRCK = 104420
MRCLK = 104422
MRCRC 014324
MRCRC1 014530
MRCRC2 014604
MRCRC3 014706
MRCRC4 015066
MRCRC5 015132
MRDCK 015204
MRDCK1 015410
MRDCK2 015464
MRDCK3 015566
MRDCK4 015746
MRDCK5 016012
MRDMD = 104416
MRDSEL 011232
MRD3 013040
MRD4 013240
MRD5 013304
MRD6 013342
MREX 017206
MREX1 017402
MREX2 017456
MREX3 017542
MRIFT 016126
MRILF 006310
MRIND = 104430
MRINT = 104424
MRLF1 006334
MRLF2 006346
MRLF3 006746
MROP 007032
MROPER 007222
MROPI 016544
MROPIA 016736
MRPAR 017060
MRRD 012472
MRRD1 012662
MRRD2 012736
MRT1 010012
MRSRC 007746
MRSRCH 007564
MRTIME 004772
MRTIM1 005040
MRT2 005116

MRT2A 005126
MRT2B 005162
MRT2C 005216
MRT3 005334
MRT4 005526
MRT4A 005600
MRT4B 005656
MRT4C 005710
MRT4D 005762
MRT4E 005770
MRT4F 006210
MRT4G 006242
MRVR 020500
MRVRR 021010
MRVR1 020640
MRVR2 021014
MRWCK 013460
MRWCK1 013634
MRWCK2 013710
MRWCK3 014012
MRWCK4 014212
MRWCK5 014256
MRWRT 011552
MRWRT1 011720
MRWRT2 011774
MRWRT3 012060
MRWR1 010162
MULSYS 021312
MULTII 001766
N = 000065
NED = 010000
NEDDON 011452
NNDD 011542
NOMI 000214
NOMI1 000220
NOPERR 021364
NOWEV 001152
NOWGO 002416
NOWOD 001154
ONCEE 001166
OR = 000200
OUTBUF 027516
PCNT 001004
PGE = 002000
PGTRAP 004754
PIP = 020000
PS = 177776
PSW = 177776
PTOONE 026506
QG = 000001
RBIT = 104444
RDLIN = 104412
RDOCT = 104410

REGCHG 021474
REPT 001202
REPT1 001204
RMRC1 010304
RMRC2 010470
RMRC3 010644
RMRC4 011016
RSAS 001116
RSBA 001106
RSBAB 001142
RSCS1 001100
RSCS1B 001134
RSCS2 001102
RSCS2B 001136
RSDA 001110
RSDB 001122
RSDS 001112
RSDT 001126
RSEB 001114
RSLA 001120
RSMR 001124
RSREG 026002
RSVCPS 001132
RSVEC 001130
RSC 001104
RSLCB 001140
RSO 001156
SAVEE 001172
SC = 100000
SCOPE = 104400
SEEC 026340
SPASS = 104450
STTEST 002102
SWR = 177570
SW10 = 002000
SW11 = 004000
SW12 = 010000
SW13 = 020000
SW14 = 040000
SW15 = 100000
SW8 = 000400
SW9 = 001000
TBDIA 020726
TIMES 024524
TIMSV 001170
TPB 001020
TPS 001016
TRE = 040000
TRMR 021531
TRY 001776
TRYNX 002212
TSTEVB 022152
TST1 002440

TST10	003472	TST36	005524	TST63	020172	XXOR	024152	.PR	025064
TST11	003516	TST37	006306	TST64	020476	ZERONE	004572	.PRF	024772
TST12	003600	TST4	003170	TST7	003440	SENDAD	021274	.PRL	024750
TST13	003632	TST40	007030	TTAGG	002664	SEND1	021304	.PTIT	024730
TST14	003656	TST41	007220	TYPE =	104402		= 030316	.PUVEC	025244
TST15	003740	TST42	007562	TYPE0 =	104404	.CLKD1	022126	.QQ =	000052
TST16	003772	TST43	007744	TYPES =	104406	.CLKD2	022222	.RBIT	022734
TST17	004016	TST44	010302	UNCMP	001164	.CLKR1	023104	.RDLIN	025366
TST2	002662	TST45	010466	UNITSV	001162	.CLKR2	023334	.RDOCT	025250
TST20	004060	TST46	010642	UNNUM	001160	.CLRDK	025604	.SAVR6	025242
TST21	004102	TST47	011014	WAITRY	025642	.DSCK	021700	.SCOPE	024334
TST22	004124	TST5	003216	WC =	000010	.GETSP	025724	.SPASS	025764
TST23	004150	TST50	011230	WCE =	040000	.HLT	024526	.SVLAD	024456
TST24	004206	TST51	011542	WCRC	001200	.HLTCT	024674	.SWOPT =	163400
TST25	004234	TST52	012470	WK15	001212	.ILLUP	025236	.TBIT	021310
TST26	004272	TST53	013456	WORK	001214	.KIT	024476	.TRAP	025512
TST27	004330	TST54	014314	WORK0	001216	.MRCK	021634	.TRP	025530
TST3	003076	TST55	015174	WORK1	001220	.MRCLK	021604	.TYPE	024176
TST30	004402	TST56	016124	WORK2	001222	.MRDMD	025632	.TYPEB	024676
TST31	004446	TST57	016344	WORK3	001224	.MRIND	021662	.TYPE0	024706
TST32	004516	TST6	003356	WORK4	001226	.MRINT	021576	.TYPES	024716
TST33	004570	TST60	016542	WORK5	001230	.OVER	024502	.XBIT	021726
TST34	004676	TST61	017056	WORK6	001232	.POWER	025110		
TST35	004770	TST62	017176	XBIT =	104432	.POWUP	025154		

. ABS. 030316 000

ERRORS DETECTED: 0

CERSDD, CERSDD.SEQ/NL: SEQ=CERSDD.SML, CERSDD.P11

RUN-TIME: 10 16 .4 SECONDS

RUN-TIME RATIO: 569/27=20.5

CORE USED: 21K (41 PAGES)

