

RX11

RX11 INTFC
CZRxBF0

AH 9341F MC

COPYRIGHT 74-79

FICHE 1 OF 1

NOV 1979

digital

MADE IN USA

PRODUCT CODE: AC-9339F-MC
PRODUCT NAME: CZRXBFO RX11 INTERFACE DIAGNOSTIC
DATE: APRIL 1979
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: DAVID L. ADAMS
MODIFIED BY MIKE PAGE TO REV F

Copyright (C) 1974, 1979
Digital Equipment Corporation, Maynard, Mass.

This software is furnished under a license for use only on a single computer system and may be copied only with the inclusion of the above copyright notice. This software, or any other copies thereof, may not be provided or otherwise made available to any other person except for use on such system and to one who agrees to these license terms. Title to and ownership of the software shall at all times remain in DEC.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

CONTENTS

- 1.0 GENERAL PROGRAM INFORMATION
 - 1.1 Abstract
 - 1.2 System Requirements
 - 1.2.1 Hardware
 - 1.2.2 Software
- 2.0 OPERATING INSTRUCTIONS
 - 2.0.1 Outline of Operating Procedure
 - 2.1 Loading Procedure
 - 2.2 Starting Addresses
 - 2.3 Operator Action Before Starting Program
 - 2.3.1 Device Address Selection
 - 2.3.2 Non-Standard Diskette Address Selection
 - 2.3.3 Software Switch Register (Loc. 176)
 - 2.3.4 Test Parameter Selection ('DTESTP' Loc. 1212)
 - 2.3.4.1 Prerequisites of Tests
 - 2.4 Operator Action to Run the Program
 - 2.4.1 Starting the Program
 - 2.4.2 Operating Conditions
 - 2.5 Test Definitions
 - 2.5.1 Pretest
 - 2.5.2 Test 1 - RXCS Test Part I /
Interrupt Test Part I
 - 2.5.3 Test 2 - Interrupt Test Part II /
Vector Address Verification
 - 2.5.4 Test 3 - Interrupt Test Part III /
Priority Level Verification Part I
 - 2.5.5 Test 4 - Interrupt Test Part IV /
Priority Verification Part II
 - 2.5.6 Test 5 - Init (Programed) / RST
 - 2.5.7 Test 6 - Fill Buffer Transfer Length Verification
 - 2.5.8 Test 7 - Empty Buffer Transfer Length and
Content Verification Part I
 - 2.5.9 Test 10 - Empty Buffer Transfer Length and
Content Verification Part II
 - 2.5.10 Test 11 - Fill / Empty Buffer All 0's
 - 2.5.11 Test 12 - Fill / Empty Buffer All 1's
 - 2.5.12 Test 13 - Drive Ready Verification
 - 2.5.13 Test 14 - Error Flag and B-Code Verification Part I
 - 2.5.14 Test 15 - Error Flag and B-Code Verification Part II
/Deleted Data Bit Sets
 - 2.5.15 Test 16 - Error Flag and B-Code Verification Part III
/Deleted Data Bit Clears
 - 2.5.16 Test 17 - Illegal Track Error and B-Code Verification
 - 2.5.17 Test 20 - Seek Verification Via Read Function
 - 2.5.18 Test 21 - Write Test
 - 2.5.19 Test 22 - Initialize Implied Read
 - 2.5.20 Test 23 - Read Test

- 2.5.21 Test 24 - Data Transfer and Verification
- 2.5.22 Test 25 - Data Transfer and Verification
/Via Deleted Data Mode
- 2.5.23 Test 26 - Head "Home" Test

3.0 ERRORS

- 3.1 Error Heading for Tests 1 - 17, 21 - 23
- 3.2 Error Output Per Test
- 3.3 Error Heading for Test 20, 24 - 26
 - 3.3.1 No Error Flag Errors
 - 3.3.2 Error Flag Errors
 - 3.3.3 Errors Resulting from Previous Errors
 - 3.3.4 Definitive Error Codes
- 3.4 Program Hung

4.0 HALTS

5.0 FLOW CHARTS

1.0 GENERAL PROGRAM INFORMATION

1.1 Abstract

The RX11 interface diagnostic consists of a series of selectable tests that may be run individually, sequence through all tests, or start at a selected test and run through remaining tests, in order, then go back to the selected test.

These tests check out the basic functions of the RX11 interface such as:

1. Done flag
2. Interrupt level / address
3. Program initialize
4. Read status registers
5. Fill / empty buffer transfer verification
6. Fill / empty buffer with data patterns

It is necessary to insure that these functions work before a data reliability test is run.

Any errors are reported by the program, and it is possible to loop on the error or a particular test for scope testing.

1.2 System Requirements

1.2.1 Hardware Requirements

The following equipment is required:

1. PDP-11 series computer with minimum of 8k memory
2. RX11 floppy disk system, including a single or dual drive RX01 and a PDP-11 Interface card [m7846].

NOTE

A diskette must be included with each drive tested.

3. Console teleprinter

1.2.2 Software Requirements

No prerequisite software

2.0 OPERATING INSTRUCTIONS

2.0.1 Outline of Operating Procedure

The standard running procedure for the diagnostic (to run all tests on both drives with no operator intervention via the switch register) is as follows:

1. Load the program into memory
 1. If it is being loaded from a diskette replace the "library" diskette with a "scratch" diskette.

NOTE

If this step is forgotten and the program was loaded via RXDP (floppy monitor) on unit 0 with unit 0 selected by user to undergo testing the program will failsafe the operation and prompt the user as follows: "Caution - If you desire to test unit 0 replace load medium with a scratch diskette then press continue"

Caution again, however -----

NOTE

When running this program on a Small 11 (e.g /04, LSI 11, etc.) Where there is no console switch register it is imperative to remember this setup.

NOTE

Before proceeding to Step B, ensure that the following modifiable locations contain the parameters you require for testing. The following table describes each location with respect to the default parameters which will be used if left unmodified by the user:

LOCATION	LABEL	CONTENTS	PROGRAM REACTION
1200	OD:	0	TRACKS 0,52,53,114(8)
1202	FIRST:	015001	SECTORS 1 THRU 32(8)
1204	KRXVEC:	264	ASSUMES PROPER DEVICE VECTOR
1206	RXCS:	177170	ASSUMES PROPER DEVICE STATUS REGISTER (CALCULATES 'RXDB' ADDRESS FROM)
1212	DTESTP:	0	TESTS BOTH UNITS AUTOMATICALLY SEQUENCES THRU ALL TESTS
1214	BRLEV:	5	ASSUMES PROPER DEVICE 'BR' LEVEL

Reference section 2 of this document for a more thorough description of each of these items and how to modify these locations if you desire to change the above mentioned default testing parameters.

2. Start the program at location 200.
3. The program will type out maindec number, a test parameter of 0 (use both drives and run all tests). Then type tracks to be accessed and sector limits. The program is now running all tests in sequence.
4. If there are no errors, at the end of the pass (approx. 50 seconds run time), a "D" will be typed and it will continue on for another pass.
5. To halt the test at any time (after or before completion of a pass) just halt the processor.
6. After completing a pass of the diagnostic, the RX11 reliability test may be run.
7. There are two types of error print out formats:
 1. Tests pretest, 1 - 17, and 21 - 23 use the format shown in section 3.1. The important address there is the "ERADR" (Error Address) go to the listing at that

location to get more information on the error condition

2. Test 20, and 24 - 26 use the format shown in section 3.3. In this case the "TEST PC" is the address of the test being run when the error occurred. Then the vital information of the error is printed (contents of all registers, address of where on the diskette the error occurred, and the type of error).

2.1 Loading The Program

Load the program into memory using the standard procedure for binary paper tapes. Make sure the total system is ready for operation. The diskettes inserted properly, doors closed on drives to be tested etc.

2.2 Starting Addresses

The program has two starting address locations as follows:

2.2.1 Initial Start [Loc.200]

This starting address tests for and selects the hardware, or software switch register, prints maindec name and revision, the test and drive selection, and tracks and sectors being used.

2.2.2 Restart [Loc.202]

This starting address directs the program to continue running using the drive and test selections specified in the previous initial start.

2.3 Operator Action Before Starting The Program

2.3.1 Device Address Selection

Like most options on the PDP-11 the RX11 Interface Card has jumperable register and vector addresses. This allows for devices with the same standard addresses to be jumpered to an other address so they will run without conflict.

The program must know what addresses are being used, as it is through these register and vector addresses that all communication between the

PDP-11 and the RX11 is handled.

If the RX11 system under test is jumpered for register addresses other than standard, which is RXCS = 177170 and RXDB = 177172 place in the memory location called 'RXCS' [Loc.1206] its new address. The program assumes the next even address above that of RXCS, will be the address of RXDB, so setting that address is not necessary. If there is a nonstandard interrupt vector address (standard is Loc. 264) then place in memory location called 'KRXVEC' [Loc. 1204] its new address.

If either of these locations is loaded with a wrong address, the program will get unpredictable errors and may halt.

NOTE

The program expects that the priority level jumpers are set for a normal 'BR' level of 5 (contents of program location 'BRLEV:' is set to 5). If the priority level jumpers are set to any other level tests 3 & 4 will report errors, unless program location 'BRLEV:' has been patched to contain the relevant 'BR' level before executing the program.

If this is being tested on a LSI 11, tests 3 and 4 will not be run as the LSI 11 has only 1 level of interrupt.

2.3.2 Non-Standard Diskette Address Selection

If it is desirable to test the diskette between track and sector address limits other than the preselected track addresses, and/or minimum (first) and maximum (last) sector addresses, this is done by the operator making changes to two memory locations before the program is started. One location is called 'OD' [Loc. 1200] which contains the two bytes for inner and outer track addresses. The other location is called 'FIRST' and it contains the two bytes for the first and last sector addresses.

1. Definitions

- OD = Address of track at outer diameter (min. 0)
- ID = Address of track at inner diameter (max. 114)
- FIRST = Address of first sector on a track (min. 1)
- LAST = Address of last sector on a track (max. 32)

2. Locations

Tracks location 1200	bits	14----8	6----0
		ID	OD
Sectors location 1202	bits	12----8	4----0

LAST FIRST

3. Restrictions

The value of 'OD' must be less than or equal to the value of 'ID'. The value of 'FIRST' must be less than or equal to the value of 'LAST'.

If these locations are changed to new limits, then the program will access only those addresses inclusive of and between these limits. The exception to this is test 26 which always uses a special track sequence.

If the 'OD' location is cleared or set to any illegal combination of tracks, the program will clear location 'OD'. The track sequence will then be tracks 0, 52, 53, and 114 (octal) only.

If the 'FIRST' location is cleared or set to any illegal combination of sector address limits then the program will set 'FIRST' to 1 and 'LAST' to 32 (octal).

2.3.3 Software Switch Register (Loc. 176)

For the PDP 11 processors that do not have a hardware switch register or if the operator wishes to select the software switch register, by putting all the switches up to a '1', (this must be done each time the program is started at location 200, otherwise the program will use the hardware SWR.) Location 176 is assigned as the switch register. Bits set to a '1' in this location have the same function as the corresponding switch in the hardware switch register. All references to the SWR are indirect and the program assigns the correct address of the SWR at 'initial start'. See Section 2.4.2 for the selection of operating conditions.

To change the software SWR. while the program is running, type 'control G'. Each time the SWR. is to be tested the program will check to see if the software SWR is selected, and the program is not running in auto mode of RXDP/ACT11. If both conditions exist then the program checks for the CTRL G in the keyboard buffer. If the CTRL G is there the contents of the software SWR. are printed and a 'new =' is asked for. The operator may now type in the new switch register contents, terminated by a carriage return (CR), or if he doesn't want to change the SWR. just terminate with the (CR). Note see the character restrictions below.

When the program detects the (CR) it will replace the contents of the software SWR., if a new one has been typed in, and return to the flow of the program.

NOTE

Character restrictions for changing the software SWR.

1. Only octal numbers 0 - 7 are accepted. Any other character typed will be printed as a ? and the whole SWR must be retyped.
2. To wipe out a "new" contents just typed in, type CTRL U. Now a new contents can be retyped.
3. Only 6 octal characters will be put into the SWR. If more than 6 characters are typed in only the last 6 will be put into the SWR.

2.3.4 Test Parameter Selection ("DTESTP" Loc. 1212)

The drive and test delection must be made before the program starts. Location "DTESTP" (Loc. 1212) is where the bits are set to tell the program what drives are wanted and what tests to run as indicated below. When the program starts it will print out the conditions under which it is running.

Bit 15 (1) Select Drive Unit 1
Bit 14 (1) Select Drive Unit 0

NOTE

If neither of the above bits are set to a 1, then the program expects both drives to be ready for operation (power on, diskettes inserted, and doors closed).

Then set the test selection in bits 4,3,2,1, and 0 as follows:

'DTESTP' BITS	15	14	13-----5	4	3	2	1	0	TESTS
	U1	U0	NOT USED						
BITS 4	3	2	1	0	TESTS				
0	0	0	0	0	(IF NO TEST SELECTED DEFAULTS TO TEST 1)				
0	0	0	0	1	TEST 1				
0	0	0	1	0	TEST 2				
0	0	0	1	1	TEST 3				
0	0	1	0	0	TEST 4				
0	0	1	0	1	TEST 5				
0	0	1	1	0	TEST 6				
0	0	1	1	1	TEST 7				
0	1	0	0	0	TEST 10				
0	1	0	0	1	TEST 11				
0	1	0	1	0	TEST 12				
0	1	0	1	1	TEST 13				
0	1	1	0	0	TEST 14				
0	1	1	0	1	TEST 15				
0	1	1	1	0	TEST 16				
0	1	1	1	1	TEST 17				
1	0	0	0	0	TEST 20				
1	0	0	0	1	TEST 21				
1	0	0	1	0	TEST 22				
1	0	0	1	1	TEST 23				
1	0	1	0	0	TEST 24				
1	0	1	0	1	TEST 25				
1	0	1	1	0	TEST 26				

NOTE

Selection of tests 27 through 37 will cause the message "illegal test" to be printed.

NOTE

When a specified test is selected the program will start at that test and then run through all the following tests until it completes test 26, indicated by the EOP type out. Then it will go back to the test selected and start the next pass. (i.e. If test 24 is selected the program will run test 24, 25, and 26, then go back to test 24.)

An expanded definition of the tests is in section 2.5

2.3.4.1 Prerequisite Of Tests: - The following tests must be run in order, as one test sets up for the next test.

Test 6 before tests 7 and test 10
Test 14 before test 15 and test 16
Test 16 before test 17
Test 21 before test 22 and test 23

See section 2.5 under the above tests for explanation

2.4 Operator Action To Run The Program

2.4.1 Starting the Program

Depending upon the starting address selected the program will do the following:

SA200 (Initial Start)

The selection of hardware or software switch register is made then the program will type its identification number, the test parameters selected in location "DTESTP", and tracks and sectors being tested. The program then proceeds to run under those conditions.

SA202 (Restart)

The program will type out the test parameters selected by the previous initial start, prints the diskette address limits, and starts running the tests. The only operator action required is to set the operating conditions as defined in Section 2.4.2, after depressing the "LOAD ADRS" switch and before depressing the start switch.

2.4.2 Operating Conditions

After the test selection has been made press the "CONT" switch. The program will then ask for operating conditions. Switches 0 and 8 through 15 are used as indicated below. Once they are set up again depress the "CONT" switch. The program is now running under the selected conditions.

SW15-SW0 (1) - Select Software Switch Register

NOTE

If there is a hardware switch register, and the operator wants the software switch register, put all switches up (1) before starting the program at the initial start address.

SW5 (1) - Halt on Error

The program halts on detecting an error, after printing the error message. Pressing "CONT" restores the normal operation of the program.

SW14 (1) Halt at End of Pass

At "end of pass" the program types a bell then an EOP indicator.

"D" means no errors during the pass
"--" means had errors during the pass

If SW14 is set the program will halt. If SW14 is off the program goes back to the test selected and recycles through to the last test, at which time another EOP indicator is printed. If the program halts due to SW14 then press "CONT" will restore the normal flow of the program. If it halts at the end of a pass it will type out the number of passes completed.

SW13 () - Inhibit Error Typeout

At the detection of an error if SW13 is set no error print out will occur. If SW13 is off the error information is printed as described in section 3.0 error detection.

SW12 (1) - Loop on Test

At the completion of a test the program checks SW12. If set the program will go back to the beginning of that test and rerun it. This produces a scope loop on a particular test. The program will stay in this test until:

1. Halt on end of test switch is set
2. Loop on test switch is turned off

At which time the program will go on to the next test.

NOTE

If SW12 is set and no test specified (0) the program will loop on test 1.

3
I
a
(
r
a
t
e
p
3
T
W
O
T

NOTE

To loop on a test that requires a previous test to be run first (Section 2.3.4), select the prerequisite test and set the "HALT AT END OF TEST" switch. Start the program and when it halts, select the desired test and set the "LOOP ON TEST" switch. The program will now stay in that test.

SW11 (1) - Lock on Error

In some tests errors can occur in several places throughout the test. When the error has been reported the program sets a PC flag to indicate where the error occurred. If SW11 is set the program goes back to the beginning of the test running, and goes through the test until:

1. It finds a different error in an earlier part of the test in which case it will lock onto that error.
2. It detects the PC flag indicating this is where the error occurred. It then goes back to the beginning of the test again.

This loop will continue until halt on error switch is set or the lock on error switch is turned off.

SW10 (1) - Halt at End of Test

When set it will halt the program at the end of the test presently running.

SW 9 - Limit Data Error Print Outs

- (0) - When off only the first 10 data byte errors will be printed on a read check test, for each sector. Any more errors will be tabulated but not printed. An error on a different sector will allow 10 more data byte errors to be printed.
- (1) - When set all data byte errors for all sectors will be printed on an error.

SW 8 (1) - Inhibit Recalibration

No recalibration of the drives will occur upon the detection of a seek error if this switch is set.

SW 0 (1) - Inhibit Bell at Error

If SW0 is off the error routine will ring the teleprinter bell at each error detected. With SW0 set no bell will ring.

2.5 Test Definitions

2.5.1 Pretest - Initialize [Key] Part I

Each time the program is started, by either starting address, it runs through a pretest.

Key initialize should set the done flag because any initialization of the RX01 microprocessor is an implied [read sector] of track 1 sector 1. Therefore any error, except parity, that may occur from a normal [read sector] command may occur during an initialize, causing the error flag to set.

Pretest insures that:

1. Done is set
2. Error flag is cleared
3. TR flag is cleared
4. Init done is set

2.5.2 Test 1 - RXCS Test Part I / Interrupt Test Part I

The purpose of this test is to verify that writing all RXCS writable bits to a 0 are not written to a 1.

The program writes the RXCS = 0

No interrupts should occur

The RXCS should remain unchanged = 40 (done)

The RXDB should = 0

2.5.3 Test 2 - Interrupt Test Part II / Vector Address Verification

The purpose of this test is to verify that writing the RXCS interrupt enable bit (bit 6) to a 1, does indeed write it to a 1, therefore because done is set an interrupt should occur (the PDP 11 priority is 0)

2.5.4 Test 3 - Interrupt Test Part III / Priority Level Test Part I

The purpose of this test is to verify the priority of the interrupt request line. The program sets the PDP-11 priority to 4

An RX01 interrupt should occur on priority level 5

If no interrupt occurs then the priority level of the RX11 is not 5, but maybe levels 4,3,2, or 1

2.5.5 Test 4 - Interrupt Test Part IV / Priority Test Part II

The purpose of this test is to verify the priority of the RX11 interrupt request line. The program sets the PDP-11 priority to 5.

No interrupt should occur. If an interrupt does occur then the priority level of the RX11 is not level 5, but maybe level 6, or 7.

2.5.6 Test 5 - Init [Programmed] B / Read Status

The purpose of this test is to verify that setting the RX11 bit 14 causes a RX01 programmed subsystem initialize.

The RXCS should = 40 (done)

The RXDB should = 4, or 104, or 204, or 304

Test 5 cont'd - RXCS test part II / RST

The purpose of this test is to verify the read status command (Function #12).. and that done bit is cleared by the function.

2.5.7 Test 6 - Fill Buffer Transfer Length Test

The purpose of this test is to verify the transfer length of the function 'fill buffer' of the RX01 microcontroller

NOTE

This test loads the sector buffer for test 7 and 10, and must be run previous to them.

2.5.8 Test 7 - Empty Buffer Transfer Length and Content Verification Part I

The purpose of this test is to verify the transfer length of the function "empty buffer" and to verify the contents of the sector buffer.

2.5.9 Test 10 - Empty Buffer Transfer Length and Content Verification Part II

The purpose of this test is to verify the previous empty buffer test did not empty and destroy the contents of the sector buffer.

2.5.10 Test 11 - Fill / Empty Buffer With All 0's

During the empty buffer function this test verifies that all 0's are in fact in the sector buffer.

2.5.11 Test 12 - Fill / Empty Buffer With All 1's

During the empty buffer function this test verifies that all 1's are in fact in the sector buffer.

2.5.12 Test 13 - Drive Ready Verification

Tests that the drive ready (RDY) bit will set for all selected drives. The RDY bit will be set after a read status function directed to the the selected drive.

2.5.13 Test 14 - Error Flag and B-Code Verification Part I

The purpose of this test is to verify that trying to read a non-existent sector will cause an error and the correct error code will be put into the RXDB when the status B is read.

NOTE

This test checks for parity error on the read status B function, the next two tests (115 & 116) do not. This test must be run before tests 15 & 16.

2.5.14 Test 15 - Error Flag and B-Code Verification Part II

This test verifies that trying to write deleted data on an illegal sector will produce an error and the correct B-code is produced. The deleted data bit should be set after this test.

2.5.15 Test 16 - Error Flag and B-Code Verification Part III

Verifies that a write function to a nonexistant sector will produce an error and the correct B-code is produced. The deleted data bit will also be cleared.

NOTE

Test 16 must be run before test 17 as test 16 clears the deleted data bit and test 17 tests that it is cleared.

2.5.16 Test 17 - Illegal Track Error Verification

This test verifies that if a track address larger than 114(octal) is accessed, an error condition will occur, and the B-code will = 40. It also expects the deleted data bit to be cleared.

2.5.17 Test 20 - Seek Verification Via Read Function

This test does a read function on the selected tracks testing for seek errors on various sections of the diskette.

2.5.18 Test 21 - Write Test

The purpose of this test is to write all ones on sector 1, track 1, and to verify that the data in the sector buffer is not changed.

NOTE

This test must be run before tests 22 & 23 as they check for data written on track 1 sector 1.

2.5.19 Test 22 - Initialize Implied Read

After previously writing data on track 1 sector 1, this test changes the contents of the sector buffer and does a programmed initialize. At the end of an init. (recal.) the sector buffer must contain the data from track 1 sector 1.

NOTE

Unit 0 must be on-line for this test to work.

2.5.20 Test 23 - Read Test

This test verifies that a read function does in fact load the sector buffer with data read from the selected address.

2.5.21 Test 24 - Data Transfer and Verification

The purpose of this test is to write then read and check data on all sectors of the selected tracks. The test alternates between drives, if both drives are selected, before changing tracks. The data pattern used is a floating 0 pattern.

2.5.22 Test 25 - Data Verification Via Deleted Data Mode.

This test is the same as Test 24 except it checks for deleted data indicators and uses a data pattern of floating 1.

2.5.23 Test 26 - Head "Home" Test

This test checks for the "home found before the desired track was reached" error code. The head is moved out 10 tracks then decremented back to track 0. It tests all selected drives, and uses a data pattern of random data.

3.0 ERRORS

Pretest and tests 1 - 17, and tests 21 - 23 handle errors as indicated in Section 3.1. For the most part these tests do not rely on an interrupt to indicate the function is completed. Whereas the other tests (tests 20, and 24 - 26) do read, write and read check functions over the selected track, sectors, and drives. These require the interrupt service and error detection that was used in the data reliability test. This is described in Section 3.3.

NOTE

If loop on error switch is up then the program will loop on the shortest set of instructions that will keep it in the failing loop. Otherwise after reporting the error the program will continue running through the remaining addresses and tests.

3.1 Error Heading For Tests 1 - 17, And 21 - 23 Plus Pretest.

The error heading is as follows:

ERADR FAST FAPT [BLANK] GOOD BAD

Under each column the error routine prints pertinent information.

ERADR = Error address
Address of the error trap instruction where the error was detected.

FAST = First address of selected test
Address of the test selected and running

FAPT = First address of present test
Address of the test or subtest presently running, or address of the scope loop.

[BLANK] Additional general information supplied by some tests on an error.

GOOD = Expected results of the test
Test results of what should have happened if there was no error.

BAD = Actual test results
The data that was received from the RX01, that caused the error.

PASS = Number of passes made up to this error

3.2 Error Output Per Test

The following are the types of print outs under the columns [blank], good, and bad for the various tests, using this error format.

TEST (SECTION)	[BLANK] (R2)	GOOD (R0)	BAD (R1)
----	-----	----	----
PRETEST (1)	N/A	40	(RXCS)
PRETEST (2)	(RXCS) INCL.DD BIT	4 OR 204	(RXCS) NO DD BIT
TEST 1 (1)	N/A	40	(RXCS)
TEST 1 (2)	N/A	0	(RXCS)
TEST 1 (3)	(KRXVEC)	N/A	N/A
TEST 2 (1)	(KRXVEC)	N/A	N/A
TEST 2 (2)	(KRXVEC)	140	(RXCS)
TEST 2 (3)	(KRXVEC)	40	(RXCS)
TEST 2 (4)	(KRXVEC)	40	(RXCS)
TEST 2 (5)	(KRXVEC)	40	(RXCS)
TEST 3 (1)	(KRXVEC)	N/A	N/A
TEST 4 (1)	(KRXVEC)	N/A	N/A
TEST 5 (1)	N/A	40	(RXCS)
TEST 5 (2)	(RXDB) INCL.DD BIT	4 OR 204	(RXDB) NO DD BIT
TEST 5 (3)	N/A	0	(RXCS)
TEST 5 (4)	N/A	40	(RXCS)
TEST 5 (5)	(RXCS) INCL.DD BIT	200	(RXCS) NO DD BIT
TEST 6 (1)	NO. OF XFERS	N/A	N/A
TEST 7 (1)	NO. OF XFERS	EXPEC. DATA	ACTUAL DATA

TEST	NO. OF XFERS	EXPEC. DATA	ACTUAL DATA
TEST 10 (1)			
TEST 11&12 (1)	[USES TEST 6 & 7 TO FILL / EMPTY BUFFER]		
TEST 13 (1)	(RXDB)	200	(RXDB) NO DD BIT
TEST 13 (2)	(RXDB)	200	(RXDB) NO DD BIT
TEST 14 (1)	NO. OF TR'S	100040	(RXCS)
TEST 14 (2)	(RXDB)	0	(RXDB) NO DD BIT
TEST 14 (3)	(RXDB)	40	(RXCS)
TEST 14 (4)	N/A	70	(RXDB) ERROR CODE
TEST 15 (1)	NO. OF TR'S	100040	(RXCS)
TEST 15 (2)	N/A	100	(RXDB)
TEST 15 (3)	N/A	70	(RXDB) ERROR CODE
TEST 16 (1)	NO. OF TR'S	100040	(RXCS)
TEST 16 (2)	N/A	0	(RXDB)
TEST 16 (3)	N/A	70	(RXDB) ERROR CODE
TEST 17 (1A)	(RXDB)	0	(RXCS)
TEST 17 (1B)	N/A	100040	(RXCS)
TEST 17 (2)	N/A	0	(RXDB)
TEST 17 (3)	(RXDB)	40	(RXCS)
TEST 17 (4)	N/A	40	(RXDB) ERROR CODE
TEST 21 (1)	(RXES) STATUS A	NO. OF BYTE	(RXDB) STATUS B
TEST 21 (2)	[USES TEST 7 TO EMPTY BUFFER]		
TEST 22	[USES TEST 6 & 7 TO FILL AND EMPTY BUFFER]		
TEST 23	[USES TEST 6 & 21 TO FILL AND CHECK BUFFER]		

3.3 Error Heading for Tests 20, 24 - 26

As previously stated these tests access all the selected sectors, tracks, and drives, and rely on the interrupt service routine to indicate that a function is completed or an error occurred. All errors, with the exceptions where noted, will type as its first or second line of the message "error conditions test PC = XXXX PASS = x". The test PC number is the starting address of the test running, and the pass number is the number of passes made up to the error.

On most errors the program will type out the contents of "Status A" and "Status B".

Status A is the contents of the RXES (error and status register) at the time the error was detected. It shows the CRC, PAR, etc. errors.

Status B is the "definitive error codes" that the RX01 detected, that may have caused the error condition. These error codes are defined in Section 3.3.4

There are three categories of errors as listed and described below.

3.3.1 No Error Flag Errors

These are errors that can occur but the error flag in the RXCS will not be set.

1. Unexpected or missing deleted data bit

This error results when the program expects and doesn't see the DD bit ('D D mark missing'), or doesn't expect and finds the deleted data bit set ('unexpected D D mark'). The program will type out at what diskette address this occurred then continue testing.

NOTE

See Section 3.3.3 for other causes of this error.

2. Data no status error

This error occurs during a read check when the data read does not match the data in the memory data buffer, and there was no CRC error indicated. This means that the data was probably read off the diskette correctly but the transfer between the sector buffer and the RXDB in the RX11 produced bad data.

The error message will include the diskette address, "byte" number in the sector, the data read from the sector buffer "bad", and the expected data from the memory buffer "good".

Byte # Bad Good
(The data patterns are formatted as shown)

0 (Track address; bits 6 - 0)
1 (Unit number bit 7)
 (Sector address bits 4 - 0)

Bytes 2 - 125 contain the selected data pattern.

126 (The sum of all bytes 0 - 125)
127 (The negative of 2 times byte 125)

The program detects a checksum error by summing all the data read from the sector buffer and comparing that sum to 0.

At the end of the data error typeout the program prints if the checksum accumulated was "good" or "bad". If bytes 0 or 1 have data errors the operator must check the results of the checksum. If it is also bad, then there was a true data error. If the checksum was good, then it might be that the head is not over the track expected, and there is a positioning error.

If switch 9 is down then only 10 data errors will be printed, and at the end of the sector the "total read check errors =" will be typed. If switch 9 is up then all the data errors for that sector will be typed out.

3. Power failure

The program tests for two types of power failure, total system power loss, and RX11 power loss resulting in a recalibration of the drives.

The total system power failure is detected by "SYSMAC" subroutine ".SPOWER". When the power is detected to be going down, the registers are saved. When the power comes back up the registers are restored and the message "power" is printed. The program then restarts.

Loss of power in the RX11 causes a recalibration of all drives. When this happens the "init done" bit is set in the RXES register along with the normal done flag. At each interrupt the program tests for the init done bit. If it is found set, the function was not completed and a power loss must have been detected. When this happens the program types out "RX11 Power" and restarts. The error heading is not typed on this error.

4. Unknown interrupt

If an interrupt occurs through the RX11 interrupt vector address and none of the status bits are set (done, error, etc.) the program will type "unknown interrupt" and return back to the program to continue the function. The error heading is not printed.

5. No interrupt at done

The program expects an interrupt at done on the functions of these tests. If an interrupt does not occur at done time then the program will type out "no interrupt at done error" then go into the interrupt service routine as if an interrupt did occur. At this point other errors may be printed if any are detected.

3.3.2 Error Flag Errors

These errors are detected as the results of the error bit being set in the RXCS at an interrupt.

1. Parity error

A parity error results from an incorrect transfer of a command word from the RX11 interface to the RX01 micro-processor controller. The program will type out the contents of the command status register (RXCS) showing the function that failed, the address of the error, contents of status A (RXES) with the parity bit set, contents of status B (RXDB) with the definitive error code of 210 set. Then a "read, write, fill buffer or empty buffer parity error" will be printed. If a parity error occurs on a "read definitive error code" function, then the contents of the RXCS and "parity error" will be typed out.

2. CRC errors

On all data transfers between the sector buffer and the diskette, a CRC word is generated and checked. If an error is detected by the micro-processor in this CRC word then a CRC error is generated. The program again types out the contents of the registers (RXCS contains function, status A with "CRC ERR" bit set, status B with an error code of 200). Then if it is a read only function, or a read check function and there were data errors it will type out "data CRC errors" then print the bad bytes if any. If it was a read check function and there were no data errors it will print "CRC error no data error".

3. Seek errors

Any error that produces a definitive error code but does not set an error bit in status A (RXDB at end of function) is labeled a seek error. See Section 3.3.4 for error codes and meanings. The same information is printed for these errors as in parity, or CRC errors, except it states that it is a "write or read seek error". If switch 8 is down then at each seek error found the program does an initialize of the RX01 so it will recalibrate to a known (home) position. The program then goes on to the next sector or track and continues testing, if the loop on error switch is off. (See Section 3.3.3. for errors caused by previous errors.) If the loop on error switch is up it will retry the function at the same address. If switch 8 is up then no "initialize" is done and the program looks at the other switches for operating conditions. Seek errors also print the track address that the head moved from at the time of the error.

4. Error flag error

If the error flag is not set in the RXCS and an error bit is set in status A or an error code is set in status B then there was an error but the error flag was not set. The message "error flag error" is printed then the program continues to type out the type of error.

3.3.3 Errors Resulting From Previous Errors

If there is a "write seek error" the program will go on to the next address without writing on the address where the error occurred. (Unless the loop on error switch 11 is up and the seek error is recovered.) If the write function is followed by a read check function and the read does not have a seek error at the same address, then there may be data errors, or unexpected or missing deleted data bit errors resulting from no data being written on that address by the previous write function.

3.3.4 Definitive Error Codes

The RX01 micro-processor has defined the error codes and meanings which are available to the program by issuing command #7 "read definitive error code"

The following are the codes and their meanings:

- 10 - Drive 0 failed to see home from initialize
- 20 - Drive 1 failed to see home from initialize
- 30 - Home found when stepping out 10 tracks for init.
- 40 - Tried to access a track greater than 76
- 50 - Home found before desired track was reached
- 60 - Self diagnostic error
- 70 - Desired sector not found after sampling 52 headers
- 100 - Write protect error
- 110 - More than 40 us and no sep clock detected
- 120 - A preamble could not be found
- 130 - Preamble found but no ID mark found in time
- 140 - CRC error on a header, no error flag
- 150 - Good header (no CRC error) but track compare error
- 160 - ID address mark not found in time
- 170 - Data mark not found in time
- 200 - Data CRC error
- 210 - Parity errors

3.4 Program Hung

If there is no response from the RX11 while waiting for the transfer request (TR) flag or the done flag, the program will type "device test hung @ PC" (only if SW13 is off) and then go on to the next test, or the beginning of the present test.

4.0 HALTS

The only halts in the program are the selectable halts (EOP, EOT, at error), the illegal vector halts, and the illegal test selection halt.

NOTE

One additional 'halt' exists in the program. It occurs when the user has loaded his program via the 'RXDP' monitor (on unit 0) and also requires testing of unit 0. A prompt message is typed reminding the user to replace his load medium with a scratch diskette before going on. The program will wait for the 'continue' switch to be depressed.

5.0 FLOW CHARTS

84	BASIC DEFINITIONS
253	TEST SELECTION VIA SWITCH REGISTER
273	OPERATIONAL SWITCH REGISTER POSITIONS
299	RXCS (RX COMMAND STATUS REGISTER)
350	RXDB (RX DATA BUFFER REGISTER)
402	START AND RESTART ADDRESSES
424	GET VALUE FOR SOFTWARE SWITCH REGISTER
535	PRETES: - INITIALIZE [KEY] PART I
908	TEST 1 - RXCS TEST PART I / INTERRUPT TEST PART I
1070	TEST 2 - INTERRUPT TEST PART II / VECTOR ADDRESS VERIFICATION
1308	TEST 3 - INTERRUPT TEST PART III / PRIORITY LEVEL VERIFICATION PART I
1366	TEST 4 - INTERRUPT TEST PART IV / PRIORITY VERIFICATION PART II
1426	TEST 5 - INIT [PROGRAMMED] / RST
1612	TEST 6 - FILL BUFFER TRANSFER LENGTH VERIFICATION
1713	TEST 10 - EMPTY BUFFER XFER LENGTH AND CONTENT VERIFICATION PART II
1721	TEST 7 - EMPTY BUFFER XFER LENGTH AND CONTENT VERIFICATION PART I
1802	TEST 12 - FILL/EMPTY BUFFER ALL 1'S
1809	TEST 11 - FILL/EMPTY BUFFER ALL 0'S
1821	TEST 13 DRIVE READY VERIFICATION
1901	TEST 14 - ERROR FLAG AND B-CODE VERIFICATION PART I
2036	TEST 15 - ERROR FLAG AND B-CODE VERIFICATION PART II
2089	TEST 16 - ERROR FLAG AND B-CODE VERIFICATION PART III
2172	TEST 17 - ILLEGAL TRACK ERROR VERIFICATION
2277	TEST 20 - SEEK VERIFICATION VIA READ FUNCTION
2315	TEST 21 - WRITE TEST
2384	TEST 22 - INITIALIZE IMPLIED READ
2406	TEST 23 - READ TEST
2421	TEST 24 - DATA TRANSFER AND VERIFICATION
2435	TEST 25 - DATA TRANSFER AND VERIFICATION VIA DELETED DATA MODE
2443	TEST 26 - HEAD 'HOME' TEST
2511	" ERROR " TRAP SERVICE ROUTINE
2586	" SCOPE " TRAP SERVICE ROUTINE
2703	DRIVE TEST SELECTION
2750	WRITE FUNCTION
2889	READ DATA FROM THE DISKETTE
3020	READ AND VERIFY DATA
3160	INTERRUPT SERVICE
3269	PATTERN GENERATOR
3412	UNIT SELECTION
3455	TRACK SEQUENCE SELECTION
3547	SECTOR SELECTION
3580	TYPE ROUTINE
3668	BINARY TO OCTAL (ASCII) AND TYPE
3745	SAVE AND RESTORE R0-R5 ROUTINES
3790	TTY INPUT ROUTINE
3937	TRAP DECODER
3960	TRAP TABLE
3981	POWER DOWN AND UP ROUTINES
4026	SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
4044	DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
4136	MESSAGES

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

.NLIST CND,MD,MC
.LIST ME
.ENABL ABS,AMA

.TITLE CZRXBFO RX11 INTERFACE TEST
.*COPYRIGHT (C) MAY 8,1979
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754

.*PROGRAM BY D. ADAMS/B. BURGESS/MIKE PAGE

.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.

0000C1
160000

.\$TN=1
.\$SWR=160000 ;:HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT

:COPYRIGHT (C) 1975,1976
:THIS SOFTWARE IS FURNISHED UNDER LICENCE FOR USE ONLY
:ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH
:THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS
:SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED
:OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
:EXCEPT FOR USE ON SUCH SYSTEM, AND TO ONE WHO AGREES TO
:THESE LICENCE TERMS. TITLE TO OWNERSHIP OF THE
:SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

:THE INFOMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
:WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT
:BY DIGITAL EQUIPMENT CORPORATION.

:DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY
:OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

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

:MODIFIED TO REV. D BY B. BURGESS NOV. 10, 1975 AS FOLLOWS:

- :A) ADDED CAPABILITY OF VARIABLE DEVICE 'BR' LEVEL. ALL RELEVANT TESTS CALCULATE 'CPU' LEVEL BASED ON CURRENT CONTENTS OF LOCATION 'BRLEV:'. DEFAULT 'BR' LEVEL, FOR THE DEVICE, SET BY THE PROGRAM IS 5. ANY OTHER 'BR' LEVEL (E.G. 6) WOULD HAVE TO BE PATCHED INTO LOCATION 'BRLEV:' BEFORE RUNNING THE PROGRAM.
- :B) ADDED TWO (2) ROUTINES TO HANDLE 'UNEXPECTED' BUS TIMEOUT AND RESERVED INSTRUCTION TRAPS (TRAPS TO VECTORS 4 & 10, RESPECTIVELY). BOTH ROUTINES WILL INDICATE WHICH TRAP OCCURRED, THE 'PC' LOCATION OF WHERE THE TRAP OCCURRED, AND ATTEMPT TO RESTART THE PROGRAM.
- :C) ADDED CODE TO FAILSAFE UNIT 0 UNDERGOING TESTING IF PROGRAM WAS LOADED VIA UNIT 0 USING 'RXDP' MONITOR AND USER STARTED RUNNING THE PROGRAM WITHOUT HAVING REPLACED HIS LOAD MEDIUM WITH A 'SCRATCH' DISKETTE.
- :D) ADDED MESSAGES TO INDICATE TO USER WHEN HE HAS SELECTED TRACK AND/OR SECTOR LIMITS 'OUT OF RANGE' AND CORRESPONDING DEFAULT LIMITS WHEN THIS CONDITION ARISES
- :E) MODIFIED TESTS 1 THRU 4 TO CORRECTLY PRINT OUT THE CONTENTS OF 'KRXVEC' (LOCATION HOLDING THE DEVICE VECTOR) AS 264 INSTEAD OF 270.
- :F) MODIFIED TEST 2 TO HANDLE A 'LOCKED IN INTERRUPT STATE' CONDITION ARISING WHEN 'INTERRUPT ENABLE' AND 'DONE' ARE BOTH QUALIFIED AND THE 'REQUEST INTERRUPT' FLOP NEVER GETS CLEARED.
- :G) ADDED EXTENSIVE MAINTENANCE INFORMATION BASED ON FAULT INSERTION RESULTS. INFORMATION IS KEYED TO THE 'ERROR' REPORT WITHIN A TEST. INFORMATION PROVIDED SHOULD BE SELF-EXPLANATORY BUT SHOULD NOT BE MISCONSTRUED AS BEING ALL ENCOMPASSING DUE TO HUMAN ERRORS IN STATISTICS GATHERING, INABILITY TO FAULT INSERT SOME CHIPS, AS WELL AS ONLY TWO (2) MODULES ABLE TO BE FAULT INSERTED I.E. - M7846 (UNIBUS INTERFACE) AND M7727 (READ/WRITE CONTROL).
- :H) ADDED FLOW CHARTS


```
80
81      .SBTTL BASIC DEFINITIONS
82
83      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***
84      001200      STACK= 1200
85      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
86      .EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
87
88      ;*MISCELLANEOUS DEFINITIONS
89      000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
90      000012      LF= 12      ;;CODE FOR LINE FEED
91      000015      CR= 15      ;;CODE FOR CARRIAGE RETURN
92      000200      CRLF= 200    ;;CODE FOR CARRIAGE RETURN-LINE FEED
93      177776      PS= 177776   ;;PROCESSOR STATUS WORD
94      .EQUIV PS,PSW
95      177774      STKLMT= 177774 ;;STACK LIMIT REGISTER
96      177772      PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
97      177570      DSWR= 177570 ;;HARDWARE SWITCH REGISTER
98      177570      DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
99
100     ;*GENERAL PURPOSE REGISTER DEFINITIONS
101     000000      R0= %0      ;;GENERAL REGISTER
102     000001      R1= %1      ;;GENERAL REGISTER
103     000002      R2= %2      ;;GENERAL REGISTER
104     000003      R3= %3      ;;GENERAL REGISTER
105     000004      R4= %4      ;;GENERAL REGISTER
106     000005      R5= %5      ;;GENERAL REGISTER
107     000006      R6= %6      ;;GENERAL REGISTER
108     000007      R7= %7      ;;GENERAL REGISTER
109     000006      SP= %6      ;;STACK POINTER
110     000007      PC= %7      ;;PROGRAM COUNTER
111
112     ;*PRIORITY LEVEL DEFINITIONS
113     000000      PR0= 0      ;;PRIORITY LEVEL 0
114     000040      PR1= 40     ;;PRIORITY LEVEL 1
115     000100      PR2= 100    ;;PRIORITY LEVEL 2
116     000140      PR3= 140    ;;PRIORITY LEVEL 3
117     000200      PR4= 200    ;;PRIORITY LEVEL 4
118     000240      PR5= 240    ;;PRIORITY LEVEL 5
119     000300      PR6= 300    ;;PRIORITY LEVEL 6
120     000340      PR7= 340    ;;PRIORITY LEVEL 7
121
122     ;*'SWITCH REGISTER' SWITCH DEFINITIONS
123     100000      SW15= 100000
124     040000      SW14= 40000
125     020000      SW13= 20000
126     010000      SW12= 10000
127     004000      SW11= 4000
128     002000      SW10= 2000
129     001000      SW09= 1000
130     000400      SW08= 400
131     000200      SW07= 200
132     000100      SW06= 100
133     000040      SW05= 40
134     000020      SW04= 20
135     000010      SW03= 10
```

```

136      000004      SW02= 4
137      000002      SW01= 2
138      000001      SW00= 1
139      .EQUIV      SW09,SW9
140      .EQUIV      SW08,SW8
141      .EQUIV      SW07,SW7
142      .EQUIV      SW06,SW6
143      .EQUIV      SW05,SW5
144      .EQUIV      SW04,SW4
145      .EQUIV      SW03,SW3
146      .EQUIV      SW02,SW2
147      .EQUIV      SW01,SW1
148      .EQUIV      SW00,SW0
149
150      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
151      100000      BIT15= 100000
152      040000      BIT14= 40000
153      020000      BIT13= 20000
154      010000      BIT12= 10000
155      004000      BIT11= 4000
156      002000      BIT10= 2000
157      001000      BIT09= 1000
158      000400      BIT08= 400
159      000200      BIT07= 200
160      000100      BIT06= 100
161      000040      BIT05= 40
162      000020      BIT04= 20
163      000010      BIT03= 10
164      000004      BIT02= 4
165      000002      BIT01= 2
166      000001      BIT00= 1
167      .EQUIV      BIT09,BIT9
168      .EQUIV      BIT08,BIT8
169      .EQUIV      BIT07,BIT7
170      .EQUIV      BIT06,BIT6
171      .EQUIV      BIT05,BIT5
172      .EQUIV      BIT04,BIT4
173      .EQUIV      BIT03,BIT3
174      .EQUIV      BIT02,BIT2
175      .EQUIV      BIT01,BIT1
176      .EQUIV      BIT00,BIT0
177
178      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
179      000004      ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
180      000010      RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
181      000014      TBITVEC=14         ;;"T" BIT
182      000014      TRTVEC= 14         ;;TRACE TRAP
183      000014      BPTVEC= 14         ;;BREAKPOINT TRAP (BPT)
184      000020      ICTVEC= 20         ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
185      000024      PWRVEC= 24         ;;POWER FAIL
186      000030      EMTVEC= 30         ;;EMULATOR TRAP (EMT) **ERROR**
187      000034      TRAPVEC=34         ;;"TRAP" TRAP
188      000060      TKVEC= 60          ;;TTY KEYBOARD VECTOR
189      000064      TPVEC= 64          ;;TTY PRINTER VECTOR
190      000240      PIRQVEC=240        ;;PROGRAM INTERRUPT REQUEST VECTOR
191

```

```

192
193
194           ;SPECIAL EQUATES
195
196
197           000017           RDER           =17           ; READ B CODE
198           000040           DONEBIT        =40
199           000101           FBIE           =101           ; IE+FULL BUFFER
200           000103           EBIE           =103           ; IE+EMPTY BUFFER
201           000105           WRTIE         =105           ; IE+WRITE SECTOR
202           000107           RDIE           =107           ; IE+READ SECTOR
203           000115           WTDDIE        =115           ; IE+WRITE DD SECTOR
204           040001           RECAL          =40001
205           000000           OPEN           =0
206
207           000000           .=0
208 000000 000000 000000           .WORD 0,0
209
210           000004           .=4
211 000004 006156           .WORD  BUSERR           ;UNEXPECTED TIMEOUT TRAP PC
212 000006 000340           .WORD  PR7              ;UNEXPECTED TIMEOUT TRAP PS
213 000010 006202           .WORD  RESERR           ;UNEXPECTED RESERVED INSTRUCTION TRAP PC
214 000012 000340           .WORD  PR7              ;UNEXPECTED RESERVED INSTRUCTION TRAP PS
215
216           000020           .=20
217 000020 006524           XSCOPE
218 000022 000340           PR7
219 000024 015160           $PWDRN
220 000026 000340           PR7
221 000030 006260           XERROR
222 000032 000340           PR7
223 000034 015074           $TRAP
224 000036 000340           PR7           ;ADDRESS OF TRAP SERVICE
225
226
227           000042           .=42
228 000042 000000           .WORD  0
229
230           000046           .=46
231 000046 002470           LOGICAL           ;ACT 11 EOP HOOKS
232
233
234           000052           .=52
235 000052 000700           .WORD  0
236
237           000174           .=174
238 000174 000000           DISPREG:          0
239 000176 000000           SWREG:            0
240
241
242           000200           .=200
243 000200 000401           BR 1$
244 000202 000402           BR 2$
245 000204 000137 001232           1$: JMP SA200           ;OPERATOR SELECTED CONDITIONS
246 000210 000137 001222           2$: JMP SA202           ;RESTART PROGRAM WITH PREVIOUS PARAMETERS
247

```

CZRXBFO RY11 INTERFACE TEST
CZRXBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 J 3
PAGE 7
BASIC DEFINITIONS

SEQ 0035

248
249

250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295

.SBTTL TEST SELECTION VIA SWITCH REGISTER

:SET TEST AND DRIVE SELECTION IN " DTESTP " LOCATION 1212

: BIT 15 = 1 - UNIT 1 SELECTED
: BIT 14 = 1 - UNIT 0 SELECTED
: BIT 15 & BIT 14 = 0 - BOTH DRIVES MUST BE READY
:
: BIT 4 - BIT 0 = OCTAL NUMBER OF DESIRED STARTING TEST
: BIT 4 - BIT 0 = 0 -ALL TESTS WILL BE SEQUENCED THROUGH

.SBTTL OPERATIONAL SWITCH REGISTER POSITIONS

: SET OPERATING CONDITIONS IN THE SWITCH REGISTER (HARDWARE)
: OR SOFTWARE SWITCH REGISTER LOCATION 176

: 15 = 1 - HALT ON ERROR
: 14 = 1 - HALT AT END OF PASS
: 13 = 1 - INHIBIT ERROR TYPEOUT
: 12 = 1 - LOOP ON TEST
: 11 = 1 - LOCK ON ERROR
: 10 = 1 - HALT AT END OF TEST
: 9 = 1 - PRINT ALL DATA ERRORS
: 9 = 0 - PRINT ONLY FIRST 10 DATA ERRORS PER SECTOR
: 8 = 1 - INHIBIT RECALIBRATION ON SEEK ERRORS.

: 0 = 1 - INHIBIT <BELL> AT ERROR

: 15-0 = 1 - SELECT SOFTWARE SWITCH REGISTER


```
296 .SBTTL RXCS (RX COMMAND STATUS REGISTER)
297
298 001200 .=STACK
299
300 001200 000000 OD: 0 ;OD/ID = 0 UNLESS SPECIFIC TRACKS SELECTED.
301 001201 001201 ID=OD+1
302 001202 015001 FIRST: 015001 ; FIRST = 1, LAST = 32
303 001203 001203 LAST=FIRST+1
304
305 001204 000264 KRXVEC: 264
306
307 001206 177170 RXCS: 177170
308
309 ; RXCS: STANDARD DEVICE ADDRESS = 177170
310
311 ; TOGGLE INTO PROGRAM LOCATION " RXCS " THE RX11 DEVICE ADDRESS IF NOT = 177170
312
313 ;KEY: R - READ ONLY BIT
314 ; W - WRITE ONLY BIT
315
316 : 15 - R - ERROR
317 : 14 - W - INITIALIZE
318 : 13 -
319 : 12 -
320 : 11 - (BITS 13-8)
321 : 10 - (NOT USED)
322 : 9 -
323 : 8 -
324 : 7 - R - TRANSFER REQUEST
325 : 6 - R/W- INTERRUPT ENABLE
326 : 5 - R - DONE
327 : 4 - W - UNIT SELECT
328 : 3 - W - FUNCTION
329 : 2 - W - FUNCTION
330 : 1 - W - FUNCTION
331 : 0 - W - GO !
332
333 ; FUNCTION
334
335 ; 3 2 1 0
336
337 ; - - - GO
338
339 : 0 + GO - FILL BUFFER
340 : 2 + GO - EMPTY BUFFER
341 : 4 + GO - WRITE SECTOR
342 : 6 + GO - READ SECTOR
343 : -
344 : 12 + GO - READ STATUS " A "
345 : 14 + GO - WRITE DELETED DATA
346 : 16 + GO - READ STATUS " B " (CODES)
```

347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398

001210 177172

001212 000000

001214 000005

001216 177570

001220 177570

```
.SBTTL RXDB (RX DATA BUFFER REGISTER)
RXDB: 177172
; RXDB: STANDARD DEVICE ADRESS = 177172
; THE FOLLOWING BIT IDENTIFICATION REPRESENTS THE STATUS AT THE END OF A FUNCTION
; (BUT NOT FUNCTION # 16 TO READ STATUS " B ") DISPLAYED WITHIN THE RX-DATA BUFFER.
;
; (A) 7 - SELECTED DRIVE READY
;      6 - DELETED DATA
;      5 -
;      4 -
;      3 - WRITE PROTECT ERROR
; (B) 2 - INITIALIZE DONE
;      1 - PARITY ERROR
;      0 - CRC ERROR
;
; (A) - VISIBLE ONLY IF THE FUNCTION WAS # 12 READ STATUS " A "
; (B) - INIT DONE VISIBLE IF AN INITIZLIAE [KEY] OR [PROGRAMMED] WAS ISSUED
DTESTP: 0
BRLEV: 5
;BRLEV: STANDARD PRIORITY INTERRUPT LEVEL = 5
;TOGGLE INTO PROGRAM LOCATION "BRLEV" THE RX11 INTERRUPT PRIORITY
;LEVEL IF NOT = 5
SWR: .WORD DSWR
DISPLAY: .WORD DDISP
; R0 - GOOD /EXPECTED RESULT OF TEST
; R1 - EAC /ACTUAL RESULT OF TEST
; R2 - BLANK /
; R3 - TEST 0
;*****
;WORD "UNITSEL" HAS THE FOLLOWING BIT DEFFINITIONS
;
;BIT15 = 1 - UNIT 1 SELECTED FOR USE
;BIT14 = 1 - UNIT 1 IN USE
;BIT8 = 1 - THIS PASS HAD AN ERROR
;BIT7 = 1 - UNIT 0 SELECTED FOR USE
;BIT6 = 1 - UNIT 0 IN USE
;BIT4 = UNIT SELECTION BIT
;*****
```

```
399                                     .SBTTL START AND RESTART ADDRESSES
400
401                                     ; THE STARTING ADDRESS WAS 202
402
403 001222 005200                      SA202: INC R0
404 001224 012706 001200                MOV #STACK,SP
405 001230 000447                      BR RESTART
406
407                                     ; THE STARTING ADDRESS WAS 200
408
409 001232 005000                      SA200: CLR R0
410 001234 012737 177570 001216        MOV #177570,SWR ;RESET TO HARDWARE SWR.
411 001242 012706 001200                MOV #STACK,SP
412 001246 104401 016746                TYPE ,MREV ;PRINT THE NAME AND REVISION
413 001252 013746 000004                MOV 4,-(SP) ;SAVE 'BUSERR' TIMEOUT 'PC'
414 001256 012737 001276 000004        MOV #1$,4 ;SET UP TIMEOUT VECTOR
415 001264 022777 177777 177724        CMP #177777,@SWR ;IS SOFTWARE SWR SELECTED
416 001272 001402                      BEQ 2$ ;YES, INSERT IT'S ADDRESS
417 001274 000423                      BR 3$ ;BR IF NO TIMEOUT TRAP OCCURS
418 001276 022626                      1$: CMP (SP)+,(SP)+ ;RESTORE THE STACK
419 001300 012737 000176 001216        2$: MOV #SWREG,SWR ;POINT TO SOFTWARE SWITCH REGISTER
420 001306 012737 000174 001220        MOV #DISPREG,DISPLAY ;POINT TO SOFTWARE DISPLAY REG.
421                                     .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
422 001314 005737 000042                TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
423 001320 001006                      BNE 64$ ;;BRANCH IF YES
424 001322 023727 001216 000176        CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
425 001330 001005                      BNE 65$ ;;BRANCH IF NO
426 001332 104405                      GTSWR ;;GET SOFT-SWR SETTINGS
427 001334 000403                      BR 65$
428 001336 112737 000001 015072        64$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
429 001344
430 001344 012637 000004                3$: MOV (SP)+,4 ;RESET TIMEOUT VECTOR TO 'BUSERR'
431 001350 000005                      RESTART:RESET ;INITIALIZE THE RX11 SYSTEM
432 001352 012746 000340                MOV #PR7,-(SP)
433 001356 012746 001364                MOV #4$,-(SP)
434 001362 000002                      RTI ;LOAD THE PSW
435 001364 013737 001206 001210        4$: MOV RXCS, RXDB ;GET ADDRESS OF RXCS
436 001372 062737 000002 001210        ADD #2, RXDB ;SET UP ADDRESS OF RXDB
437 001400 012737 001234 012654        MOV #001234,RAN1 ; INITIALIZE CONSTANTS OF
438 001406 012737 000765 012656        MOV #000765,RAN2 ; RANDOM NUMBER GENERATOR
439 001414 005037 002554                CLR CCOUNT
440 001420 005037 002556                CLR PASS
441 001424 005037 007004                CLR HANGER
442 001430 012737 177740 007006        MOV #177740,HANGPL
443 001436 005700                      TST R0
444 001440 001064                      BNE XSA202 ; STARTING ADDRESS WAS 202
445 001442 005037 012752                CLR UNITSEL
446 001446 032737 140000 001212        BIT #140000,DTESTP ;WERE ANY DRIVES SELECTED
447 001454 001004                      BNE 1$ ;YES GO SET THEIR BITS
448 001456 052737 100200 012752        BIS #100200,UNITSEL ;NO, BOTH UNITS MUST BE READY
449 001464 000415                      BR 2$
450 001466 032737 040000 001212        1$: BIT #BIT14,DTESTP ;WAS UNIT 0 SELECTED
451 001474 001443                      BEQ 3$ ;NO, MUST BE UNIT 1
452 001476 052737 000200 012752        BIS #200,UNITSEL ;YES,SET SELECTED BIT
453 001504 005737 001212                TST DTESTP ;WAS UNIT 1 SELECTED
454 001510 100003                      BPL 2$ ;NO
```



```
455 001512 052737 100000 012752      BIS #BIT15,UNITSEL      ;YES,SET THE SELECTED BIT
456 001520 005737 000042      TST @#42                ;AUTO MODE?
457 001524 001432              BEQ XSA202               ;BRANCH IF NOT
458 001526 023737 000042 000046  CMP @#42,@#46           ;ACT MODE ?
459 001534 001007              BNE 6$                  ;BRANCH IF NOT
460 001536 012777 000176 177452  MOV #SWREG,@SWR         ;GET SOFT SWITCH REG FOR AUTO MODE
461 001544 052777 100000 177444  BIS #BIT15,@SWR         ;SET FOR HALT ON ERROR
462 001552 000417              BR XSA202
463 001554 132737 000010 000041 6$: BITB #10,@#41          ;WAS PROG. LOADED FROM RX01 ?
464 001562 001413              BEQ XSA202               ;BRANCH IF NOT
465 001564 042737 000200 012752  BIC #200,UNITSEL
466 001572 104401 016235      TYPE ,MUNIT1
467 001576 104401 016245      TYPE ,MONLY
468 001602 000403              BR XSA202
469 001604 052737 100000 012752 3$: BIS #BIT15,UNITSEL
470 001612 042737 100200 001200 XSA202: BIC #100200,OD    ;CLEAR 1ST TIME BITS FOR BOTH DRIVES
471 001620 104401 015656      TYPE, MDTESTP
472 001624 013746 001212      MOV DTESTP,-(SP)        ;;SAVE DTESTP FOR TYPEOUT
473 001630 104403              TYPOS                    ;;GO TYPE--OCTAL ASCII
474 001632 006                .BYTE 6                  ;;TYPE 6 DIGIT(S)
475 001633 000                .BYTE 0                  ;;SUPPRESS LEADING ZEROS
476 001634 104401 016162      TYPE ,MCRLF
477 001640 005737 001200      LIMITS: TST OD
478 001644 001005              BNE TRKLMT
479 001646 005037 013164      CLR SEQUEN
480 001652 104401 016461      TYPE ,MLIMTRK
481 001656 000432              BR SECLMT
482
483
484
485 001660 123727 001201 000114 TRKLMT: CMPB ID,#114
486 001666 101021              BHI 1$
487 001670 123737 001200 001201  CMPB OD,ID
488 001676 101015              BHI 1$
489 001700 104401 016507      TYPE ,MOD
490 001704 113746 001200      MOVB OD,-(SP)
491 001710 104403              TYPOS
492 001712 003                .BYTE 3
493 001713 000                .BYTE 0
494 001714 104401 016513      TYPE ,MID
495 001720 113746 001201      MOVB ID,-(SP)
496 001724 104403              TYPOS
497 001726 003                .BYTE 3
498 001727 000                .BYTE 0
499 001730 000405              BR SECLMT
500 001732 104401 017126 1$: TYPE, OD2BIG          ;TYPE MSG. INDICATING ID OR OD
501
502
503 001736 005037 001200      CLR OD                  ;TOO BIG & DEFAULTING TO TRACKS
504 001742 000736              BR LIMITS                ;0, 52, 53, 114
505
506
507
508 001744 105737 001202      ; 1 <= FIRST <= LAST <= 32
509 001750 001003              SECLMT: TSTB FIRST
510 001752 112737 000001 001202  BNE 1$
                    MOVB #1,FIRST
```

```

511 001760 123727 001203 000032 1$:  CMPB LAST,#32
512 001766 101023          BHI 2$
513 001770 123737 001202 001203  CMPB FIRST, LAST
514 001776 101017          BHI 2$
515 002000 104401 016521 3$:  TYPE ,MFIRST
516 002004 113746 001202  MOVB FIRST,-(SP)
517 002010 104403          TYPOS
518 002012 003           .BYTE 3
519 002013 000           .BYTE 0
520 002014 104401 016532  TYPE ,MLAST
521 002020 113746 001203  MOVB LAST,-(SP)
522 002024 104403          TYPOS
523 002026 003           .BYTE 3
524 002027 000           .BYTE 0
525 002030 104401 016162  TYPE ,MCRLF
526 002034 000406          BR PRETEST
527 002036 104401 017213 2$:  TYPE, S2BIG
528
529
530 002042 012737 015001 001202  MOV #15001,FIRST
531 002050 000753          BR 3$
  
```

```

;TYPE MSG. INDICATING THAT
;SECTOR RANGE SELECTED WAS
;INVALID AND DEFAULTING TO A
; 1ST SECTOR VALUE OF 1 AND
;A LAST SECTOR VALUE OF 32
  
```

```

532 .SBTTL PRETEST - INITIALIZE [KEY] PART 1
533
534 "KEY" INITIALIZE SHOULD HAVE [SET] THE DONE FLAG BECAUSE
535 ANY [INIT] OF THE RX01 MICROCONTROLLER IS AN IMPLIED [READ SECTOR]
536 OF TRACK 1 SECTOR 1 (FOR SYSTEMS PROGRAMMING BOOTSTRAP APPLICATIONS).
537
538 THEREFORE, ANY ERROR (EXCEPT PARITY) THAT MAY OCCUR FROM A NORMAL
539 "READ SECTOR" COMMAND MAY OCCUR HERE CAUSING THE ERROR FLAG TO SET, AND
540 DISPLAYING THE ERROR STATUS WITHIN THE TRANSFER REGISTER AT "DONE".
541
542 THE TRANSFER REQUEST FLAG SHOULD BE CLEARED.
543
544 ;NOTE: SCOPE LOOPING IS NOT OFFERED BECAUSE THE "INIT" FUNCTION
545 WHICH PRODUCED THE ERROR HAS NOT YET BEEN VERIFIED.
546
547 002052 042737 000400 012752 PRETEST: BIC #BIT8,UNITSEL
548 002060 005337 006520 CLR ERRORS
549 002064 000004 SCOPE ; REFRESHES FAST FOR ERROR TYPEOUT
550 002066 013737 006604 002560 MOV PCSCOPE,FAST ; FOR ERROR TYPEOUT INFORMATION
551 002074 012700 000040 MOV #40, R0 ; PROGRAM EXPECTS DONE FLAG
552 002100 017701 177102 3$: MOV @RXCS, R1 ; ACTUAL CONTENTS OF RXCS
553 002104 020100 CMP R1, R0
554 002106 001407 BEQ 1$ ; OK
555 002110 062737 000001 007004 ADD #1,HANGER ;WAIT FOR THE INIT. FUNCTION TO
556 002116 005537 007006 ADC HANGPL ;FINISH BEFORE CALLING IT AN ERROR
557 002122 001366 BNE 3$
558
559 ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A
560
561 002124 104000 ERROR ; RXCS NOT = 40 (DONE)
562

```

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
;THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
;BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
;AREAS TO CHECK FOR THE RELEVANT FAULT/S.

```

```

;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
;ANALYZE THE FOLLOWING AREA/S:

```

```

;M7846 (UNIBUS INTERFACE)

```

```

; SIGNAL NAME REASON POSSIBLE CHIPS

```

```

; NOTE: MAKE SURE THE DRIVES ARE CONNECTED CORECTLY,THE
; DISKETTES INSERTED,AND THE DOORS OF THE SELECTED DRIVES
; ARE CLOSED. IF THE THESE CONDITIONS ARE NOT SET THERE
; WILL BE AN ERROR AT THIS POINT.

```

```

;B TRANSFER REQUEST STUCK HIGH E22

```

```

: DONE                STUCK LOW                E22,E36,E1
: RUN(0) H            STUCK LOW                E22
: SELECT 00          STUCK LOW                E18,E34,E17,E40,E11
: B DONE             STUCK LOW                E18,E15,E19,E41
: RUN(1) H           STUCK HIGH               E18
: RX INIT            STUCK HIGH               E32
: OUT                STUCK HIGH               E23,E4
: B INIT             STUCK HIGH               E36,E8
: BUS -> RXCS        STUCK HIGH/LOW           E36,E40
: BUS INTR           STUCK HIGH               E38
: BUS D02/D04        STUCK LOW                E38
: RUN(1) H           STUCK LOW                E37
: INT ENB(1) H       STUCK LOW                E37
: TRANSFER REQUEST   STUCK HIGH               E1
: INT ENB(1) H       STUCK HIGH               E1
: -----
: -----
: STROBE DISABLED    E1
: 'A' INPUTS NOT SELECTED E1
: CMD                STUCK LOW                E17,E21
: B DONE             STUCK HIGH               E15
: RUN(0) H           STUCK HIGH               E15
: OUT                STUCK LOW                E24
: PX INIT            STUCK LOW                E21,E11
: IN                 STUCK HIGH               E21
: RX RUN             STUCK LOW                E14
: -----
: -----
: LOCKED IN 'PRESET' STATE E2
: NO STROBE SIGNAL   E3
: BUS D15            STUCK LOW                E9
: DATA              STUCK HIGH               E11

```

: IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTER-ACE MODULE
 : AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
 : M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
 : MODULE M7727.

NOTE: ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCIVE TO
 THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
 ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
 HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
 REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
 LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.

: M7727 (READ/WRITE CONTROL)

SIGNAL NAME	REASON	POSSIBLE CHIPS
-----	PIN 15 NOT AT GROUND	E15
: SEL TRK 0	STUCK LOW	E15
: DK0 TRK 0	STUCK LOW	E15
: SEL DK0	STUCK LOW	E13
: DC LO	STUCK LOW	E13
: OUT	STUCK HIGH	E13
: -----	E18 CLOCK LOCKED HIGH/LOW	E14,E18
: -----	REPLACE E18	-----
: INIT	STUCK LOW	E18,E16
: -----	PIN 14 NOT +5V THRU 1k	E16

NOTE: MAKE SURE THE DRIVES ARE CONNECTED CORECTLY, THE
 DISKETTES INSERTED, AND THE DOORS OF THE SELECTED DRIVES
 ARE CLOSED. IF THE THESE CONDITIONS ARE NOT SET THERE
 WILL BE AN ERROR AT THIS POINT.

```

;B SHIFT          STUCK LOW/HIGH          E15,E19,E2
;B DONE          STUCK HIGH             E15
;TO RX01         STUCK LOW/HIGH         E15
;LOAD            STUCK HIGH             E17,E18
;SELECT 00       STUCK HIGH             E17,E18
;SELECT 02       STUCK LOW             E17,E18,E34
;IN              STUCK LOW             E17,E21
;CMD             STUCK LOW/HIGH         E21
;BUS D15         STUCK HIGH             E40
;RX INIT         STUCK HIGH             E32
;BUS D00 -> D03  STUCK HIGH             E41,E7
;DATA            STUCK LOW             E18,E11
;-----
;-----
;-----
;B INIT          STUCK LOW             E34,E8
;RX BUSY         STUCK LOW             E34,E22,E19
;B DONE         STUCK LOW             E19
;BUS D05         STUCK LOW             E4
;B SER DATA     STUCK LOW/HIGH         E9
;INT ENB(1) H    STUCK HIGH            E37
  
```

;IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
 ;AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
 ;M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
 ;MODULE M7727.

NOTE: ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCIVE TO
 THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
 ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
 HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
 REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
 LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.

M7727 (READ/WRITE CONTROL)

SIGNAL NAME	REASON	POSSIBLE CHIPS
SEL WT PROT	STUCK LOW	E4,E6,E15

```

;:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*
;:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*
;:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*:\*
  
```

```

753 002176 000542
754 002200 004737 002604
755 002204 005723
  
```

```

BR          REBEGIN
MORETESTS: JSR PC, LOCKUP
TONOTHERE: TST (R3)*
  
```

;ADJUST R3 FOR NEXT TEST ADDRESS

```
756 002206 016337 002232 006604 FIRSTTEST: MOV TESTS(R3), PCSCOPE ; EQUIVALENT TO " SCOPE "  
757 002214 013737 006604 002560 MOV PCSCOPE,FAST ;SAVE THE FIRST ADDRESS OF THE TEST  
758 002222 012706 001200 MOV #STACK,SP  
759 002226 000173 002232 JMP @TESTS(R3)  
760 002232 002204 002646 002766 TESTS: TONOTHERE, T1, T2, T3, T4, T5, T6, T7  
761 002240 003310 003446 003612  
762 002246 004036 004170  
763 002252 004166 004254 004244 T10, T11, T12, T13, T14, T15, T16, T17  
764 002260 004320 004444 004636  
765 002266 004776 005244  
766 002272 005510 005534 005720 T20, T21, T22, T23, T24, T25, T26, NOMORETESTS  
767 002300 005776 006044 006076  
768 002306 006110 002312  
769  
770 ; TEST 1 - RXCS TEST PART I / INTERRUPT TEST PART I  
771  
772 ; TEST 2 - INTERRUPT TEST PART II / VECTOR ADDRESS VERIFICATION  
773  
774 ; * TEST 3 - INTERRUPT TEST PART III / PRIORITY LEVEL VERIFICATION PART I  
775  
776 ; * TEST 4 - INTERRUPT TEST PART IV / PRIORITY VERIFICATION PART II  
777  
778 ; TEST 5 - INIT [PROGRAMMED] / RST  
779  
780 ; TEST 6 - FILL BUFFER TRANSFER LENGTH VERIFICATION  
781  
782 ; TEST 7 - EMPTY BUFFER TRANSFER LENGTH AND CONTENT VERIFICATION PART I  
783  
784 ; TEST 10 - EMPTY BUFFER TRANSFER LENGTH AND CONTENT VERIFICATION PART II  
785  
786 ; TEST 11 - FILL/EMPTY BUFFER ALL 0'S  
787  
788 ; TEST 12 - FILL/EMPTY BUFFER ALL 1'S  
789  
790 ; TEST 13 - DRIVE READY VERIFICATION FOR SELECTED DRIVES  
791  
792 ; TEST 14 - ERROR FLAG AND B - CODE VERIFICATION PART I  
793  
794 ; TEST 15 - ERROR FLAG AND B - CODE VERIFICATION PART II  
795 ; /DELETED DATA BIT SETS  
796  
797 ;TEST 16 - ERROR FLAG AND B - CODE VERIFICATION PART III  
798 ; /DELETED DATA BIT CLEARS  
799  
800 ;TEST 17 - ILLEGAL TRACK ERROR AND B - CODE VERIFICATION  
801  
802 ;TEST 20 - SEEK VERIFICATION VIA READ FUNCTION  
803  
804 ;TEST 21 - WRITE TEST  
805  
806 ;TEST 22 - INITIALIZE IMPLIED READ  
807  
808 ;TEST 23 - READ TEST  
809  
810 ;TEST 24 - DATA TRANSFER & VERIFICATION  
811
```

CZRXBFO RX11 INTERFACE TEST
CZRXBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 PAGE 19
PRETEST - INITIALIZE [KEY] PART 1

SEQ 0047

812
813
814
815
816
817
818
819

;TEST 25 - DATA TRANSFER & VERIFICATION VIA DELETED DATA MODE

;TEST 26 - HEAD "HOME" TEST

;THERE ARE NO MORE TESTS

; * NOTE: ON PROCESSORS WITHOUT HARDWARE PROCESSOR STATUS WORDS (PSW)
; THESE TEST WILL NOT BE RUN.


```
820 ;PRINT AN END OF PASS INDICATOR
821
822 ; C - RX11/RX01 TEST PASS OK
823 ; D - RX11/RX01 AND DRIVE TESTING OK
824 ; - - AN ERROR OCCURRED (DURING C OR D)
825
826 ; NOTE: IF BIT 8 OF UNITSEL IS A 1
827 ; THEN AN ERROR HAS OCCURRED FOR THIS PASS
828
829 002312 042777 000100 176666 NOMORETESTS: BIC #BIT6,@RXCS ;CLEAR 'IE' BIT BEFORE NEXT PASS
830 002320 005037 007004 CLR HANGER
831 002324 032737 000400 012752 BIT #BIT8,UNITSEL ;'C' OR 'D' MEANS ERRORLESS PASS.
832 002332 001403 BEQ 1$
833 002334 012737 000055 002562 MOV #'-, MX ; " - " MEANS UN-ERRORLESS PASS
834 002342 005737 002554 1$: TST CCOUNT
835 002346 001002 BNE 3$
836 002350 104401 016162 TYPE, MCRLF
837 002354 005237 002554 3$: INC CCOUNT
838 002360 022737 000110 002554 CMP #72., CCOUNT
839 002366 001002 BNE 4$
840 002370 005037 002554 CLR CCOUNT
841 002374 104401 002562 4$: TYPE, MX
842 002400 104401 006516 TYPE, MABELL
843 002404 005237 002556 2$: INC PASS
844 002410 102775 BVS 2$
845 002412 104406 CKSWR
846 002414 032777 040000 176574 BIT #SW14,@SWR ; AC SW 14 = 1 TO HALT AT END OF PASS
847 002422 001413 BEQ 6$
848 002424 104401 016162 TYPE, MCRLF
849 002430 104401 006753 TYPE, MPASS
850 002434 013737 002556 002446 MOV PASS, 5$
851 002442 004537 015642 JSR R5, SGLDEC
852 002446 000000 5$: OPEN
853 002450 000000 HALT
854 002452 005237 007004 6$: INC HANGER ;WAIT FOR EOP INDICATOR TO BE PRINTED
855 002456 001375 BNE 6$
856 002460 013705 000042 MOV @#42, R5 ;ACT 11 END OF PASS HOOKS
857 002464 001405 BEQ HERE
858 002466 000005 RESET
859 002470 004715 LOGICAL: JSR PC, (R5)
860 002472 000240 NOP
861 002474 000240 NOP
862 002476 000240 NOP
863 002500 000137 002504 HERE: JMP REBEGIN
864
865 002504 042737 000400 012752 REBEGIN: BIC #BIT8,UNITSEL ;CLEAR HARD ERROR INDICATOR
866 002512 013703 001212 MOV DTESTP, R3
867 002516 042703 177740 BIC #177740, R3 ; R3 CONTAINS TEST # 0 TO 26
868 002522 020327 000027 CMP R3, #27
869 002526 103006 BHS 1$
870 002530 006303 ASL R3
871 002532 032777 000040 176446 2$: BIT #40,@RXCS
872 002540 001774 BEQ 2$
873 002542 000621 BR FIRSTTEST
874
875 002544 104401 002564 1$: TYPE, MILTST
```

CZRXBFO RX11 INTERFACE TEST
CZRXBFP11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 ^{K 4} PAGE 21
PRETEST - INITIALIZE [KEY] PART 1

SEQ 0049

876	002550	000137	001232		JMP SA200
877					
878	002554	000000			CCOUNT: 0
879	002556	000000			PASS: 0
880	002560	000000			FAST: 0
881					
882	002562	000103			MX: .ASCIZ 'C'
883					
884	002564	046111	042514	040507	MILTST: .ASCIZ 'ILLEGAL TEST'<15><12>
885	002572	020114	042524	052123	
886	002600	005015	000		
887					
888		002604			.EVEN
889					

CZRXBFO RX11 INTERFACE TEST
CZRXBFP11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 PAGE 22
PRETEST - INITIALIZE [KEY] PART 1

L 4

SEQ 0050

```
890 ; DATA SW 10 = 1 TO HALT AT END OF TEST
891
892 002604 104406 LOCKUP: CKSWR
893 002606 032777 002000 176402 BIT #SW10,@SWR
894 002614 001401 BEQ 1$
895 002616 000000 HALT
896
897 ; DATA SW 12 = 1 TO LOCK SCOPE LOOP ON TEST OK OR NOT
898
899 002620 032777 010000 176370 1$: BIT #SW12,@SWR ;IS LOOP ON TEST SWITCH SET
900 002626 001403 BEQ 2$ ;IF NOT SET GO ON TO NEXT TEST
901 002630 062716 000002 ADD #2,@$P ;IF SET RETURN TO FIRSTTEST
902 002634 000207 RTS PC
903 002636 042737 040100 012752 2$: BIC #40100,UNITSEL ;CLEAR UNIT USED BITS
904 002644 000207 RTS PC
```

```

905 .SBTTL TEST 1 - RXCS TEST PART 1 / INTERRUPT TEST PART 1
906
907 ; THE PURPOSE OF THIS TEST IS TO VERIFY THAT WRITING ALL RXCS WRITABLE BITS TO A 0
908 ; ARE [NOT] WRITTEN TO A 1
909
910 ;THE PROGRAM WRITES THE RXCS = 0
911
912 ;NO INTERRUPTS SHOULD OCCUR
913
914 ;THE RXCS SHOULD REMAIN UNCHANGED = 40 (DONE)
915
916 ;THE RXDB SHOULD = 0
917
918 002646 005077 176334 T1: CLR @ RXCS ; WRITING RXCS TO 0, ALSO WRITES RXDB = 0
919 002652 012700 000040 MOV #40, R0
920 002656 017701 176324 MOV @ RXCS, R1
921 002662 020001 CMP R0, R1
922 002664 001401 BEQ 1$
923
924 ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A
925
926 002666 104000 ERROR ; RXCS NOT = 40 (DONE)
927

```

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
;THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
;BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
;AREAS TO CHECK FOR THE RELEVANT FAULT/S.



IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
ANALYZE THE FOLLOWING AREA/S:

:M7846 (UNIBUS INTERFACE)

SIGNAL NAME	REASON	POSSIBLE CHIPS
:BUS D06	STUCK HIGH	E4
-----	CAN'T SEND DATA	E41
:B TRANSFER REQUEST	STUCK HIGH	E15
:RX INIT	STUCK LOW	E36,E11
:BUS D15	STUCK LOW	E14
:D00(1) H	STUCK HIGH	E22
:RX RUN	STUCK LOW	E18
:BUS D00	STUCK LOW	E7

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

958 002670 104413 1$: SUBSCOPE
959
960

```

```

961 :////////////////////
962 : RXDB AT "DONE"
963 :   7       6       -       -       3       2       1       0       /
964 :
965 :   SEL           WRITE  INIT  PAR           /
966 :   DRIVE DD     PROTECT [DONE]  CRC           /
967 :   RDY                                     (N/A)  /
968 :
969 :////////////////////
970
971
972

```

```

973 002672 005000          CLR R0
974 002674 017701 176310  MOV @ RXDB, R1
975 002700 020100          CMP R1, R0
976 002702 001401          BEQ 3$
977
978           ; (R0) = 0 ; (R1) = ACTUAL RXDB ; (R2) = N/A
979
980 002704 104000          ERROR           ; RXDB NOT = 0
981

```

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

:THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
:THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
:BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
:AREAS TO CHECK FOR THE RELEVANT FAULT/S.

:IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
:ANALYZE THE FOLLOWING AREA/S:

:M7846 (UNIBUS INTERFACE)

SIGNAL NAME	REASON	POSSIBLE CHIPS
BUS D01 -> D03	STUCK HIGH	E7
LOAD	STUCK LOW	E17,E18
-----	NO SACK	E12
-----	CLOCK STUCK HIGH/LOW	E12
IN	STUCK LOW/HIGH	E21
RX BUSY	STUCK HIGH	E22
OUT	STUCK LOW	E22
BUS D00/D02/D03	STUCK HIGH	E4
BUS -> RXCS	STUCK LOW	E40
REG SELECT	STUCK HIGH	E23
B DONE	STUCK HIGH	E19
-----	'B2' INPUT STUCK HIGH	E1

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

1016 002706 104413 3\$: SUBSCOPE

```

1017
1018
1019 002710 013702 001204
1020 002714 010246
1021
1022 002716 012722 002756
1023 002722 012722 000340
1024 002726 012602
1025
1026 002730 005046
1027 002732 012746 002740
1028 002736 000002
1029 002740 000240
1030 002742 000240
1031 002744 012746 000340
1032 002750 012746 002760
1033 002754 000002
1034
1035
1036
1037
1038
1039
1040 002756 104000
1041

```

```

; INTERRUPT TEST PART I ; NO INTERRUPTS SHOULD OCCUR
MOV KRXVEC, R2
MOV R2, -(SP) ;SAVE INTERRUPT VECTOR FOR
;ERROR REPORT
; RX01 VECTOR ADDRESS
MOV #4$, (R2)+
MOV #PR7, (R2)+
MOV (SP)+, R2 ;RESTORE INTERRUPT VECTOR FOR
;ERROR REPORT
; PDP PRIORITY <ON>
CLR -(SP)
MOV #2$, -(SP)
RTI
2$: NOP
NOP
MOV #PR7, -(SP) ; RESET PDP PRIORITY <7> OFF
MOV #5$, -(SP)
RTI
; RETURN TO HERE WITH PDP PRIORITY = 7 IF AN RX01 INTERRUPT
; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
4$: ERROR ; UNEXPECTED RX01 INTERRUPT

```

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
;THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
;BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
;AREAS TO CHECK FOR THE RELEVANT FAULT/S.

```

```

;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
;ANALYZE THE FOLLOWING AREA/S:

```

```

;M7846 (UNIBUS INTERFACE)

```

SIGNAL NAME	REASON	POSSIBLE CHIPS
INT ENB	STUCK HIGH	E36

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

1065 002760 000004
1066 002762 000137 004274
5$: SCOPE
JMP CEXIT ;END OF TEST 1

```



```

1179 003060 000000 CATERR: .WORD 0
1180
1181
1182
1183
1184
1185 003062 000000 DEATH: HALT
1186
1187
1188
1189
1190 003064 104413 OK2GO: SUBSCOPE
1191 003066 005037 003060 CLR CATERR
1192
1193
1194 ; THE RXCS SHOULD = 140 (INTERRUPT ENABLE+DONE)
1195
1196 003072 012700 000140 MOV #140, R0
1197 003076 020077 176104 CMP R0, @ RXCS
1198 003102 001403 BEQ 2$
1199 003104 017701 176076 MOV @ RXCS, R1
1200
1201 ; (R0) = 140 ; (R1) = ACTUAL RXCS ; (R2) = N/A
1202
1203 003110 104000 ERROR ; THE RXCS NOT - 140
1204

```

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

; THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
; THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
; BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
; AREAS TO CHECK FOR THE RELEVANT FAULT/S.
;
; IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
; ANALYZE THE FOLLOWING AREA/S:

```

; M7846 (UNIBUS INTERFACE)

SIGNAL NAME	REASON	POSSIBLE CHIPS
; BUS D07	STUCK HIGH	E4
; BUS D06	STUCK HIGH	E1

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

1229 003112 104413 2$: SUBSCOPE
1230
1231
1232 ; TEST 2 - CONT'D
1233
1234 ; THE PURPOSE OF THIS TEST IS TO VERIFY THAT BIT 6 OF THE RXCS (INTERRUPT ENABLE)

```

```
1235 ; CAN BE CLEARED AFTER IT WAS KNOWN TO BE SET
1236
1237 003114 013702 001204      MOV KRXVEC, R2
1238 003120 010246              MOV R2, -(SP)                ;SAVE INTERRUPT VECTOR FOR
1239                               ;ERROR REPORT
1240 003122 012722 003212      MOV #4$, (R2)+                ; RX11 VECTOR ADDRESS
1241 003126 012722 000340      MOV #PR7, (R2)+
1242 003132 012602              MOV (SP)+, R2                ;RESTORE INTERRUPT VECTOR FOR
1243                               ;ERROR REPORT
1244 003134 042777 000100 176044 BIC #BIT6, @ RXCS            ; CLEAR THE RX11 INTERRUPT ENABLE BIT
1245
1246 ; THE RXCS SHOULD = 40 (DONE)
1247
1248 003142 012700 000040      MOV #40, R0
1249 003146 020077 176034      CMP R0, @ RXCS
1250 003152 001403              BEQ 3$
1251 003154 017701 176026      MOV @ RXCS, R1
1252
1253 ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A
1254
1255 003160 104000              ERROR                        ; RXCS NOT = 40
1256 003162 104413              3$: SUBSCOPE
1257 ; NO RX11 INTERRUPTS SHOULD OCCUR [YET]
1258
1259 003164 005046              CLR -(SP)                    ; PDP PRIORITY <ON>
1260 003166 012746 003174      MOV #10$, -(SP)
1261 003172 000002              RTI
1262 003174 000240              10$: NOP
1263 003176 000240              NOP
1264 003200 012746 000340      MOV #PR7, -(SP)              ; PDP PRIORLITY <OFF> 7
1265 003204 012746 003214      MOV #11$, -(SP)
1266 003210 000002              RTI
1267
1268 ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN UNEXPECTED RX11 INTERRUPT
1269 ; WHILE CLEARING THE RX11 INTERRUPT ENABLE BIT 6
1270
1271 ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
1272
1273 003212 104000              4$: ERROR                    ; UNEXPECTED RX11 INTERRUPT
1274 003214 104413              11$: SUBSCOPE
1275
1276 ; AN RX11 INTERRUPT SHOULD OCCUR [NOW]
1277
1278 003216 013702 001204      MOV KRXVEC, R2
1279 003222 010246              MOV R2, -(SP)                ;SAVE INTERRUPT VECTOR FOR
1280                               ;ERROR REPORT
1281 003224 012722 003274      MOV #5$, (R2)+                ; RX11 VECTOR ADDRESS
1282 003230 012722 000340      MOV #PR7, (R2)+
1283 003234 012602              MOV (SP)+, R2                ;RESTORE INTERRUPT VECTOR FOR
1284                               ;ERROR REPORT
1285 003236 005046              CLR -(SP)                    ; PDP PRIORITY <ON>
1286 003240 012746 003246      MOV #12$, -(SP)
1287 003244 000002              RTI
1288 003246 052777 000100 175732 12$: BIS #BIT6, @ RXCS            ; SET RX11 INTERRUPT ENABLE BIT
1289 003254 000240              NOP
1290 003256 000240              NOP
```

CZRXBFO RX11 INTERFACE TEST
CZRXBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 ^{6 5} PAGE 30
TEST 2 - INTERRUPT TEST PART II / VECTOR ADDRESS VERIFICATION

SEQ 0058

1291	003260	012746	000340		MOV #PR7,-(SP)
1292	003264	012746	003272		MOV #13\$,-(SP)
1293	003270	000002			RTI
1294					
1295					; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
1296					
1297	003272	104000			13\$: ERROR ; NO RX11 INTERRUPT OCCURRED
1298					
1299					; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN RX01 INTERRUPT
1300					
1301	003274	000004			5\$: SCOPE
1302	003276	042777	000100	175702	BIC #BIT6, @ RXCS ; CLEAR THE RX11 INTERRUPT ENABLE
1303					
1304	003304	000137	004274		JMP CEXIT ;END OF TEST 2

.SBTTL TEST 3 - INTERRUPT TEST PART III / PRIORITY LEVEL VERIFICATION PART I

; THE PURPOSE OF THIS TEST IS TO VERIFY THE PRIORITY OF THE RX11 INTERRUPT REQUEST LINE
; THE PROGRAM SETS THE PDP PRIORITY TO 1 LESS THAN THE DEVICE LEVEL
; (DEVICE LEVEL SPECIFIED BY CONTENTS OF LOCATION 'BRLEV:' -- NORMALLY 5)
; AN RX01 INTERRUPT SHOULD OCCUR
; IF NO INTERRUPT OCCURS THEN THE PRIORITY LEVEL OF THE RX11 IS [NOT] = NORMAL
; DEVICE LEVEL OF 5 OR THE DEVICE LEVEL AS SPECIFIED BY THE CONTENTS OF
; LOCATION 'BRLEV:' WHICH MAY HAVE BEEN CHANGED BY THE USER BEFORE PROGRAM
; EXECUTION, BUT MAYBE SOME VALUE LESS THAN THE CONTENTS OF LOCATION 'BRLEV:'.
; NOTE: IF THERE IS NO HARDWARE 'PSW' THIS TEST WILL BE SKIPPED.

```
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317 003310 005001 T3: CLR R1 ;INDICATOR TO CPU PRIORITY  
1318 ;ROUTINE TO DROP CPU PRIORITY  
1319 ;TO 1 LESS THAN DEVICE LEVEL  
1320 003312 013702 001204 MOV KRXVEC, R2  
1321 003316 010246 MOV R2, -(SP) ;SAVE INTERRUPT VECTOR FOR  
1322 ;ERROR REPORT  
1323 003320 012722 003432 MOV #1$, (R2)+ ;RX01 VECTOR ADDRESS  
1324 003324 012722 000340 MOV #PR7, (R2)+  
1325 003330 012602 MOV (SP)+, R2 ;RESTORE INTERRUPT VECTOR FOR  
1326 ;ERROR REPORT  
1327 003332 013746 000004 MOV 4, -(SP) ;SAVE 'BUSERR' TIMEOUT 'PC'  
1328 003336 012737 003354 000004 MOV #2$, 4 ;SET TIMEOUT VECTOR  
1329 003344 012737 000200 177776 MOV #PR4, PSW ;SET LEVEL TO 4 IF 'PSW' EXISTS  
1330 003352 000404 BR 3$ ;GO TO RESET VECTOR 4 & DO TEST  
1331 003354 022626 2$: CMP (SP)+, (SP)+ ;CORRECT STACK FROM BUS TIMEOUT  
1332 003356 012637 000004 MOV (SP)+, 4 ;RESTORE TIMEOUT VECTOR TO 'BUSERR'  
1333 003362 000427 BR 4$ ;NO HARDWARE PSW - SKIP THIS TEST  
1334 003364 012637 000004 3$: MOV (SP)+, 4 ;RESET TIMEOUT VECTOR TO 'BUSERR'  
1335 003370 004737 006226 JSR PC, CPUPRI ;CALCULATE PRIORITY LEVEL OF CPU  
1336 ;BASED ON CURRENT DEVICE PRIORITY  
1337 ;LEVEL RESIDING IN LOC. 'BRLEV'  
1338 003374 010046 MOV R0, -(SP) ;;PUT NEW PS ON STACK  
1339 003376 012746 003404 MOV #64$, -(SP) ;;PUT NEW PC ON STACK  
1340 003402 000002 RTI ;;POP NEW PC AND PS  
1341 003404 64$:  
1342 003404 052777 000100 175574 BIS #BIT6, @ RXCS ; SET THE RX01 INTERRUPT ENABLE  
1343 003412 000240 NOP  
1344 003414 000240 NOP  
1345 003416 013746 000340 MOV PR7, -(SP) ;;PUT NEW PS ON STACK  
1346 003422 012746 003430 MOV #65$, -(SP) ;;PUT NEW PC ON STACK  
1347 003426 000002 RTI ;;POP NEW PC AND PS  
1348 003430 65$:  
1349  
1350 ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A  
1351  
1352 003430 104000 ERROR ;PRIORITY LEVEL IS NOT = CONTENTS  
1353 ;OF 'BRLEV:' (NORMALLY 5) BUT  
1354 ;MAYBE SOME VALUE LESS THAN THE  
1355 ;THE CURRENT CONTENTS OF 'BRLEV:'  
1356  
1357 ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN RX01 INTERRUPT  
1358  
1359 003432 000004 1$: SCOPE  
1360 003434 042777 000100 175544 BIC #BIT6, @ RXCS ; CLEAR THE RX11 INTERRUPT ENABLE
```

```
1361 003442 000137 004274 4$: JMP CEXIT ;END OF TEST 3
1362
1363 .SBTTL TEST 4 - INTERRUPT TEST PART IV / PRIORITY VERIFICATION PART II
1364
1365 ; THE PURPOSE OF THIS TEST IS TO VERIFY THE PRIORITY OF THE RX11 INTERRUPT REQUEST LINE
1366 ; THE PROGRAM SETS THE PDP PRIORITY = THE DEVICE LEVEL, (NORMALLY 5 OR THE CONTENTS OF L
1367 ; NO RX01 INTERRUPTS SHOULD OCCUR
1368 ; IF AN INTERRUPT DOES OCCUR THEN THE PRIORITY LEVEL OF THE RX11 IS [NOT]
1369 ; = THE NORMAL DEVICE LEVEL OF 5, OR WHATEVER IS THE VALUE IN LOCATION 'BRLEV:'
1370 ; BUT MAYBE SOME VALUE GREATER THAN THE CONTENTS OF LOC. 'BRLEV:'
1371 ; NOTE: IF THERE IS NO HARDWARE 'PSW' THIS TEST WILL BE SKIPPED.
1372
1373 003446 005001 T4: CLR R1 ;INDICATOR TO CPU PRIORITY ROUTINE
1374 ;TO DROP CPU PRIORITY 1 LEVEL
1375 ;LESS THAN THE DEVICE LEVEL
1376 003450 013702 001204 MOV KRXVEC, R2
1377 003454 010246 MOV R2, -(SP) ;SAVE INTERRUPT VECTOR FOR
1378 ;ERROR REPORT
1379 003456 012722 003574 MOV #1$, (R2)+ ;RX01 VECTOR ADDRESS
1380 003462 012722 000340 MOV #PR7, (R2)+
1381 003466 012602 MOV (SP)+, R2 ;RESTORE INTERRUPT VECTOR FOR
1382 ;ERROR REPORT
1383 003470 052701 000200 BIS #BIT7, R1 ;SET INDICATOR TO CPU PRIORITY
1384 ;ROUTINE TO SET CPU PRIORITY LEVEL
1385 ;TO THE SAME LEVEL AS THE DEVICE
1386 003474 013746 000004 MOV 4, -(SP) ;SAVE 'BUSERR' TIMEOUT 'PC'
1387 003500 012737 003516 000004 MOV #3$, 4 ;SET TIMEOUT VECTOR
1388 003506 012737 000240 177776 MOV #PR5, PSW ;SET LEVEL TO 5 IF 'PSW' EXISTS
1389 003514 000404 BR 4$ ;GO ON TO RESET VECTOR & DO TEST
1390 003516 022626 3$: CMP (SP)+, (SP)+ ;CORRECT STACK FROM BUS TIMEOUT
1391 003520 012637 000004 MOV (SP)+, 4 ;RESTORE TIMEOUT VECTOR TO 'BUSERR'
1392 003524 000430 BR 5$ ;NO HARDWARE PSW - SKIP THIS TEST
1393 003526 012637 000004 4$: MOV (SP)+, 4 ;RESET TIMEOUT VECTOR TO BUSERR
1394 003532 004737 006226 JSR PC, CPUPRI ;CALCULATE CPU PRIORITY LEVEL TO
1395 ;BE THE SAME AS THE DEVICE LEVEL
1396 ;I.E. - SAME AS CONTENTS OF LOC.
1397 ; 'BRLEV'
1398 003536 010046 MOV R0, -(SP) ;;PUT NEW PS ON STACK
1399 003540 012746 003546 MOV #64$, -(SP) ;;PUT NEW PC ON STACK
1400 003544 000002 RTI ;;POP NEW PC AND PS
1401 003546 64$:
1402 003546 052777 000100 175432 BIS #BIT6, @RXCS ;SET RX01 INTERRUPT ENABLE
1403 003554 000240 NOP
1404 003556 000240 NOP
1405 003560 013746 000340 MOV PR7, -(SP) ;;PUT NEW PS ON STACK
1406 003564 012746 003572 MOV #65$, -(SP) ;;PUT NEW PC ON STACK
1407 003570 000002 RTI ;;POP NEW PC AND PS
1408 003572 65$:
1409 003572 000401 BR 2$
1410
1411 ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN RX01 INTERRUPT
1412
1413 ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
1414
1415 003574 104000 1$: ERROR ;PRIORITY LEVEL NOT = TO CONTENTS
1416 ;OF LOCATION 'BRLEV:' (NORMALLY 5)
```

1417
1418
1419
1420 003576 0000C4
1421 003600 042777 000100 175400
1422 003606 000137 004274
2\$: SCOPE
5\$: BIC #BIT6, @ RXCS
JMP CEXIT

;BUT MAYBE SOME VALUE GREATER THAN
;THE CONTENTS SPECIFIED BY LOC.
;'BRLEV:'
; CLEAR THE RX01 INTERRUPT ENABLE
;END OF TEST 4


```

1479          :          RDY                      (N/A) /
1480          :
1481          : ///////////////////////////////////////////////////////////////////
1482          :
1483          : THE RXDB SHOULD = 4, 104, IF TESTING UNIT 1
1484          : OR 204, OR 304 IF TESTING UNIT 0 (SEL. DRIVE RDY. BIT SET)
1485
1486 003646 012700 000204          MOV #204,R0
1487 003652 017702 175332          MOV @RXDB,R2
1488 003656 010201          MOV R2,R1
1489 003660 042701 000100          BIC #BIT6,R1          ;CLEAR DELETED DATA BIT
1490 003664 105737 012752          TSTB UNITSEL          ;WAS UNIT 0 SELECTED
1491 003670 100404          BMI 3$
1492 003672 042701 000200          BIC #BIT7,R1          ;UNIT 0 WAS NOT SELECTED
1493 003676 042700 000200          BIC #BIT7,R0          ;CLEAR UNIT 0 RDY BITS
1494 003702 020100          3$: CMP R1, R0
1495 003704 001401          BEQ 4$
1496
1497          : (R0) = 4, OR 204
1498          : (R1) = ACTUAL RXDB MINUS DELETED DATA BIT#6
1499          : (R2) = ACTUAL RXDB
1500
1501 003706 104000          ERROR          ; RXDB NOT = 4, OR 104, OR 204, OR 304
1502

```

```

/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\
/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\
/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\

```

```

;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
;THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
;BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
;
;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
;ANALYZE THE FOLLOWING AREA/S:

```

```

:M7846 (UNIBUS INTERFACE)

```

SIGNAL NAME	REASON	POSSIBLE CHIPS
:RX INIT	STUCK HIGH	E11,E36
:IN	STUCK HIGH	E11
:BUS DOS	STUCK LOW	E1
:RX DATA	STUCK LOW	E14
:-----	SACK FLOP CLOCK LOCKED LOW	E9
:B SER DATA	STUCK LOW	E9
:DONE	STUCK HIGH	E22
:-----	LOAD PULSE STUCK LOW	E3

```

/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\
/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\
/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\/*\

```

```

1533 003710 104413          4$: SUBSCOPE

```



```

1534 : TEST 5 CONT'D - RXCS TEST PART II / RST
1535
1536 : THE PURPOSE OF THIS TEST IS TO VERIFY THE RST COMMAND #12+GO
1537 : AND THAT THE DONE BIT IS CLEARED AFTER THE FUNCTION IS ISSUED.
1538
1539 003712 012777 000013 175266 MOV #13, @ RXCS ; RST COMMAND
1540 003720 032777 000040 175260 BIT #BITS, @ RXCS ; TEST DONE BIT (SHOULD = 0)
1541 003726 001404 BEQ 5$
1542 003730 017701 175252 MOV @ RXCS, R1
1543 003734 005000 CLR R0
1544
1545 : (R0) = 0 ; (R1) = ACTUAL RXCS ; (R2) = N/A
1546
1547 003736 104000 ERROR ; DONE BIT NOT = 0
1548 003740 104413 5$: SUBSCOPE
1549 003742 004737 006622 9$: JSR PC, SDN ; WAIT FOR DONE FLAG
1550 003746 000775 BR 9$
1551 003750 012700 000040 MOV #40, R0 ; PROGRAM EXPECTS RXCS = 40 (DONE)
1552 003754 017701 175226 MOV @ RXCS, R1 ; ACTUAL RXCS
1553 003760 020100 CMP R1, R0
1554 003762 001401 BEQ 10$
1555
1556 : (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A
1557
1558 003764 104000 ERROR ; RXCS NOT = 40
1559

```

```

; /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\:
; /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\:
; /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\:

```

THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL AREAS TO CHECK FOR THE RELEVANT FAULT/S.

IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS ANALYZE THE FOLLOWING AREA/S:

M7846 (UNIBUS INTERFACE)

SIGNAL NAME	REASON	POSSIBLE CHIPS
-----	PARITY FLOP CLOCK LOCKED HIGH	E2
-----	PARITY FLOP 'Q' OUTPUT LOCKED HIGH	E2
RX DATA	STUCK LOW	E6

```

; /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\:
; /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\:
; /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\: /*\:

```

```

1586 003766 104413 10$: SUBSCOPE
1587
1588 : THE RXDB SHOULD = 200 (IF DRIVE 0 IS READY), OR 0 IF UNIT 0 IS NOT SELECTED
1589 : MAYBE 300 (IF DRIVE 0 IS READY AND SECTOR 1 WAS WRITTEN WITH DELETED DATA)

```

```
1590
1591 003770 017702 175214          MOV @RXDB, R2          ; ACTUAL RXDB
1592 003774 010201          MOV R2, R1
1593 003776 042701 000100          BIC #BIT6, R1         ; CLEAR N/A DELETED DATA BIT
1594 004002 012700 000200          MOV #200, R0          ; EXPECT UNIT 0 RDY SET
1595 004006 105737 012752          TSTB UNITSEL
1596 004012 100403          BMI 11$
1597 004014 042701 000200          BIC #BIT7, R1         ; UNIT 0 NOT SELECTED
1598 004020 005000          CLR R0                ; DISREGUARD RDY BITS
1599 004022 020100
1600 004024 001401
1601
1602
1603
1604
1605
1606 004026 104000
1607 004030 000004
1608 004032 000137 004274

11$: CMP R1, R0
    BEQ 12$

; (R0) = 0 OR 200
; (R1) = ACTUAL RXDB MINUS DELETED DATA BIT#6
; (R2) = ACTUAL RXDB

12$: ERROR          ; RXDB NOT = 200, OR NOT = 0
    SCOPE
    JMP CEXIT       ; END OF TEST 5
```


.SBTTL TEST 14 - ERROR FLAG AND B-CODE VERIFICATION PART I

;THE PURPOSE OF THIS TEST IS TO VERIFY THAT TRYING TO READ A NON-EXISTANT
;SECTOR WILL CAUSE AN ERROR, AND THE CORRECT ERROR CODE WILL BE PUT
;INTO THE RXDB WHEN THE B STATUS IS READ.

;THIS SECTION INSURES THAT ONLY 2 TR FLAGS WERE REQUIRED TO TAKE THE
;DISKETTE ADDRESS, AND THAT AN ERROR DOES EXIST.

1898									
1899									
1900									
1901									
1902									
1903									
1904									
1905									
1906									
1907	004444	005002			T14:	CLR R2			
1908	004446	005000				CLR R0			
1909	004450	105737	012752			TSTB UNITSEL		;IS UNIT 0 SELECTED	
1910	004454	100004				BPL 10\$			
1911	004456	012777	000007	174522		MOV #7,@RXCS		;SET READ SECTOR FUNCTION AND GO	
1912	004464	000403				BR 11\$			
1913	004466	012777	000027	174512	10\$:	MOV #27,@RXCS		;SEND READ FUNCTION TO UNIT 1	
1914	004474	004737	005134		11\$:	JSR PC,ILLADR		;SEND THE BAD SECTOR ADDRESS	
1915	004500	000406				BR 1\$;PREMATURE ERROR CONDITION	
1916	004502	012700	100040			MOV #100040,R0		;EXPECT ERROR AND DONE BITS	
1917	004506	017701	174474			MOV @RXCS,R1		;SAVE THE RXCS	
1918	004512	020001				CMP R0,R1			
1919	004514	001401				BEQ 2\$			
1920									
1921									
1922									
1923	004516	104000			1\$:	ERROR			
1924									

;R0 = 100040 ; R1 = ACTUAL RXCS ; R2 = # OF TR FLAGS

;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
;THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
;BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
;AREAS TO CHECK FOR THE RELEVANT FAULT/S.

;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
;ANALYZE THE FOLLOWING AREA/S:

;M7846 (UNIBUS INTERFACE)

SIGNAL NAME	REASON	POSSIBLE CHIPS
BUS D15	STUCK HIGH	E14,E9,E40,E24
RX RUN	STUCK LOW	E41

;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

1949	004520	104413			2\$:	SUBSCOPE			
1950									
1951									
1952									
1953									

;T14 CONT. - THIS SECTION TESTS THAT NO PARITY OR CRC ERROR OCCURRED
;ON PREVIOUS FUNCTION.

.SBTTL TEST 15 - ERROR FLAG AND B-CODE VERIFICATION PART II

:THIS TEST VERIFIES THAT TRYING TO WRITE, USING DELETED DATA MODE, ON A
:NON-EXISTANT SECTOR WILL PRODUCE AN ERROR AND THE CORRECT B-CODE IS PRODUCED
:THIS SECTION SENDS OUT AN ILLEGAL SECTOR ADDRESS AND EXPECTS AN ERROR
: NOTE TEST 14 MUST BE RUN BEFORE THIS TEST

```
2033
2034
2035
2036
2037
2038
2039
2040 004636 005000 T15: CLR R0
2041 004640 005002 CLR R2
2042 004642 105737 012752 TSTB UNITSEL ;WAS UNIT 0 SELECTED
2043 004646 100004 BPL 10$
2044 004650 012777 000015 174330 MOV #15,@RXCS ;SET WRITE DELETED DATA FUNCTION
2045 004656 000403 BR 11$
2046 004660 012777 000035 174320 10$: MOV #35,@RXCS ;SEND WTR DD FUNCTION TO UNIT 1
2047 004666 004737 005134 11$: JSR PC,ILLADR ;SEND THE ILLEGAL SECTOR ADDRESS
2048 004672 000406 BR 1$ ;PREMATURE ERROR CONDITION
2049 004674 012700 100040 MOV #100040,R0 ;EXPECT ERROR AND DONE FLAGS
2050 004700 017701 174302 MOV @RXCS,R1
2051 004704 020001 CMP R0,R1
2052 004706 001401 BEQ 2$
2053
2054 ;R0 = 100040 ;R1 = ACTUAL RXCS ; R2 = # OF TR FLAGS
2055
2056 004710 104000 1$: ERROR ;RXCS NOT = 100040
2057 004712 104413 2$: SUBSCOPE
2058
2059 ;T15 CONT. - THIS SECTION TESTS THAT THERE IS NO PARITY, CRC ERROR
2060 ;AND THAT THE DELETED DATA BIT IS SET.
2061
2062 004714 005002 CLR R2
2063 004716 012700 000100 MOV #100,R0 ;EXPECT DELETED DATA BIT TO BE SET
2064 004722 017701 174262 MOV @RXDB,R1
2065 004726 020001 CMP R0,R1
2066 004730 001401 BEQ 3$
2067
2068 ; R0 = 100 ; R1 = ACTUAL RXDB ; R2 = N/A
2069 004732 104000 ERROR ;DELETED DATA NOT SET OR OTHER ERRORS
2070 004734 104413 3$: SUBSCOPE
2071
2072 ;T15 CONT. - THIS SECTION TESTS FOR THE B-CODE FOR ILLEGAL SECTOR.
2073
2074 004736 012777 000017 174242 MOV #17,@RXCS ;SET READ STATUS B FUNCTION
2075 004744 004737 006622 4$: JSR PC,SDN ;WAIT FOR DONE FLAG
2076 004750 000775 BR 4$
2077 004752 012700 000070 MOV #70,R0
2078 004756 017701 174226 MOV @RXDB,R1
2079 004762 020001 CMP R0,R1
2080 004764 001401 BEQ 5$
2081
2082 ; R0 = 70 ; R1 = ACTUAL B-CODE ; R2 = N/A
2083 004766 104000 ERROR ;RXDB NOT = 70
2084 004770 000004 5$: SCOPE
2085 004772 000137 004306 JMP DEXIT ;END OF TEST 15
```

.SBTTL TEST 16 - ERROR FLAG AND B-CODE VERIFICATION PART 111

: THIS TEST VERIFIES THAT A WRITE FUNCTION TO A NON-EXISTANT SECTOR WILL
: PRODUCE AN ERROR AND A B-CODE OF 70. THE DELETED DATA BIT SHOULD ALSO BE CLEARED
: THIS SECTION TRANSFERS AN ILLEGAL SECTOR ADDRESS FOR A WRITE FUNCTION
: NOTE TEST 14 MUST BE RUN BEFORE THIS TEST

2086									
2087									
2088									
2089									
2090									
2091									
2092									
2093	004776	005000			T16:	CLR R0			
2094	005000	005002				CLR R2			
2095	005002	105737	012752			TSTB UNITSEL			; WAS UNIT 0 SELECTED
2096	005006	100004				BPL 10\$			
2097	005010	012777	000005	174170		MOV #5,@RXCS			; SET THE WRITE FUNCTION
2098	005016	000403				BR 11\$			
2099	005020	012777	000025	174160	10\$:	MOV #25,@RXCS			; SEND WRITE FUNCTION TO UNIT 1
2100	005026	004737	005134		11\$:	JSR PC,ILLADP			; SEND THE ILLEGAL ADDRESS
2101	005032	000406				BR 1\$; PREMATURE ERROR CONDITION
2102	005034	012700	100040			MOV #100040,R0			; EXPECT ERROR AND DONE BITS SET
2103	005040	017701	174142			MOV @RXCS,R1			
2104	005044	020001				CMP R0,R1			
2105	005046	001401				BEQ 2\$			
2106									
2107									
2108									
2109	005050	104000			1\$:	ERROR			
2110	005052	104413			2\$:	SUBSCOPE			
2111									
2112									
2113									
2114	005054	005002				CLR R2			
2115	005056	005000				CLR R0			; NO BITS SHOULD BE SET IN THE RXDB
2116	005060	017701	174124			MOV @RXDB,R1			
2117	005064	020001				CMP R0,R1			
2118	005066	001401				BEQ 3\$			
2119									
2120									
2121	005070	104000							
2122	005072	104413							
2123									
2124									
2125									
2126	005074	012777	000017	174104		MOV #17,@RXCS			; SET READ STATUS B FUNCTION
2127	005102	004737	006622		4\$:	JSR PC,SDN			; WAIT FOR DONE FLAG
2128	005106	000775				BR 4\$			
2129	005110	012700	000070			MOV #70,R0			
2130	005114	017701	174070			MOV @RXDB,R1			
2131	005120	020001				CMP R0,R1			; IS B-CODE = 70
2132	005122	001401				BEQ 5\$; YES, CONTINUE
2133									
2134									
2135	005124	104000							
2136	005126	000004							
2137	005130	000137	004306		5\$:	SCOPE			
						JMP DEXIT			; END OF TEST 16

```
2138
2139           ;GENERATE AN ILLEGAL SECTOR ADDRESS
2140
2141 005134 004737 006606      ILLADR: JSR PC,STR           ;LOOK FOR A TR FLAG
2142 005140 000402           BR 2$                ;NO TR FLAG, IS DONE SET
2143 005142 005202           INC R2                ;TR FLAG COUNTER
2144 005144 000404           BR 3$
2145 005146 004737 006622      2$: JSR PC,SDN           ;LOOK FOR DONE FLAG
2146 005152 000770           BR ILLADR          ;NOT DONE RECHECK TR FLAG
2147 005154 000430           BR 1$                ;DONE IS SET TOO EARLY GO TO ERROR
2148 005156 005077 174026      3$: CLR @RXDB          ;0 SECTOR ADDRESS (ILLEGAL)
2149 005162 004737 006606      7$: JSR PC,STR           ;LOOK FOR SECOND TR FLAG
2150 005166 000402           BR 5$                ;NOT TR, IS IT DONE
2151 005170 005202           INC R2
2152 005172 000404           BR 6$                ;TR FLAG SEND TRACK ADDRESS
2153 005174 004737 006622      5$: JSR PC,SDN           ;LOOK FOR DONE FLAG
2154 005200 000770           BR 7$                ;NOT DONE, RECHECK TR FLAG
2155 005202 000415           BR 1$                ;DONE TOO SOON GO TO ERROR
2156 005204 005077 174000      6$: CLR @RXDB          ;SEND TRACK ADDRESS OF 0
2157 005210 004737 006606      11$: JSR PC,STR          ;ARE THERE ANY MORE TR FLAGS
2158 005214 000402           BR 10$             ;NO, LOOK FOR DONE
2159 005216 005202           INC R2                ;YES
2160 005220 000406           BR 1$                ;TOO MANY TR FLAGS OR MICROCONTROLLER
2161
2162 005222 004737 006622      10$: JSR PC,SDN          ;DID NOT DETECT THE ERROR
2163 005226 000770           BR 11$             ;LOOK FOR DONE FLAG
2164 005230 062716 000002      ADD #2,@SP        ;NOT DONE RETEST TR FLAG
2165 005234 000207           RTS PC
2166 005236 017701 173744      4$: MOV @RXCS,R1
2167 005242 000774           BR 4$
```

2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207

```

.SBTTL TEST 17 - ILLEGAL TRACK ERROR VERIFICATION
;THIS TEST VERIFIES THAT IF A TRACK ADDRESS LARGER THAN 114 (OCTAL) IS
;ACCESSED, AN ERROR CONDITION WILL EXIST, AND A B-CODE WILL = 40
T17: CLR R2
      CLR R0
      MOV #7,@RXCS ;SET READ FUNCTION
3$: JSR PC,STR ;LOOK FOR TR FLAG
     BR 1$ ;NO TR FLAG CHECK DONE
     BR 2$
1$: JSR PC,SDN
     BR 3$
     MOV @RXCS,R1 ;DONE OCCURRED TOO SOON SET UP FOR ERROR
     MOV @RXDB,R2
     BR 4$
2$: MOV #1,@RXDB ;SEND LEGAL SECTOR ADDRESS
5$: JSR PC,STR ;LOOK FOR TR FLAG
     BR 6$
     BR 7$
6$: JSR PC,SDN
     BR 5$
     MOV @RXCS,R1 ;DONE SET TOO EARLY
     MOV @RXDB,R2
     BR 4$
7$: MOV #115,@RXDB ;SEND ILLEGAL TRACK ADDRESS
10$: JSR PC,SDN ;WAIT FOR DONE ON THE ERROR
      BR 10$
      MOV #100040,R0 ;EXPECT ERROR AND DONE SET
      MOV @RXCS,R1
      CMP R0,R1
      BEQ 11$
4$: ERROR ;DONE SET TOO SOON OR NO ERROR OCCURRED

```

```

;TWO ERROR CONDITIONS TO REPORT
;IF R0 = 0 THEN R1 = RXCS ;R2 = RXDB ON A DONE TOO SOON ERROR
;IF R0 = 100040 THEN R1 = ACTUAL RXCS ; R2 = N/A

```

```

; /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\
; /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\
; /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\ /*\

```

```

;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
;THIS ERROR REPORT WAS GENERATED. THE INFORMATION SHOWN IS
;BASED ON FAULT INSERTION RESULTS, AND SHOULD PROVIDE LOGICAL
;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
;
;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
;ANALYZE THE FOLLOWING AREA/S:

```

```

;M7846 (UNIBUS INTERFACE)

```

```

; SIGNAL NAME REASON POSSIBLE CHIPS

```

 INT ENB FLOP CLOCK LOCKED HIGH E40

./:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/
 ./:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/.
 ./:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/:.*/.

2231 005376 104413 11\$: SUBSCOPE
 2232
 2233 :T17 CONT. - TEST THAT THERE WERE NO OTHER ERRORS THAN THE ILLEGAL TRACK ERROR EXPECTED
 2234 :AND THAT THE DELETED DATA BIT WAS CLEARED BY TEST 16.
 2235
 2236 005400 005000 CLR R0 ;DD BIT CLEARED IN TEST 16 SO RXDB = 0
 2237 005402 005002 CLR R2
 2238 005404 017701 173600 MOV @RXDB,R1
 2239 005410 020001 CMP R0,R1
 2240 005412 001401 BEQ 12\$
 2241
 2242 ; R0 = 0 ; R1 = ACTUAL RXDB ; R2 = N/A
 2243
 2244 005414 104000 ERROR
 2245 005416 104413 12\$: SUBSCOPE
 2246
 2247 :T17 CONT. - VERIFIES THAT READ STATUS B HAS NO ERRORS
 2248
 2249 005420 012777 000017 173560 MOV #17,@RXCS ;SET READ STATUS B FUNCTION
 2250 005426 004737 006622 JSR PC,SDN ;WAIT FOR DONE FLAG
 2251 005432 000775 BR 13\$
 2252 005434 005777 173546 TST @RXCS ;WAS THERE AN ERROR ON THIS FUNCTION
 2253 005440 100007 BPL 14\$;NO
 2254 005442 012700 000040 MOV #40,R0
 2255 005446 017701 173534 MOV @RXCS,R1
 2256 005452 017702 173532 MOV @RXDB,R2
 2257
 2258 ; R0 = 40 (DONE) ; R1 = ACTUAL RXCS ; R2 = ACTUAL RXDB
 2259
 2260 005456 104000 ERROR
 2261 005460 104413 14\$: SUBSCOPE ;THERE WAS AN ERROR NO READ STATUS B
 2262 005462 005002 CLR R2
 2263 005464 012700 000040 MOV #40,R0 ;B-CODE FOR ILLEGAL TRACK ADDRESS
 2264 005470 017701 173514 MOV @RXDB,R1
 2265 005474 020001 CMP R0,R1
 2266 005476 001401 BEQ 15\$;B-CODE IS CORRECT
 2267
 2268 ; R0 = 40 ; R1 = ACTUAL B-CODE ; R2 = N/A
 2269
 2270 005500 104000 ERROR
 2271 005502 000004 15\$: SCOPE
 2272 005504 000137 004306 JMP DEXIT ;END OF TEST 17

CZRXBFO RX11 INTERFACE TEST
CZRXBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23^{D 7} PAGE 53
TEST 21 - WRITE TEST

SEQ 0081

2329 005616 000771
2330 005620 012737 000005 007730 28:
2331 005626 004737 005636
2332 005632 000137 004306

BR 18
MOV #5,FUNCTION
JSR PC,T21X
JMP DEXIT

;SET WRITE FUNCTION
;GO ISSUE THE COMMAND
;END OF TEST 21

```

2333 005636 004737 007662      T21X: JSR PC,COMMWORD
2334 005642 004737 006622      3$: JSR PC,SDN
2335 005646 000775              BR 3$
2336 005650 005777 173332      TST @RXCS ;IS THE ERROR BIT SET
2337 005654 100013              BPL 4$ ;NO ERRORS
2338 005656 017701 173326      MOV @RXDB,R1 ;SAVE STATUS A
2339 005662 012777 000017 173316 MOV #17,@RXCS ;SET READ STATUS B FUNCTION
2340 005670 004737 006622      5$: JSR PC,SDN
2341 005674 000775              BR 5$
2342 005676 017702 173306      MOV @RXDB,R2 ;SAVE STATUS B
2343
2344 ;RO = ADDR OF LAST DATA BYTE ; R1 = STATUS A ; R2 = STATUS B
2345
2346 005702 104000              ERROR
2347

```

```

; /*:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

;IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
 ;AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
 ;M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
 ;MODULE M7727.

NOTE: ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCTIVE TO
 THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
 ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
 HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
 REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
 LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.

M7727 (READ/WRITE CONTROL)

SIGNAL NAME	REASON	POSSIBLE CHIPS
WT DATA	STUCK LOW/HIGH	E5
WT GATE	STUCK LOW	E13

```

; /*:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

2376 005704 104413      4$: SUBSCOPE
2377 005706 005000      CLR R0
2378 005710 005000      CLR R1
2379 005712 004737 004200 JSR PC,T7EMPTY ;EMPTY BUFFER AND CHECK CONTENTS
2380 005716 000207      RTS PC

```

2381
2382
2383
2384
2385
2386
2387
2388
2389 005720 105737 012752
2390 005724 100022
2391 005726 005037 012352
2392 005732 004737 004054
2393 005736 005237 012352
2394 005742 004737 012306
2395 005746 004737 011014
2396 005752 052777 040001 173226
2397 005760 004737 006622
2398 005764 000775
2399 005766 004737 004200
2400 005772 000137 004306
2401
2402
2403
2404
2405
2406
2407
2408 005776 005037 012352
2409 006002 004737 004054
2410 006006 005237 012352
2411 006012 004737 012662
2412 006016 004737 012306
2413 006022 004737 011014
2414 006026 012737 000007 007730
2415 006034 004737 005636
2416 006040 000137 004306

.SBTTL TEST 22 - INITIALIZE IMPLIED READ
;AFTER PREVIOUSLY WRITING A PATTERN ON SECTOR 1 TRACK 1, THIS TEST
;CHANGES THE CONTENTS OF THE SECTOR BUFFER AND DOES A PROGRAMMED INITIALIZE.
;AFTER WHICH THE SECTOR BUFFER MUST AGAIN CONTAIN THE DATA PREVIOUSLY
;WRITTEN ON THAT SECTOR AND TRACK.
;NOTE: THIS TEST WILL ONLY WORK IF UNIT 0 IS SELECTED AND ON LINE.

T22: TSTB UNITSEL ;IF UNIT 0 IS NOT SELECTED SKIP THIS TEST
BPL 2\$
CLR PAT
JSR PC,T6FILL ;LOAD THE SECTOR BUFFER WITH 0
INC PAT ;RELOAD CORE BUFFER WITH 1'S
JSR PC,GETPATTERN
JSR PC,ADJSUM
BIS #RECAL,@RXCS ;SET THE INIT. BIT
1\$: JSR PC,SDN
BR 1\$
2\$: JSR PC,T7EMPTY ;EMPTY THE SECTOR BUFFER AND CHECK IT.
JMP DEXIT ;END OF TEST 22

.SBTTL TEST 23 - READ TEST

;THIS TEST VERIFIES THAT A READ FUNCTION DOES IN FACT LOAD THE SECTOR
;BUFFER WITH DATA READ FROM THE SELECTED ADDRESS.

T23: CLR PAT
JSR PC,T6FILL ;LOAD SECTOR BUFFER WITH 0'S
INC PAT
JSR PC,GETUNIT
JSR PC,GETPATTERN ;RELOAD CORE BUFFER WITH 1'S
JSR PC,ADJSUM ;SET UP FOR CHECK SUM
MOV #7,FUNCTION ;SET READ FUNCTION AND GO
JSR PC,T21X ;ISSUE COMMAND, WAIT FOR DONE, & TEST DATA
JMP DEXIT ;END OF TEST 23

.SBTTL TEST 24 - DATA TRANSFER AND VERIFICATION

;THE PURPOSE OF THIS TEST IS TO WRITE THEN READ AND VERIFY DATA
;ON ALL SECTORS OF THE SELECTED TRACKS. THE TEST ALTERNATES BETWEEN
;DRIVES ON THE SELECTED TRACKS. DATA PATTERN IS A FLOATING 0.

2417									
2418									
2419									
2420									
2421									
2422									
2423									
2424	006044	012737	000002	012352	T24:	MOV #2,PAT			;SET DATA PATTERN TO FLOATING 0
2425	006052	013702	001204		T24X:	MOV KRXVEC,R2			;SET INTERRUPT ADDRESSES
2426	006056	012722	011554			MOV #INTSERV,(R2)+			
2427	006062	012712	000340			MOV #PR7,(R2)			
2428	006066	004737	007040			JSR PC,DRVSWP			;GO TRANSFER THE DATA
2429	006072	000137	004306			JMP DEXIT			;END OF TEST 24 OR 25

.SBTTL TEST 25 - DATA TRANSFER AND VERIFICATION VIA DELETED DATA MODE

;THIS TEST TRANSFERES DATA JUST LIKE TEST 24 EXCEPT IT USES THE
;DELETED DATA FORMAT AND A DATA PATTERN OF FLOATING 1.

2430									
2431									
2432									
2433									
2434									
2435									
2436									
2437	006076	012737	000003	012352	T25:	MOV #3,PAT			;SET DATA PATTERN TO FLOATING 1
2438	006104	000137	006052			JMP T24X			;GO TRANSFER THE DATA

.SBTTL TEST 26 - HEAD "HOME" TEST

;THIS TEST MOVES THE HEAD OUT TO TRACK 12 (OCTAL) AND THEN WRITES/READ CHECKS
;ALL SECTORS (RANDOM DATA) ON EACH TRACK. THE TRACK SEQUENCE
;IS DECREMENTED BACK TO TRACK 0 (HOME). AFTER COMPLETING
;DRIVE 0 IT SWITCHES OVER TO DRIVE 1 DOING THE SAME TEST.

2440									
2441									
2442									
2443									
2444									
2445									
2446									
2447									
2448	006110	052737	000200	013164	T26:	BIS #BIT7,SEQUEN			;SPECIAL DECREMENT SEQUENCE
2449	006116	012737	000007	012352		MOV #7,PAT			;SELECT RANDOM DATA
2450	006124	013702	001204			MOV KRXVEC,R2			
2451	006130	012722	011554			MOV #INTSERV,(R2)+			
2452	006134	012712	000340			MOV #PR7,(R2)			
2453	006140	004737	007114			JSR PC,WTRDCK			
2454	006144	042737	000200	013164		BIC #BIT7,SEQUEN			
2455	006152	000137	004306			JMP DEXIT			;END OF TEST 26

```
2456
2457
2458
2459
2460 006156 104401 017007
2461
2462 006162 012646
2463
2464 006164 104403
2465 006166 006
2466 006167 000
2467 006170 104401 017122
2468 006174 012716 002504
2469
2470 006200 000002
2471
2472
2473
2474
2475
2476 006202 104401 017054
2477
2478
2479 006206 012646
2480
2481 006210 104403
2482 006212 006
2483 006213 000
2484 006214 104401 017122
2485 006220 012716 002504
2486
2487 006224 000002
2488
2489
2490
2491
2492 006226 013700 001214
2493
2494 006232 105701
2495
2496 006234 100401
2497 006236 005300
2498
2499 006240 006300
2500 006242 006300
2501 006244 006300
2502 006246 006300
2503 006250 006300
2504 006252 042700 000037
2505
2506 006256 000207
2507
```

```

;THE FOLLOWING SECTION OF CODE WILL ALLOW PROVIDING INFORMATION
;TO THE USER WHEN AN 'UNEXPECTED' BUS TIMEOUT TO LOCATION 4 OCCURS
BUSERR: TYPE, LOC4M
MOV (SP)+,-(SP)
TYPOS
.BYTE 6
.BYTE 0
TYPE, PCM
MOV #REBEGIN,(SP)
RTI

;TYPE MESSAGE INDICATING AN
;UNEXPECTED BUS TIMEOUT OCCURRED
;;SAVE (SP)+ FOR TYPEOUT
;;SETUP TO TYPE PC WHERE TIMEOUT OCCURRED
;;GO TYPE--OCTAL ASCII
;;TYPE 6 DIGITS
;;SUPPRESS LEADING ZEROS
;TYPE MESSAGE '=PC'
;SET RETURN 'PC' TO START THE
;PROGRAM OVER AGAIN
;RETURN TO BEGINNING OF PROGRAM

;THE FOLLOWING SECTION OF CODE WILL ALLOW PROVIDING INFORMATION
;TO THE USER WHEN AN 'UNEXPECTED' RESERVED INSTRUCTION TRAP TO LOCATION
;10 OCCURS
RESERR: TYPE, LOC10M
MOV (SP)+,-(SP)
TYPOS
.BYTE 6
.BYTE 0
TYPE, PCM
MOV #REBEGIN,(SP)
RTI

;TYPE MESSAGE INDICATING AN
;UNEXPECTED RESERVED INSTRUCTION
;TRAP OCCURRED
;;SAVE (SP)+ FOR TYPEOUT
;;SETUP TO TYPE PC WHERE RESERVED TRAP OCCURRED
;;GO TYPE--OCTAL ASCII
;;TYPE 6 DIGITS
;;SUPPRESS LEADING ZEROS
;TYPE MESSAGE '=PC'
;SET RETURN 'PC' TO START THE
;PROGRAM OVER AGAIN
;RETURN TO BEGINNING OF PROGRAM

;THIS ROUTINE WILL CALCULATE THE PRIORITY LEVEL FOR THE PROCESSOR
;BASED ON THE CURRENT PRIORITY LEVEL OF THE DEVICE (CONTENTS OF 'BRLEV:')
CPUPRI: MOV BRLEV,RO
TSTB R1
BMI 1$
DEC RO
1$: ASL RO
ASL RO
ASL RO
ASL RO
ASL RO
BIC #37,RO
RTS PC

;GET THE PROPOSED RX11 DEVICE
;INTERRUPT PRIORITY LEVEL VALUE
;IS CPU LEVEL TO BE THE SAME AS
;THE DEVICE LEVEL OR 1 LESS?
;BRANCH IF SAME AS!
;DROP DEVICE LEVEL PRIORITY
;BY 1 LEVEL FOR PSW
;FORM BITS <7-5> FOR PSW
;ENSURE THAT T,N,Z,V, & C BITS
;FOR THE PROCESSOR ARE CLEAR
;RETURN TO MAINLINE CODE
```

```
2508 .SBTTL " ERROR " TRAP SERVICE ROUTINE
2509
2510 :*****
2511 :*****
2512
2513 :
2514 " ERROR "
2515
2516 :*****
2517 :*****
2518 006260 011637 006522 XERROR: MOV @ SP, EPCSCOPE ; RETURN ADDRESS FROM " ERROR"
2519 006264 062737 000002 006522 ADD #2, EPCSCOPE ; NOW (EPCSCOPE) = SUBSCOPE+2, OR SCOPE+2
2520 006272 005237 006520 INCERRORS: INC ERRORS
2521 006276 001775 BEQ INCERRORS
2522
2523 : DATA SW 13 = 0 TO PRINT APPROPRIATE ERROR MESSAGE
2524
2525 006300 104406 CKSWR
2526 006302 032777 020000 172706 BIT #SW13,@SWR
2527 006310 001056 BNE NOPRINT
2528 006312 005037 002554 CLR CCOUNT
2529 006316 032737 000400 012752 BIT #BIT8,UNITSEL ; WAS PREVIOUS ERROR REPORTED ON THIS PASS
2530 006324 001002 BNE 1$
2531 006326 104401 015700 TYPE, MXEHEADER
2532
2533 006332 104401 016162 1$: TYPE, MCRLF
2534 006336 011604 MOV @ SP, R4 ; ERADR
2535 006340 162704 000002 SUB #2, R4
2536 006344 010446 MOV R4, -(SP)
2537 006346 104403 TYPOS
2538 006350 006 .BYTE 6
2539 006351 001 .BYTE 1
2540 006352 104401 016703 TYPE, SPACE
2541 006356 013746 002560 MOV FAST, -(SP) ; FAST (FIRST ADDRESS OF SELECTED TEST)
2542 006362 104404 TYPON
2543 006364 104401 016703 TYPE, SPACE
2544 006370 013746 006604 MOV PCSCOPE, -(SP) ; FAPT (FIRST ADDRESS OF PRESENT TEST)
2545 006374 104404 TYPON
2546 006376 104401 016703 TYPE, SPACE
2547 006402 010246 MOV R2, -(SP) ; BLANK
2548 006404 104404 TYPON
2549 006406 104401 016703 TYPE, SPACE
2550 006412 010046 MOV R0, -(SP) ; EXPECTED (GOOD) RESULT OF TEST
2551 006414 104404 TYPON
2552 006416 104401 016703 TYPE, SPACE
2553 006422 010146 MOV R1, -(SP) ; ACTUAL (BAD) RESULT OF TEST
2554 006424 104404 TYPON
2555 006426 104401 016703 TYPE, SPACE
2556 006432 013737 002556 006444 MOV PASS,2$
2557 006440 004537 015642 JSR R5,SGLDEC
2558 006444 000000 2$: OPEN
```

CZRXBFO RX11 INTERFACE TEST
CZRXBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 PAGE 59
" ERROR " TRAP SERVICE ROUTINE

SEQ 0087

```
2559 ; DATA SW 0 = 0 TO RING BELL AT ERROR
2560
2561 006446 052737 000400 012752 NOPRINT: BIS #BIT8,UNITSEL ;SET HARD ERROR FLAG
2562 006454 004737 006476 JSR PC,DING
2563
2564 ; DATA SW 15 = 1 TO HALT AT ERROR
2565
2566 006460 104406 1$: CKSWR
2567 006462 032777 100000 172526 BIT #SW15,@SWR
2568 006470 001401 BEQ 2$
2569 006472 000000 HALT
2570 006474 000002 2$: RTI
2571
2572 006476 104406 DING: CKSWR
2573 006500 032777 000001 172510 BIT #SW0,@SWR
2574 006506 001002 BNE 1$
2575 006510 104401 006516 TYPE ,MABELL
2576 006514 000207 1$: RTS PC
2577
2578 006516 000007 MABELL: .ASCIZ <07> ; DING - A - LING
2579 .EVEN
2580
2581 006520 000000 ERRORS: 0
2582 006522 000000 EPCSCOPE: 0
```



```
2583 .SBTTL " SCOPE " TRAP SERVICE ROUTINE
2584
2585 ; " SCOPE "
2586
2587 006524 005737 006520 XSCOPE: TST ERRORS
2588 006530 001015 BNE SCOPING
2589
2590 ; NO ERRORS HAVE BEEN DETECTED
2591
2592 ; JUST SET (PCSCOPE) = FIRST ADDRESS OF THE SCOPE LOOP
2593
2594 ; (IN CASE ERRORS ARE DETECTED LATER)
2595
2596 006532 005037 006520 NOSCOPE: CLR ERRORS
2597 006536 011637 006604 MOV @ SP, PCSCOPE
2598 006542 000002 RTI
2599
2600 ; " SUBSCOPE "
2601
2602 006544 005737 006520 XSUBSCOPE: TST ERRORS
2603 006550 001001 BNE 1$
2604 006552 000002 RTI ; NO ERRORS EXIST
2605
2606 ; ERRORS DO EXIST
2607
2608 ; IF THIS ERROR ADDRESS IS THE SAME ADDRESS WITHIN PROGRAM LOCATION " EPCSCOPE"
2609
2610 ; THEN THIS IS A SCOPING LOOP
2611
2612 ; IF NOT - THEN EXIT
2613
2614 006554 021637 006522 1$: CMP @ SP, EPCSCOPE
2615 006560 001401 BEQ SCOPING
2616 006562 000002 RTI
2617
2618 ; SW 11 = 1 TO LOCK ON SCOPING LOOP
2619
2620 ; THIS IS A SCOPING LOOP
2621
2622 006564 104406 SCOPING: CKSWR
2623 006566 032777 004000 172422 BIT #SW11,@SWR
2624 006574 001756 BEQ NOSCOPE ;DO NOT LOOP ON ERROR
2625 006576 013716 006604 MOV PCSCOPE, @ SP
2626 006602 000002 RTI ; LOCK FOR SCOPE LOOP
2627 006604 000000 PCSCOPE: 0
```

```

2628          ; WAIT FOR TRANSFER REQUEST FLAG TO SET
2629
2630 006606 105777 172374 STR:   TSTB @ RXCS
2631 006612 100002          BPL RTSPC
2632 006614 062716 000002          ADD #2, @ SP          ; ADJUST FOR EXIT
2633 006620 000207          RTSPC:  RTS PC
2634
2635          ; WAIT FOR THE DONE FLAG ; ) SET
2636
2637 006622 032777 000040 172356 SDN:   BIT #BIT5, @ RXCS          ; TEST THE DONE BIT # 5
2638 006630 001055          BNE XSDN
2639 006632 062737 000001 007004          ADD #1, HANGER
2640 006640 005537 007006          ADC HANGPL
2641 006644 001401          BEQ HUNGUP
2642 006646 000207          RTS PC
2643
2644          ; THE DEVICE TEST IS HUNG - DONE HAS NOT SET
2645
2646          ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
2647
2648 006650 104000          HUNGUP: ERROR

```

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

:
: IF THE DIAGNOSTIC HITS THIS ERROR REPORT AND THE 'HUNG @ PC'
: LOCATION IS WITHIN ONE OF THE TESTS TABULATED BELOW THEN THE
: POSSIBLE CHIPS ARE AS FOLLOWS:
:

```

```

: HUNG @ TEST          POSSIBLE CHIPS
:
: TEST 5 ( PART 1 )   E7,E34,E4,E22,E15,E1,E11,E37
: TEST 5 ( PART 2 )   E7,E34,E2,E14,E6,E1,E37,E8
: TEST 6              E4,E15,E1,E19,E18,E22

```

```

; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
; /*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

```

```

2668
2669 006652 005037 007004          CLR HANGER          ; RESET THE HANG COUNTERS
2670 006656 012737 177740 007006          MOV #177740, HANGPL
2671 006664 104406          CKSWR
2672 006666 032777 020000 172322          BIT #SW13, @SWR
2673 006674 001402          BEQ 1$
2674 006676 000137 002200          JMP MORETESTS
2675 006702 104401 006722          1$:   TYPE, MHUNGPC
2676
2677          ; THE PC IS ALREADY ON THE STACK
2678
2679 006706 162716 000002          SUB #2, @ SP
2680 006712 104403          TYPOS
2681 006714          006          .BYTE 6
2682 006715          001          .BYTE 1
2683 006716 000137 002200          JMP MORETESTS

```

CZRXBFO RX11 INTERFACE TEST
CZRXBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 M 7 PAGE 62
" SCOPE " TRAP SERVICE ROUTINE

SEQ 0090

2684					
2685	006722	005015	042504	044526	MHUNGPC: .ASCIZ <15><12>'DEVICE TEST HUNG @ PC ''
2686	006730	042503	052040	051505	
2687	006736	020124	052510	043516	
2688	006744	040040	050040	020103	
2689	006752	000			
2690	006753	040	040520	051523	MPASS: .ASCIZ '' PASS =''
2691	006760	036440	000		
2692		006764			.EVEN
2693					
2694	006764	005037	007004		XSDN: CLR HANGER
2695	006770	012737	177740	007006	MOV #177740,HANGPL
2696	006776	062716	000002		ADD #2, @ SP
2697	007002	000207			RTS PC
2698	007004	000000			HANGER: 0
2699	007006	177740			HANGPL: 177740

; UPDATE FOR EXIT

2700
2701
2702
2703
2704
2705 007010 004737 013000
2706 007014 004737 012662
2707 007020 004737 013132
2708 007024 004737 010042
2709 007030 005337 013152
2710 007034 001371
2711 007036 000207
2712
2713
2714
2715
2716
2717
2718 007040 004737 012306
2719 007044 004737 013000
2720 007050 004737 012662
2721 007054 004737 013132
2722 007060 004737 007162
2723 007064 004737 010662
2724 007070 004737 012662
2725 007074 004737 007162
2726 007100 004737 010662
2727 007104 005337 013152
2728 007110 001357
2729 007112 000207
2730
2731
2732
2733
2734
2735
2736 007114 004737 012306
2737 007120 004737 013000
2738 007124 004737 012662
2739 007130 004737 013132
2740 007134 004737 007162
2741 007140 004737 010662
2742 007144 005337 013152
2743 007150 001367
2744 007152 004737 012754
2745 007156 000207
2746 007160 000757

.SBTTL DRIVE TEST SELECTION
;DO A READ ONLY FUNCTION ON ALL SECTORS.
;THIS DOES NOT VERIFY THE DATA, ONLY TESTS FOR CRC ERRORS.
RDONLY: JSR PC,INITTRACKS
JSR PC,GETUNIT
1\$: JSR PC,GETTRACK
JSR PC,READ
DEC TRKCNT
BNE 1\$
RTS PC
;*****
;WRITE AND READ DATA ON SPECIFIED TRACK AND ALTERNATE
;DRIVES BEFORE GOING TO THE NEXT TRACK.
DRVSWP: JSR PC,GETPATTERN
JSR PC,INITTRACKS
1\$: JSR PC,GETUNIT
JSR PC,GETTRACK
JSR PC,WRITE
JSR PC,READCHK
JSR PC,GETUNIT
JSR PC,WRITE
JSR PC,READCHK
DEC TRKCNT
BNE 1\$
RTS PC
;*****
;WRITE ALL SECTORS AND READ/VERIFY ALL SECTORS
WTRDCK: JSR PC,GETPATTERN
XWTRDCK: JSR PC,INITTRACKS
JSR PC,GETUNIT
1\$: JSR PC,GETTRACK
JSR PC,WRITE
JSR PC,READCHK
DEC TRKCNT
BNE 1\$
JSR PC,DONE ;HAVE BOTH DRIVES BEEN TESTED
RTS PC ;YES
BR XWTRDCK ;NO, GO TO OTHER UNIT

```

2747          .SBTTL WRITE FUNCTION
2748
2749 007162 004737 013352 WRITE: JSR PC,INITSECTOR ;SET UP FIRST, LAST, AND SECTOR COUNTER
2750 007166 004737 013450 XWRITE: JSR PC,GETSECTOR ;PICK UP NEXT SECTOR
2751 007172 004737 011014 FILLBUF: JSR PC,ADJSUM ;ADJUST DATA BUFFER AND CHECK SUM FOR ADDRESSES
2752 007176 012746 007360 MOV #FILLDONE,-(SP) ;PUT GOOD RETURN ON STACK
2753 007202 012746 007250 MOV #FILLER,-(SP) ;PUT ERROR RETURN ON STACK
2754 007206 005037 011500 CLR BYTECNTR
2755 007212 005046          CLR          ;LOWER 'CPU' LEVEL
2756 007214 012746 007222 MOV          ;SET RETURN 'PC'
2757 007220 000002          RTI          ;GET 'CPU' LEVEL INTO 'PSW'
2758 007222 012777 000101 171756 1$: MOV #FBIE,@RXCS ;EXECUTE FILLBUFFER COMMAND
2759 007230 105777 171752 FILLFLAG: TSTB @RXCS ;TEST FOR TRANSFER REQUEST FLAG
2760 007234 100375          BPL FILLFLAG
2761 007236 112077 171746 XFRBYTE: MOVB (RO)+,@RXDB ;TRANSFER DATA BYTE
2762 007242 005237 011500 INC BYTECNTR
2763 007246 000770          BR FILLFLAG ;WAIT FOR NEXT TR FLAG
2764
2765 007250 005726          FILLER: TST (SP)+ ;REMOVE THE DONE RETURN FROM THE STACK
2766 007252 012737 016430 007314 MOV #MFIL,PTYP1+2 ;PUT ADDR OF FILLBUF MESSAGE IN PAR ERR TYP0UT 1
2767 007260 012737 007166 007356 MOV #XWRITE,PCONT+2 ;IF NO LOOP ON ERROR GO TO NEXT SECTOR
2768 007266 012737 007172 007352 MOV #FILLBUF,PLOOP+2 ;IF LOOP ON ERROR RETURN THROUGH PLOOP
2769 007274 000137 007300 JMP PARTEST ;PRINT OUT PAR ERR AND TEST CONDITIONS FOR RETRY
2770
2771 007300 104406          PARTEST: CKSWR
2772 007302 032777 020000 171706 BIT #SW13,@SWR ;TEST DON'T PRINT ERROR SWITCH
2773 007310 001006          BNE CONT4
2774 007312 104401 000000 PTP1: TYPE ,OPEN ;PRINT THE PARITY ERROR MESSAGE
2775 007316 104401 016641 TYPE ,MPAR
2776 007322 104401 016162 TYPE ,MCRLF
2777 007326 104406          CONT4: CKSWR
2778 007330 005777 171662 TST @SWR ;TEST HALT ON ERROR SWITCH
2779 007334 100001          BPL CONT13
2780 007336 000000          HLT6: HALT ;HALT ON ERROR
2781 007340 032777 004000 171650 CONT13: BIT #SW11,@SWR ;TEST LOOP ON ERROR SWITCH
2782 007346 001402          BEQ PCONT ;IF NOT SET GO TO NEXT SECTOR
2783 007350 000137 007172 PLOOP: JMP FILLBUF ;RETURN TO LOOP ON TEST THROUGH HERE
2784 007354 000137 010160 PCONT: JMP NEXTRD ;GO TO NEXT SECTOR THROUGH HERE
2785
2786 007360 005037 007004 FILLDONE: CLR HANGER
2787 007364 012746 007456 MOV #WRTDONE,-(SP) ;SET GOOD RETURN ON STACK
2788 007370 012746 007472 MOV #WRTER,-(SP) ;SET ERROR RETURN ON STACK
2789 007374 112737 000105 007730 MOVB #WRTIE,FUNCTION ;SET FUNCTION WORD TO WRITE
2790 007402 022737 006076 006604 CMP #T25,PCSCOPE ;IS THIS THE DELETED DATA TEST
2791 007410 001003          BNE 1$
2792 007412 112737 000115 007730 MOVB #WTDDIE,FUNCTION
2793 007420 004737 007662 1$: JSR PC,COMMWORD ;TRANSFER COMMAND TO DRIVE
2794 007424 005046          CLR          ;LOWER 'CPU' LEVEL
2795 007426 012746 007434 MOV          ;SET RETURN 'PC'
2796 007432 000002          RTI          ;GET 'CPU' LEVEL INTO 'PSW'
2797 007434 032777 000040 171544 2$: BIT #DONEBIT,@RXCS ;WAIT FOR DONE
2798 007442 001774          BEQ 2$
2799 007444 005237 007004 3$: INC HANGER ;WAIT FOR INTERRUPT
2800 007450 001375          BNE 3$
2801 007452 000137 011510 JMP NOINTER ;NO INTERRUPT ERROR

```

```
2802 007456 005337 013442      WRTDONE:      DEC SECCNTR      ;TEST SECTOR COUNTER
2803 007462 001001              BNE 2$          ;NOT LAST SECTOR GO TO NEXT ONE
2804 007464 000207              RTS PC
2805 007466 000137 007166      2$:           JMP XWRITE
2806
2807 007472 005726              WRTER:        TST (SP)+       ;REMOVE THE DONE RETURN FROM THE STACK
2808 007474 032737 000002 012174 BIT #BIT1,ASTAT ;IS THIS A PARITY ERROR
2809 007502 001413              BEQ WRTSEK     ;NO, IT MUST BE A SEEK ERROR
2810              ;PARITY ERROR DURING A WRITE FUNCTION
2811 007504 012737 016632 007314 MOV #MWRITE,PTYP1+2 ;PUT ADDR OF WRITE MESSAGE IN PAR ER TYPOUT 1
2812 007512 012737 007456 007356 MOV #WRTDONE,PCONT+2 ;IF NO LOOP GO TO NEXT SECTOR
2813 007520 012737 007360 007352 MOV #FILLDONE,PLOOP+2 ;IF LOOP RETURN THROUGH PLOOP TO REWRITE
2814 007526 000137 007300              JMP PARTEST    ;GO INC LOG AND TEST FOR RETRY
2815
2816              ;SEEK ERROR DURING A WRITE FUNCTION
2817 007532 012737 007172 007632 WRTSEK:      MOV #FILLBUF,SEKRTY+2 ;SETUP FOR WRT RETRY ON SEEK ERROR
2818              ;(AFTER A RECAL. THE CONTENTS OF SECTOR 1,
2819              ;TRACK 1 ARE LOADED INTO THE SECTOR BUFFER.
2820              ;TO REWRITE THE CORRECT DATA THE PROGRAM
2821              ;MUST REFILL THE SECTOR BUFFER.
2822 007540 012737 016632 007570 MOV #MWRITE,STYP1+2 ;PUT ADDR OF WRITE MESSAGE IN SEEK ER TYPEOUT 1
2823 007546 004737 007554              JSR PC,SEEKER  ;RECORD SEEK ERROR
2824 007552 000741              BR WRTDONE     ;GO TO NEXT SECTOR CAN'T FIND THIS ONE
2825
2826 007554 104406              SEEKER:      CKSWR
2827 007556 032777 020000 171432 BIT #SW13,@SWR  ;CHECK DON'T PRINT ERROR SWITCH
2828 007564 001004              BNE SWHLT1
2829 007566 104401 016632      STYP1:      TYPE ,MWRITE   ;PRINT WRITE (READ) SEEK ERROR
2830 007572 004737 007634              JSR PC,SEKTYP
2831 007576 104406              SWHLT1:     CKSWR
2832 007600 005777 171412              TST @SWR      ;TEST THE HALT ON ERROR SWITCH
2833 007604 100001              BPL CONT14
2834 007606 000000              HLT7:       HALT        ;HALT ON THE ERROR
2835 007610 004737 007734      CONT14:    JSR PC,HOME   ;RECALIBRATE ON SEEK ERRORS
2836 007614 104406              CKSWR
2837 007616 032777 004000 171372 BIT #SW11,@SWR ;CHECK THE LOOP ON ERROR SWITCH
2838 007624 001001              BNE SEKRTY   ;IF SET LOOP ON THE ERROR THROUGH SEEK RETRY.
2839 007626 000207              RTS PC
2840 007630 000137 007172      SEKRTY:    JMP FILLBUF   ;RETRY WRITE COMMAND (READ COMAND)
2841
2842 007634 104401 016617      SEKTYP:    TYPE ,MSEEK   ;TYPE SEEK ERROR
2843 007640 104401 016074              TYPE ,MPRES   ;TYPE ADDRESS OF TRACK MOVED FROM
2844 007644 013746 013156              PRESTRK,-(SP) ;:SAVE PRESTRK FOR TYPEOUT
2845 007650 104403              MOV          ;:GO TYPE--OCTAL ASCII
2846 007652 003              TYPOS      ;:TYPE 3 DIGIT(S)
2847 007653 000              .BYTE     3
2848 007654 104401 016162              .BYTE     0
2849 007660 000207              .BYTE     0
2850              TYPE ,MCRLF ;:SUPPRESS LEADING ZEROS
2850              RTS PC
```

2851	007662	153737	012752	007730	COMMWORD:	BISB UNITSEL,FUNCTION	;SET UNIT SELECTION BIT IN COMMAND WORD
2852	007670	013777	007730	171310		MOV FUNCTION,@RXCS	;SEND OUT COMMAND TO DRIVE
2853	007676	004737	006606		1\$:	JSR PC,STR	;WAIT FOR TR FLAG
2854	007702	000775				BR 1\$	
2855	007704	113777	013444	171276		MOVB TSECTOR,@RXDB	;SEND OUT TARGET SECTOR
2856	007712	004737	006606		2\$:	JSR PC,STR	;WAIT FOR TR FLAG
2857	007716	000775				BR 2\$	
2858	007720	113777	013154	171262		MOVB TARGET,@RXDB	;SEND OUT TARGET TRACK
2859	007726	000207				RTS PC	
2860							
2861	007730	000000			FUNCTION:	0	
2862	007732	000000			DATAACK:	0	;DATA CHECK ON CRC ERROR FLAG
2863							
2864	007734	104406			HOME:	CKSWR	
2865	007736	032777	000400	171252		BIT #SW8,@SWR	;TEST NO RECAL SWITCH
2866	007744	001035				BNE RTN	
2867	007746	012777	040001	171232		MOV #RECAL,@RXCS	;ISSUE RECAL FUNCTION
2868	007754	004737	006622		2\$:	JSR PC,SDN	
2869	007760	000775				BR 2\$	
2870	007762	005777	171220			TST @RXCS	;WAS THERE AN ERROR
2871	007766	100021				BPL XHOME	;NO
2872	007770	104406				CKSWR	
2873	007772	032777	020000	171216		BIT #BIT13,@SWR	;YES, SHOULD IT BE PRINTED
2874	010000	001002				BNE 1\$;NO
2875	010002	004737	012200			JSR PC,RDCODE	
2876	010006	104406			1\$:	CKSWR	
2877	010010	005777	171202			TST @SWR	;TEST HALT ON ERROR SWITCH
2878	010014	100001				BPL 3\$	
2879	010016	000000				HALT	
2880	010020	032777	004000	171170	3\$:	BIT #SW11,@SWR	;TEST LOOP ON ERROR SWITCH
2881	010026	001342				BNE HOME	
2882	010030	000207				RTS PC	
2883	010032	012737	000001	013156	XHOME:	MOV #1,PRESTRK	;SET THE PRESENT TRACK TO TRACK 1
2884	010040	000207			RTN:	RTS PC	
2885							

```
.SBTTL READ DATA FROM THE DISKETTE

2886
2887
2888
2889 010042 004737 013352 READ: JSR PC,INITSECTOR
2890 010046 004737 013450 XREAD: JSR PC,GETSECTOR
2891 010052 005037 007732 REREAD: CLR DATAACK ;CLEAR CRC DATA CHECK FLAG
2892 010056 005037 007004 CLR HANGER
2893 010062 012746 010136 MOV #RDDONE,-(SP) ;SET GOOD RETURN ON STACK
2894 010066 012746 010170 MOV #RDERR,-(SP) ;SET READ ERROR RETURN ON STACK
2895 010072 112737 000107 007730 MOVB #RDIE,FUNCTION
2896 010100 004737 007662 JSR PC,COMMWORD
2897 010104 005046 CLR MOV ;LOWER 'CPU' LEVEL
2898 010106 012746 010114 MOV #18,-(SP) ;SET RETURN 'PC'
2899 010112 000002 RTI ;GET 'CPU' LEVEL INTO 'PSW'
2900 010114 032777 000040 171064 1$: BIT #DONEBIT,DRXCS ;WAIT FOR DONE BIT
2901 010122 001774 BEQ 1$
2902 010124 005237 007004 2$: INC HANGER ;WAIT FOR INTERRUPT
2903 010130 001375 BNE 2$
2904 010132 000137 011510 JMP NOINTER ;NO INTERRUPT ON DONE
2905
2906 010136 022737 005510 006604 RDDONE: CMP #T20,PCSCOPE ;IS THIS THE READ ONLY TEST (T20)
2907 010144 001405 BEQ NEXTRD ;YES,DON'T CHECK FOR DELETED DATA
2908 010146 004737 010436 JSR PC,DDCHK ;CHECK FOR DELETED DATA INDICATOR
2909 010152 005701 TST R1 ;BIT 15 OF R1 IS READ 1 SECTOR FLAG
2910 010154 100001 BPL NEXTRD
2911 010156 000207 RTS PC ;IF SET,GO VERIFY DATA JUST READ
2912 010160 005337 013442 NEXTRD: DEC SECCNTR
2913 010164 001330 BNE XREAD
2914 010166 000207 RTS PC ;READ FUNCTION IS DONE
2915
2916 010170 005726 RDERR: TST (SP)+ ;REMOVE THE DONE RETURN FROM THE STACK
2917 010172 032737 000002 012174 BIT #BIT11,ASTAT ;IS THIS A PARITY ERROR
2918 010200 001413 BEQ 1$ ;NO, SEE IF ITS A CRC ERROR
2919 ;PARITY ERROR DURING A READ FUNCTION
2920 010202 012737 016572 007314 MOV #MREAD,PTYP1+2 ;PUT ADDR OF READ MESSAGE IN PAR ERR TYPEOUT 1
2921 010210 012737 010052 007352 MOV #REREAD,PLOOP+2 ;IF LOOP ON ERROR LOOP THROUGH PLOOP
2922 010216 012737 010160 007356 MOV #NEXTRD,PCONT+2 ;IF NO LOOP GO TO NEXT READ
2923 010224 000137 007300 JMP PARTEST ;RECORD PARITY ERROR AND RETRY FUNCTION
2924 010230 032737 000001 012174 1$: BIT #BIT0,ASTAT ;IS THIS A CRC ERROR
2925 010236 001011 BNE CRCER ;YES GO TEST AND LOG IT
2926 ;SEEK ERROR DURING A READ FUNCTION
2927 010240 012737 010052 007632 MOV #REREAD,SEKRTY+2 ;SET SEEK CONTINUE FOR READ RETRY
2928 010246 012737 016572 007570 MOV #MREAD,STYP1+2 ;SET ADDR OF READ MESSAGE IN SEEK ER TYPEOUT 1
2929 010254 004737 007554 JSR PC,SEEKER ;RECORD SEEK ERROR
2930 010260 000737 BR NEXTRD ;GO TO NEXT SECTOR,CAN'T READ THIS ONE
```



```
2931                                     :CRC ERROR DETECTED WHILE READING
2932
2933 010262 005701          CRCER:          TST R1          ;IF READ ONLY. REPORT DATA CRC ERROR
2934 010264 100034          BPL DATAAC      ;
2935 010266 005237 007732  INC DATAK      ;SET DATA CHECK FLAG
2936 010272 004737 010672  JSR PC,EMPBUFF   ;CHECK FOR A DATA ERROR
2937 010276 005737 011506  TST ERCNTR      ;WAS THERE A DATA ERROR
2938 010302 001025          BNE DATAAC      ;YES, REPORT IT
2939 010304 104406          CKSWR
2940 010306 032777 020000 170702  BIT #SW13,@SWR  ;TEST DON'T PRINT SWITCH
2941 010314 001004          BNE 2$
2942 010316 104401 016542  TYPE ,MBADCRC   ;TYPE CRC GENERATOR ERROR
2943 010322 104401 016162  TYPE ,MCRLF
2944 010326 104406          CKSWR
2945 010330 005777 170662  TST @SWR        ;TEST HALT ON ERROR SWITCH
2946 010334 100001          BPL CONT15
2947 010336 000000          HLT10:          HALT          ;HALT ON ERROR
2948 010340 032777 004000 170650  CONT15:        BIT #SW11,@SWR  ;CHECK LOOP ON ERROR SWITCH
2949 010346 001001          BNE 3$
2950 010350 000703          BR NEXTRD      ;DON'T LOOP GO TO NEXT SECTOR
2951 010352 000137 010052  JMP REREAD      ;LOOP ON TEST.
2952
2953                                     :DATA CRC ERROR
2954
2955 010356 104406          DATAAC:         CKSWR
2956 010360 032777 020000 170630  BIT #SW13,@SWR  ;TEST DON'T PRINT ERROR SWITCH
2957 010366 001004          BNE 4$
2958 010370 104401 016600  TYPE ,MCRC      ;TYPE DATA CRC ERROR
2959 010374 104401 016162  TYPE ,MCRLF
2960 010400 104406          CKSWR
2961 010402 005777 170610  TST @SWR        ;TEST HALT ON ERROR SWITCH
2962 010406 100001          BPL CONT16
2963 010410 000000          HLT12:          HALT          ;HALT ON ERROR
2964 010412 032777 004000 170576  CONT16:        BIT #SW11,@SWR  ;TEST LOOP ON ERROR
2965 010420 001004          BNE 5$          ;IF SET LOOP ON THE TEST
2966 010422 062706 000002  ADD #2,SP        ;REMOVE READ DONE ADDRESS FROM STACK
2967 010426 000137 010160  JMP NEXTRD      ;READ NEXT SECTOR CAN'T READ THIS ONE
2968 010432 000137 010052  JMP REREAD      ;NO,GO REREAD THIS SECTOR
2969
2970
```

```
2971 010436 022737 006076 006604 DDCHK:      CMP #T25,PCSCOPE      ;IS THIS TEST 25
2972 010444 001041          BNE CONT10
2973 010446 132737 000100 012174      BITB #BIT6,ASTAT      ;THIS IS TEST 25
2974 010454 001056          BNE RETURN           ;DD BIT SHOULD BE SET
2975 010456 104406          CKSWR
2976 010460 032777 020000 170530      BIT #SW13,@SWR       ;TEST DON'T PRINT ERROR SWITCH
2977 010466 001013          BNE CONT11
2978 010470 004737 010614      JSR PC,ERMSG
2979 010474 104401 016011      TYPE ,MDDMIS         ;TYPE MISSING DELETED DATA BIT
2980 010500 052737 000400 012752 DDERR:      BIS #BIT8,UNITSEL    ;SET HARD ERROR FLAG
2981 010506 004737 012076      JSR PC,TYPADR        ;TYPE ADDRESS OF ERROR
2982 010512 104401 016162          TYPE ,MCRLF
2983 010516 104406          CKSWR
2984 010520 005777 170472          TST @SWR              ;TEST HALT ON ERROR SWITCH
2985 010524 100001          BPL CONT17
2986 010526 000000          HALT                  ;HALT ON DELETED DATA ERROR
2987 010530 032777 004000 170460 CONT17:      BIT #SW11,@SWR       ;TEST LOOP ON ERROR
2988 010536 001402          BEQ 4$
2989 010540 000137 010052          JMP REREAD            ;LOOP ON TEST
2990 010544 000137 010160          JMP NEXTRD           ;READ NEXT SECTOR
2991 010550 032737 000100 012174 CONT10:      BIT #BIT6,ASTAT      ;THIS IS NOT A DELETED DATA TRANSFER
2992 010556 001415          BEQ RETURN
2993 010560 052737 000400 012752      BIS #BIT8,UNITSEL    ;SET HARD ERROR FLAG
2994 010566 104406          CKSWR
2995 010570 032777 020000 170420      BIT #SW13,@SWR       ;TEST DON'T PRINT ERROR SWITCH
2996 010576 001347          BNE CONT11
2997 010600 004737 010614      JSR PC,ERMSG
2998 010604 104401 015763      TYPE ,MUNXDD         ;TYPE UNEXPECTED DELETED DATA BIT
2999 010610 000733          BR DDERR
3000 010612 000207          RETURN:
3001
3002
3003 010614 104401 016165      ERMSG:      TYPE ,MERHEADER
3004 010620 013746 006604          MOV PCSCOPE,-(SP)    ;;SAVE PCSCOPE FOR TYPEOUT
3005 010624 104403          TYPOS              ;;GO TYPE--OCTAL ASCII
3006 010626 006          .BYTE 6            ;;TYPE 6 DIGITS
3007 010627 000          .BYTE 0            ;;SUPPRESS LEADING ZEROS
3008 010630 104401 006753          TYPE ,MPASS
3009 010634 013737 002556 010646      MOV PASS,1$
3010 010642 004537 015642          JSR R5,SGLDEC
3011 010646 000000          1$:      OPEN
3012 010650 104401 016162          TYPE ,MCRLF
3013 010654 004737 006476          JSR PC,DING
3014 010660 000207          RTS PC
3015
3016
```

```

3017                                     .SBTTL READ AND VERIFY DATA
3018
3019                                     ;READ A SECTOR,EMPTY THE SECTOR BUFFER AND VERIFY
3020                                     ;THE DATA READ AGAINST CORE DATA BUFFER
3021
3022 010662 052701 100000 READCHK:      BIS #BIT15,R1      ;SET READ ONE SECTOR FLAG
3023 010666 004737 010042      JSR PC,READ      ;GO READ ONE SECTOR
3024 010672 005737 013442 EMPBUFF:    TST SECCNTR      ;IF CLEARED NO SECTORS WERE FOUND
3025 010676 001002      BNE 1$
3026 010700 000137 011474      JMP EXIT          ;GO TO NEXT TRACK
3027 010704 005037 011500 1$:          CLR BYTECNTR     ;CLEAR THE BYTE AND ERROR COUNTERS
3028 010710 005037 011506      CLR ERCNTR
3029 010714 052701 000200      BIS #BIT7,R1     ;R1 BIT 7 IS USED AS FIRST ERROR FLAG
3030 010720 004737 011014      JSR PC,ADJSUM    ;ADJUST DATA AND CK SUM FOR ADDRESSES
3031 010724 005037 011104      CLR CKSUM        ;SET UP FOR CHECK SUM ACCUMULATION
3032 010730 012746 011334      MOV #EMPDONE,-(SP) ;SET UP RETURN ADDRESSES
3033 010734 012746 011106      MOV #EMPER,-(SP)
3034 010740 005046      CLR             ;LOWER 'CPU' LEVEL
3035 010742 012746 010750      MOV             ;SET RETURN 'PC'
3036 010746 000002      RTI            ;GET 'CPU' LEVEL INTO 'PSW'
3037 010750 012777 000103 170230 2$:          MOV #EBIE,@RXCS ;LOAD EMPTY BUFFER FUNCTION
3038 010756 105777 170224 EMPFLAG:   TSTB @RXCS      ;TEST FOR TR FLAG
3039 010762 100375      BPL EMPFLAG
3040
3041 010764 117737 170220 011502 CKBYTE:    MOVB @RXDB,BADBYTE ;SAVE BYTE FROM DISKETTE
3042 010772 063737 011502 011104      ADD BADBYTE,CKSUM ;ACCUMULATE CHECK SUM
3043 011000 123720 011502      CMPB BADBYTE,(RO)+ ;COMPARE AGAINST GOOD BYTE
3044 011004 001054      BNE DATAER     ;IF NOT EQUAL GO TO DATAER
3045 011006 005237 011500      INC BYTECNTR
3046 011012 000761      BR EMPFLAG      ;GET NEXT BYTE
3047
3048 011014 113737 013154 017456 ADJSUM:    MOVB TARGET,BUFADR ;SET FIRST AND SECOND BYTES WITH ADDRESSES
3049 011022 113737 013444 017457      MOVB TSECTOR,BUFADR+1
3050 011030 013737 012564 011104      MOV SUM,CKSUM    ;GET THE PATTERN SUM
3051 011036 063737 013154 011104      ADD TARGET,CKSUM ;ADD TRACK ADDRESS TO CHECK SUM
3052 011044 063737 013444 011104      ADD TSECTOR,CKSUM ;ADD SECTOR ADDRESS TO CHECK SUM
3053 011052 113737 011104 017654      MOVB CKSUM,BUFADR+176 ;INSERT CHECK SUM TO DATA BUFFER
3054 011060 106337 011104      ASLB CKSUM       ;GENERATE NEGITIVE CHECK SUM
3055 011064 105437 011104      NEGB CKSUM
3056 011070 113737 011104 017655      MOVB CKSUM,BUFADR+177 ;INSERT NEG,SUM INTO DATA BUFFER
3057 011076 012700 017456      MOV #BUFADR,RO  ;SET ADDRESS OF BYTE IN RO
3058 011102 000207      RTS PC          ;RETURN
3059
3060 011104 000000      CKSUM:         0
3061
3062 011106 005726      EMPER:         TST (SP)+        ;REMOVE THE DONE RETURN FROM THE STACK
3063 011110 012737 016444 007314      MOV #EMPTY,PTYP1+2 ;PUT ADDR OF EMPTYBUF MESSAGE IN PAR ER TYP0UT
3064 011116 012737 010672 007352      MOV #EMPBUFF,PLOOP+2 ;RETURN THROUGH HERE TO LOOP ON ERROR
3065 011124 012737 011456 007356      MOV #NXREAD,PCONT+2 ;IF NO LOOP ON ERROR GO TO NEXT SECTOR
3066 011132 000137 007300      JMP PARTEST     ;REPORT PARITY ERRUR
3067

```

3068	011136	052737	000400	012752	DATAER:	BIS #BIT8,UNITSEL	;SET THE HAD ERROR FLAG
3069	011144	005237	011506			INC ERCNTR	;INC THE BYTE ERROR COUNTER
3070	011150	104406				CKSWR	
3071	011152	032777	020000	170036		BIT #SW13,@SWR	;TEST PRINT ERROR SW IN SWR
3072	011160	001054				BNE NOERTYP	;DON'T PRINT THE ERROR
3073	011162	032777	001000	170026		BIT #SW9,@SWR	;TEST PRINT 10 ERRORS SWITCH
3074	011170	001004				BNE 1\$;IF SET PRINT ALL ERRORS
3075	011172	023727	011506	000012		CMP ERCNTR,#10.	;HAVE 10 ERRORS BEEN TYPED
3076	011200	003044				BGT NOERTYP	;YES,DON'T PRINT ANY MORE
3077	011202	105701			1\$.	TSTB R1	;TEST FIRST ERROR FLAG
3078	011204	100014				BPL TYPERR	
3079	011206	004737	010614			JSR PC,ERMSG	;PRINT ADDRESS OF TEST
3080	011212	104401	016034			TYPE ,MDERHDR	;FIRST ERROR, PRINT ERROR HEADER
3081	011216	104401	016162			TYPE ,MCRLF	
3082	011222	004737	012076			JSR PC,TYPADR	;PRINT TRACK AND SECTOR LOCATIONS
3083	011226	104401	016123			TYPE ,MCOLMUN	;SET UP COLMUN HEADINGS
3084	011232	042701	000200			BIC #BIT7,R1	;CLEAR FIRST ERROR FLAG
3085	011236	013737	011500	011250	TYPERR:	MOV BYTECNTR,1\$;PRINT BYTE NUMBER
3086	011244	004537	015642			JSR R5,SGLDEC	
3087	011250	000000			1\$:	OPEN	
3088	011252	104401	016157			TYPE ,DBLSP	
3089	011256	013746	011502			MOV BADBYTE,-(SP)	;PRINT BYTE READ FROM DISKETTE
3090	011262	104403				TYPOS	
3091	011264	000003				.WORD 3	
3092	011266	104401	016157			TYPE ,DBLSP	
3093	011272	114037	011504			MOVB -(R0),GOODBYTE	;GET GOOD BYTE
3094	011276	005200				INC R0	;RETURN R0 TO NEXT BYTE IN BUFFER
3095	011300	013746	011504			MOV GOODBYTE,-(SP)	
3096	011304	104404				TYPON	;PRINT GOOD DATA
3097	011306	104401	016162			TYPE ,MCRLF	
3098	011312	104406			NOERTYP:	CKSWR	
3099	011314	005777	167676			TST @SWR	;TEST HALT ON ERROR SWITCH
3100	011320	100001				BPL CONT20	
3101	011322	000000			HLT14:	HALT	
3102	011324	005237	011500		CONT20:	INC BYTECNTR	
3103	011330	000137	010756			JMP EMPFLAG	
3104							

```
3105 011334 005737 007732 EMPDONE: TST DATAK ;WAS THIS READ CHECK CAUSED BY A CRC ERROR
3106 011340 00140i BEQ 1$ ;NO
3107 011342 000207 RTS PC ;YES, RETURN TO CRC HANDLER
3108 011344 005737 011506 1$: TST ERCNTR ;WAS THERE ERRORS
3109 011350 001442 BEQ NXREAD ;NO ERRORS
3110 011352 104406 CKSWR
3111 011354 032777 020000 167634 BIT #SW13,@SWR ;YES, TEST DON'T PRINT SWITCH
3112 011362 001024 BNE 2$ ;DON'T PRINT THE ERROR
3113 011364 104401 016375 TYPE ,MERC1 ;PRINT THE TOTAL DATA ERROR COUNT
3114 011370 013737 011506 011402 MOV ERCNTR,3$
3115 011376 004537 015642 JSR R5,SGLDEC
3116 011402 000000 3$: OPEN
3117 011404 104401 016712 TYPE ,MSUM ;INDICATE IF CHECK SUM WAS GOOD OR HAD ERRORS
3118 011410 105737 011104 TSTB CKSUM
3119 011414 001403 BEQ 4$
3120 011416 104401 016677 TYPE ,MBAD
3121 011422 000402 BR 5$
3122 011424 104401 016705 4$: TYPE ,MGOOD
3123 011430 104401 016162 5$: TYPE ,MCRLF
3124 011434 104406 2$: CKSWR
3125 011436 032777 004000 167552 BIT #SW11,@SWR ;TEST LOOP ON ERROR SWITCH
3126 011444 001404 BEQ NXREAD ;IF NOT SET GO TO NEXT SECTOR
3127 011446 004737 010052 JSR PC,RERead ;YES, GO REREAD THE DATA
3128 011452 000137 010672 JMP EMPBUFF ;GO RECHECK THE DATA
3129 011456 005337 013442 NXREAD: DEC SECCNTR
3130 011462 001404 BEQ EXIT
3131 011464 004737 010046 JSR PC,XREAD ;READ THE NEXT SECTOR
3132 011470 000137 010672 JMP EMPBUFF
3133 011474 005001 EXIT: CLR R1 ;CLEAR THE ONE READ FLAG
3134 011476 000207 RTS PC
3135
3136 011500 000000 BYTECNTR: 0
3137 011502 000000 BADBYTE: 0
3138 011504 000000 GOODBYTE: 0
3139 011506 000000 ERCNTR: 0
3140
3141 ;:*****
3142
3143 ;AN INTERRUPT DID NOT OCCURE AT A FUNCTION DONE FLAG.
3144
3145 011510 104406 NO:INTER: CKSWR
3146 011512 032777 020000 167476 BIT #SW13,@SWR ;TEST DON'T PRINT ERROR SWITCH
3147 011520 001006 BNE 1$
3148 011522 004737 010614 JSR PC,ERMSG
3149 011526 104401 016320 TYPE ,MINTER ;TYPE NO INTERRUPT ON DONE ERROR
3150 011532 104401 016162 TYPE ,MCRLF
3151 011536 104406 1$: CKSWR
3152 011540 005777 167452 TST @SWR ;TEST HALT ON ERROR SWITCH
3153 011544 100001 BPL CONT21
3154 011546 000000 HLT15: HALT ;HALT ON ERROR
3155 011550 004737 011554 CONT21: JSR PC,INTSERV ;JSR TO INTSERV AS IF IT WAS AN INTERRUPT
3156
```

.SBTTL INTERRUPT SERVICE

3157									
3158									
3159	011554	117737	167430	012174	INTSERV:	MOVB @RXDB,ASTAT			:SAVE THE ERROR AND STATUS WORD
3160	011562	005777	167420			TST @RXCS			:TEST THE ERROR FLAG
3161	011566	100444				BMI RXERROR			:THERE WAS AN ERROR GO REPORT IT
3162	011570	032737	000004	012174		BIT #BIT2,ASTAT			:IS INIT DONE SET
3163	011576	001402				BEQ 2\$:NO,CONTINUE
3164	011600	000137	012034			JMP RXPWR			:YES,REPORT POWER FAILED AND RESTART
3165	011604	032737	000003	012174	2\$:	BIT #3,ASTAT			:ARE PAR OR CRC BITS SET
3166	011612	001021				BNE 1\$:YES GO REPORT ERROR
3167	011614	132777	000040	167364		BITB #DONEBIT,@RXCS			:IS DONE SET
3168	011622	001012				BNE 3\$:IF SET RETURN TO TEST
3169	011624	104406				CKSWR			
3170	011626	032777	020000	167362		BIT #SW13,@SWR			:TEST DON'T PRINT ERROR SWITCH
3171	011634	001004				BNE 4\$:DON'T PRINT
3172	011636	104401	016353			TYPE ,MUKNINT			:TYPE UNKNOWN INTERRUPT
3173	011642	104401	016162			TYPE ,MCRLF			
3174	011646	000002			4\$:	RTI			:RETURN FROM THE INTERRUPT
3175	011650	062706	000006		3\$:	ADD #6,SP			:BYPASS INTERRUPT POINTERS ON STACK
3176	011654	000207				RTS PC			:RETURN TO PROGRAM
3177	011656	104406			1\$:	CKSWR			
3178	011660	032777	020000	167330		BIT #SW13,@SWR			:TEST DON'T PRINT ERROR SWITCH
3179	011666	001004				BNE RXERROR			
3180	011670	104401	016656			TYPE ,MNOFLAG			:TYPE NO STATUS ERROR ERROR
3181	011674	104401	016162			TYPE ,MCRLF			
3182	011700	005237	006520		RXERROR:	INC ERRORS			:AN ERROR INDICATOR
3183	011704	001775				BEQ RXERROR			
3184	011706	052737	000400	012752		BIS #BIT8,UNITSEL			:SET HARD ERROR FLAG
3185	011714	012777	000017	167264	2\$:	MOV #RDR,@RXCS			:GET THE ERROR CODE
3186	011722	004737	006622		3\$:	JSR PC,SDN			:TEST FOR DONE FLAG
3187	011726	000775				BR 3\$			
3188	011730	032777	000002	167252		BIT #2,@RXDB			:WAS THERE A PARITY ERROR
3189	011736	001403				BEQ 1\$:NO,CONTINUE
3190	011740	004737	012050			JSR PC,PARTYP			:YES,GO REPORT THE PARITY ERROR
3191	011744	000763				BR 2\$:REISSUE THE FUNCTION
3192	011746	117737	167236	012176	1\$:	MOVB @RXDB,BSTAT			:SAVE THE ERROR CODE IN B STATUS
3193	011754	104406			NOPRNT:	CKSWR			
3194	011756	032777	020000	167232		BIT #SW13,@SWR			:TEST PRINT ERROR SWITCH IN SWR
3195	011764	001020				BNE 2\$			
3196	011766	104401	016162			TYPE ,MCRLF			
3197	011772	004737	010614			JSR PC,ERMSG			:TYPE ERROR AND MESSAGES
3198	011776	104401	016254			TYPE ,MRXCS			:TYPE COMMAND STATUS REGISTER
3199	012002	013746	007730			FUNCTION,-(SP)			::SAVE FUNCTION FOR TYPEOUT
3200	012006	104403							::GO TYPE--OCTAL ASCII
3201	012010	006				.BYTE 6			::TYPE 6 DIGIT(S)
3202	012011	000				.BYTE 0			::SUPPRESS LEADING ZEROS
3203	012012	004737	012076			JSR PC,TYPADR			:TYPE ADDRESSES AND RUN CONDITIONS
3204	012016	104401	016162			TYPE ,MCRLF			
3205	012022	004737	012246			JSR PC,TYPCODE			:PRINT THE STATUS REGISTERS
3206	012026	062706	000004		2\$:	ADD #4,SP			:MOVE ERROR RETURN TO TOP OF STACK
3207	012032	000207				RTS PC			
3208									
3209	012034	104401	016727		RXPWR:	TYPE ,MRX11			:ONLY THE RX11 POWER HAS FAILED
3210	012040	104401	015332			TYPE ,\$POWER			:PRINT POWER FAILED
3211	012044	000137	001350			JMP RESTART			:GO TO RESTART

Address	Hex 1	Hex 2	Hex 3	Hex 4	Hex 5	Code	Comment
3212	012050	104401	016254			PARTYP:	TYPE ,MRXCS
3213	012054	017746	167126			MOV	@RXCS,-(SP) ;:SAVE @RXCS FOR TYPEOUT
3214	012060	104403				TYPOS	;;GO TYPE--OCTAL ASCII
3215	012062	006				.BYTE	6 ;:TYPE 6 DIGIT(S)
3216	012063	000				.BYTE	0 ;:SUPPRESS LEADING ZEROS
3217	012064	104401	016641			TYPE ,MPAR	
3218	012070	104401	016162			TYPE ,MCRLF	
3219	012074	000207				RTS PC	
3220							
3221	012076	104401	016062			TYPADR:	TYPE ,MTRK ;:TYPE TRACK ADDRESS
3222	012102	013746	013154			MOV	TARGET,-(SP) ;:SAVE TARGET FOR TYPEOUT
3223	012106	104403				TYPOS	;;GO TYPE--OCTAL ASCII
3224	012110	003				.BYTE	3 ;:TYPE 3 DIGIT(S)
3225	012111	000				.BYTE	0 ;:SUPPRESS LEADING ZEROS
3226	012112	104401	016111			TYPE ,MSECT	;:TYPE SECTOR ADDRESS
3227	012116	013737	013444	012172		MOV TSECTOR,2\$	
3228	012124	042737	177740	012172		BIC #177740,2\$	
3229	012132	013746	012172			MOV	2\$,-(SP) ;:CLEAR ALL BUT SECTOR ADDRESS
3230	012136	104403				TYPOS	;;SAVE 2\$ FOR TYPEOUT
3231	012140	002				.BYTE	2 ;:GO TYPE--OCTAL ASCII
3232	012141	000				.BYTE	0 ;:TYPE 2 DIGIT(S)
3233	012142	104401	016157			TYPE ,DBLSP	;;SUPPRESS LEADING ZEROS
3234	012146	032737	000020	012752		BIT #BIT4,UNITSEL	;:WHITCH DRIVE IS BEING USED
3235	012154	001003				BNE 1\$	
3236	012156	104401	016225			TYPE ,MUNIT0	;:TYPE UNIT 0
3237	012162	000402				BR 4\$	
3238	012164	104401	016235			1\$:	TYPE ,MUNIT1 ;:TYPE UNIT 1
3239	012170	000207				4\$:	RTS PC
3240	012172	000000				2\$:	OPEN
3241							
3242	012174	000000				ASTAT:	0
3243	012176	000000				BSTAT:	0
3244							
3245							
3246	012200	117737	167004	012174		RDCODE:	MOVB @RXDB,ASTAT ;:SAVE THE A STATUS
3247	012206	012777	000017	166772		2\$:	MOV #ORDER,@RXCS ;:READ THE B STATUS REGISTER
3248	012214	004737	006622			3\$:	JSR PC,SDN ;:WAIT FOR DONE FLAG
3249	012220	000775				BR 3\$	
3250	012222	032777	000002	166760		BIT #2,@RXDB	;:WAS THERE A PARITY ERROR
3251	012230	001403				BEQ 1\$;:NO,CONTINUE
3252	012232	004737	012050			JSR PC,PARTYP	;:YES,REPORT THE PARITY ERROR
3253	012236	000763				BR 2\$;:RETRY READING STATUS B
3254	012240	117737	166744	012176		1\$:	MOVB @RXDB,BSTAT ;:SAVE THE B STATUS CODES
3255	012246	104401	016264			TYPCODE:	TYPE ,MASTAT ;:TYPE THE CONTENTS OF THE TWO STATUS REGISTERS
3256	012252	013746	012174			MOV	ASTAT,-(SP) ;:SAVE ASTAT FOR TYPEOUT
3257	012256	104403				TYPOS	;;GO TYPE--OCTAL ASCII
3258	012260	003				.BYTE	3 ;:TYPE 3 DIGIT(S)
3259	012261	000				.BYTE	0 ;:SUPPRESS LEADING ZEROS
3260	012262	104401	016150			TYPE ,TAB	
3261	012266	104401	016300			TYPE ,MBSTAT	
3262	012272	013746	012176			MOV BSTAT,-(SP)	
3263	012276	104404				TYPON	
3264	012300	104401	016162			TYPE ,MCRLF	
3265	012304	000207				RTS PC	

3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313

012306 012704 017456
012312 005037 012564
012316 013705 012352
012322 006305
012324 000175 012330
012330 012354
012332 012366
012334 012376
012336 012440
012340 012446
012342 012466
012344 012476
012346 012516

012350 000000
012352 000000

012354 005037 012350
012360 004737 012536
012364 000775

012366 112737 000377 012350
012374 000771

```
.SBTTL PATTERN GENERATOR

;NOTE: ALL DATA PATTERNS WILL BE MODIFIED SO THE FIRST BYTE WILL
;CONTAIN THE TRACK ADDRESS. THE SECOND BYTE WILL CONTAIN THE UNIT
;NUMBER AND SECTOR ADDRESS IN WHICH THE DATA IS WRITTEN. THE MOST
;SIGNIFICANT BIT OF THIS SECOND BYTE INDICATES THE UNIT. UNIT 0
;IF '0' UNIT 1 IF '1'. THE LAST TWO BYTES CONTAIN THE CHECK SUM.
;*****

GETPATTERN:  MOV #BUFADR,R4      ;SET ADDRESS OF FIRST DATA BYTE
              CLR SUM           ;SET UP FOR ACCUMULATION OF CHECK SUM
              MOV PAT,R5       ;GET PATTERN BITS
              ASL R5
              JMP @PATTERNS(R5)

PATTERNS:   DATA0             ;000 DATA BYTE
            DATA1             ;377 DATA BYTE
            FLOAT0             ;FLOAT A 0 THROUGH ALL 1'S
            FLOAT1             ;FLOAT A 1 THROUGH ALL 0'S
            PAT125             ;125/052 DATA WORD
            PAT314             ;314/063 DATA WORD
            COUNT              ;INCRUMENT DATA PATTERN
            RANDATA            ;RANDOM DATA BYTE

DATABYTE:   0
PAT:        0

;*****

;LOAD SOFTWARE BUFFER WITH ALL ZEROS
; PAT = 0

DATA0:      CLR DATABYTE
PATGEN:     JSR PC,LOAD        ;GO LOAD THE DATA BUFFER
            BR PATGEN

;*****

;LOAD SOFTWARE BUFFER WITH ALL ONES
; PAT = 1

DATA1:      MOVB #377,DATABYTE
            BR PATGEN
```



```
3314                                     ;FLOAT A 0 THROUGH ONES IN SOFTWARE BUFFER
3315                                     ; PAT = 2
3316
3317 012376 112737 000376 012350 FLOAT0:      MOVB #376,DATABYTE      ;SET UP A ONES FIELD
3318 012404 000261                                SEC                ;SET THE C BIT TO ROTATE THROUGH THE DATA
3319 012406 012702 000000 XPATGEN:      MOV #0,R2          ;CLR R2 (CAN'T USE "CLR" IT CLEARS "C" BIT)
3320 012412 103001                                BCC 2$            ;BR IF "C" BIT IS CLEARED
3321 012414 005202                                INC R2           ;SET R2 IF "C" BIT IS SET
3322 012416 004737 012536 2$:                JSR PC,LOAD      ;GO LOAD THE DATA BUFFER
3323 012422 000241                                CLC              ;CLEAR THE "C" BIT
3324 012424 005702                                TST R2          ;IS R2 NONZERO
3325 012426 001401                                BEQ 3$
3326 012430 000261                                SEC
3327 012432 106137 012350 3$:                ROLB DATABYTE   ;YES, SET THE "C" BIT
3328 012436 000763                                BR 1$
3329
3330                                     ;*****
3331                                     ;FLOAT A 1 THROUGH ALL ZEROS IN SOFTWARE BUFFER
3332                                     ; PAT = 3
3333
3334
3335 012440 005037 012350 FLOAT1:      CLR DATABYTE
3336 012444 000757                                BR XPATGEN
3337
3338                                     ;*****
3339
3340                                     ;LOAD SOFTWARE BUFFER WITH ALTERNATING 1 AND 0 FOR
3341                                     ;ONE BYTE AND THE COMPLIMENT INTO THE NEXT
3342                                     ; PAT = 4
3343
3344 012446 112737 000125 012350 PAT125:     MOVB #125,DATABYTE
3345 012454 004737 012536 XXPATGEN:   JSR PC,LOAD
3346 012460 105137 012350                                COMB DATABYTE
3347 012464 000773                                BR XXPATGEN
3348
3349                                     ;*****
3350
3351                                     ;LOAD SOFTWARE BUFFER WITH ALTERNATING PAIRS OF 1 AND 0 AND
3352                                     ;COMPLIMENT INTO THE NEXT
3353                                     ; PAT = 5
3354
3355 012466 112737 000314 012350 PAT314:     MOVB #314,DATABYTE
3356 012474 000767                                BR XXPATGEN
3357
3358                                     ;*****
3359
3360                                     ;LOAD SOFTWARE BUFFER WITH COUNT PATTERN
3361                                     ; PAT = 6
3362
3363 012476 012737 000377 012350 COUNT:      MOV #377,DATABYTE
3364 012504 005237 012350 1$:                INC DATABYTE
3365 012510 004737 012536                                JSR PC,LOAD
3366 012514 000773                                BR 1$
```

```

3367
3368
3369
3370
3371
3372 012516 004737 012566
3373 012522 113737 012660 012350
3374 012530 004737 012536
3375 012534 000770
3376
3377 012536 063737 012350 012564
3378 012544 113724 012350
3379 012550 022704 017656
3380 012554 001401
3381 012556 000207
3382 012560 005726
3383 012562 000207
3384
3385 012564 000000
3386
3387 012566 012700 000001
3388 012572 063700 012654
3389 012576 063700 012656
3390 012602 042700 170000
3391 012606 000241
3392 012610 006100
3393 012612 006100
3394 012614 010037 012654
3395 012620 005000
3396 012622 013700 012656
3397 012626 006000
3398 012630 006000
3399 012632 063700 012654
3400 012636 042700 170000
3401 012642 010037 012656
3402 012646 010037 012660
3403 012652 000207
3404
3405 012654 001234
3406 012656 000765
3407 012660 000000
3408
  
```

;LOAD SOFTWARE BUFFER WITH RANDOM DATA PATTERN
 ; PAT = 7

RANDATA: JSR PC,RANGEN ;GET RANDOM NUMBER
 MOV B,RANUM,DATABYTE
 JSR PC,LOAD
 BR RANDATA

LOAD: ADD DATABYTE,SUM ;ACCUMULATE THE PATTERN CHECK SUM
 MOV B,DATABYTE,(R4)+ ;LOAD THE DATA BUFFER
 CMP #BUFADR+200,R4 ;HAVE 128 BYTES BEEN GENERATED
 BEQ 1\$;IF YES,RETURN TO TEST
 RTS PC ;IF NO,RETURN TO PATTERN GENERATOR
 1\$: TST (SP)+ ;TAKE PATTERN RETURN ADDRESS OF STACK
 RTS PC ;RETURN TO TEST

SUM: 0

RANGEN: MOV #1,R0
 ADD RAN1,R0
 ADD RAN2,R0
 BIC #170000,R0
 CLC
 ROL R0
 ROL R0
 MOV R0,RAN1
 CLR R0
 MOV RAN2,R0
 ROR R0
 ROR R0
 ADD RAN1,R0
 BIC #170000,R0
 MOV R0,RAN2
 MOV R0,RANUM
 RTS PC

RAN1: 001234
 RAN2: 000765
 RANUM: 0

3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451

.SBTTL UNIT SELECTION

;TEST FOR SELECTED UNITS,DRIVE READY,AND USED CONDITIONS
;ALSO CONTAINS A "HAD ERROR" FLAG TO BE TESTED AT EOP.
;THE BITS IN UNITSEL ARE USED AS FOLLOWS

- ;BIT15 =UNIT 1 SELECTED VIA SWR
- ;BIT14 =UNIT 1 USED BIT
- ;BIT8 =THIS PASS HAD AN ERROR
- ;BIT7 =UNIT 0 SELECTED VIA SWR
- ;BIT6 =UNIT 0 USED BIT
- ;BIT4 =UNIT SELECTION FOR FUNCTION WORD

::*****

```
GETUNIT:      BIT #BIT6,UNITSEL      ;WAS UNIT 0 JUST USED
              BNE 1$                ;UNIT 0 USED CHECK UNIT 1
              TSTB UNITSEL          ;WAS UNIT 0 SELECTED
              BPL 1$                ;NO GO TO UNIT 1
              BIC #40020,UNITSEL     ;CLEAR UNIT 1 USED BIT AND FUNCTION UNIT BIT
              BIS #BIT6,UNITSEL     ;SET UNIT 0 USED BIT
              RTS PC
1$:           TST UNITSEL            ;WAS UNIT 1 SELECTED
              BPL 2$                ;NO RETURN
              BIT #BIT14,UNITSEL     ;HAS UNIT 1 BEEN USED
              BNE 2$                ;YES RETURN
              BIC #BIT6,UNITSEL     ;CLEAR UNIT 0 USED BIT
              BIS #40020,UNITSEL     ;SET UNIT 1 USED BIT AND FUNCTION UNIT BIT
              RTS PC
2$:           
```

UNITSEL: 0

;TEST THAT ALL UNITS HAVE BEEN ACCESSED

```
DONE:        TST UNITSEL            ;IS UNIT 1 SELECTED
              BPL 1$                ;NO RETURN
              BIT #BIT14,UNITSEL     ;YES HAS IT BEEN USED
              BNE 1$                ;YES RETURN
              ADD #2,@SP            ;BYPASS NOT DONE RETURE ON STACK
1$:          RTS PC
```

.SBTTL TRACK SEQUENCE SELECTION

:INITIALIZE TRACK SEQUENCE

:NOTE: IF WORD SEQUEN IS CLEARED THEN TRACK SEQUENCE IS FROM 0-52-53-114 ONLY
:IF BIT 15 OF SEQUEN IS '1' THEN TRACK SELECTION IS INC. BETWEEN SELECTED OD/ID LIMITS.
:IF BIT 7 IS '1' THEN TEST 25 DECREMENT SEQUENCE IS REQUIRED.

3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498

013000	105737	013164		
013004	100442			
013006	042737	100200	001200	
013014	005737	001200		
013020	001440			
013022	052737	100000	013164	
013030	113737	001200	013154	
013036	005037	013162		
013042	113737	001201	013162	
013050	005037	013160		
013054	113737	001200	013160	
013062	013737	013162	013152	
013070	163737	013160	013152	
013076	005237	013152		
013102	052737	100200	001200	1\$:
013110	000207			
013112	012737	000013	013152	2\$:
013120	000770			
013122	012737	000004	013152	3\$:
013130	000764			

```
INITTRACK:  TSTB SEQUEN      ;IS THIS TEST 26 SPECIAL SEQUENCE
             BMI 2$          ;YES, DEC FROM TRACK 12 TO 0
             BIC #100200,OD  ;CLEAR FIRST USED BITS
             TST OD          ;TEST CONTENTS OF ID,OD FOR 0
             BEQ 3$          ;SEQUENCE WILL BE FROM 'HOME'-52-53-114-0
             BIS #BIT15,SEQUEN ;LIMITS WERE SELECTED, INC FROM OD TO ID.
             MOVB OD,TARGET  ;INIT OD AS PRESENT TRACK
             CLR XID         ;INIT WORKING ID AND OD LOCATIONS
             MOVB ID,XID
             CLR XOD
             MOVB OD,XOD
             MOV XID,TRKCNT  ;SET UP NUMBER OF TRACK MOVEMENTS
             SUB XOD,TRKCNT
             INC TRKCNT
             BIS #100200,OD  ;SET FIRST TIME BITS IN ID,OD
             RTS PC
             MOV #13,TRKCNT ;SET TRACK COUNTER
             BR 1$
             MOV #4,TRKCNT ;SET THE TRACK COUNTER
             BR 1$
```

013132	113737	013154	013156	
013140	005737	013164		
013144	001410			
013146	100446			
013150	000463			
013152	000000			
013154	000000			
013156	000000			
013160	000000			
013162	000000			
013164	000000			

```
GETTRACK:  MOVB TARGET,PRESTRK ;RESET TO PRESENT TRACK
           TST SEQUEN          ;IS THIS THE LIMITED SEQUENCE
           BEQ LIMTRK         ;YES, DOING ONLY 0-52-53-114
           BMI SEQ1           ;NO,SEQUENCE IS BETWEEN SELECTED LIMITS
           BR SEQ2            ;NO,THIS IS TEST 26 DEC SEQUENCE

TRKCNT:    0
TARGET:    0
PRESTRK:   0
XOD:       0
XID:       0
SEQUEN:    0
```

```
3499  
3500  
3501  
3502 013166 005737 001200  
3503 013172 100007  
3504 013174 012737 000052 013154  
3505 013202 042737 100000 001200  
3506 013210 000207  
3507 013212 105737 001200  
3508 013216 100007  
3509 013220 012737 000053 013154  
3510 013226 042737 000200 001200  
3511 013234 000207  
3512 013236 023727 013154 000114  
3513 013244 001404  
3514 013246 012737 000114 013154  
3515 013254 000207  
3516 013256 005037 013154  
3517 013262 000207  
3518  
3519  
3520  
3521  
3522  
3523  
3524 013264 042737 100200 001200  
3525 013272 123737 013162 013156  
3526 013300 001004  
3527 013302 113737 001200 013154  
3528 013310 000207  
3529 013312 005237 013154  
3530 013316 000207  
3531  
3532  
3533  
3534  
3535  
3536  
3537 013320 005737 001200  
3538 013324 100007  
3539 013326 042737 100200 001200  
3540 013334 012737 000012 013154  
3541 013342 000207  
3542 013344 005337 013154  
3543 013350 000207
```

LIMTRK: ;LIMITED SEQUENCE, ACCESS TRACKS 52 TO 53 TO 114 BACK TO 0
TST OD ;TEST HIGH ORDER FIRST TIME BIT
BPL 1\$;NOT SET, ON TRACK 52
MOV #52,TARGET ;GO TO TRACK 52
BIC #BIT15,OD ;CLEAR FIRST TIME BIT
RTS PC
1\$: TSTB OD ;TEST LOW ORDER FIRST TIME BIT
BPL 2\$;NOT SET, ON TRACK 53
MOV #53,TARGET ;GO TO TRACK 53
BIC #BIT7,OD
RTS PC
2\$: CMP TARGET,#114 ;IS IT ON TRACK 114
BEQ 3\$;YES,GO TO TRACK 0
MOV #114,TARGET ;NO, GO TO TRACK 114
RTS PC
3\$: CLR TARGET ;GO TO TRACK 0
RTS PC

;INCREMENT FROM OD+1 TO ID AND RETURN TO OD
;USED WHEN TRACK LIMITS ARE SELECTED

SEQ1: BIC #100200,OD ;CLEAR FIRST TIME BITS
CMPB XID,PRESTRK ;PRESENT TRACK EQUAL TO ID
BNE 1\$;NO GET NEW TRACK
MOVB OD,TARGET ;YES RETURN TO OD
RTS PC
1\$: INC TARGET ;ADD 1 TO TARGET TRACK
RTS PC

;DECREMENT FROM ID = 12 TO OD = 0
;USED IN TEST 26 ONLY

SEQ2: TST OD ;FIRST TIME BIT SET
BPL 1\$;NO GET NEXT TRACK
BIC #100200,OD ;YES CLEAR FIRST TIME BITS
MOV #12,TARGET ;MOVE OUT 10 TRACKS
RTS PC
1\$: DEC TARGET ;MOVE TO NEXT TRACK
RTS PC


```
3577 .SBTTL TYPE ROUTINE
3578
3579
3580 *****
3581 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3582 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3583 *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3584 *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3585 *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3586
3587 *CALL:
3588 *1) USING A TRAP INSTRUCTION
3589 * TYPE ,MESADR ;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
3590
3591 *OR
3592 * TYPE
3593 * MESADR
3594
3594 013534 105737 013763 $TYPE: TSTB $TPFLG ;: IS THERE A TERMINAL?
3595 013540 100002 BPL 1$ ;: BR IF YES
3596 013542 000000 HALT ;: HALT HERE IF NO TERMINAL
3597 013544 000407 BR 3$ ;: LEAVE
3598 013546 010046 1$: MOV RO,-(SP) ;: SAVE RO
3599 013550 017600 000002 MOV @2(SP),RO ;: GET ADDRESS OF ASCIZ STRING
3600 013554 112046 2$: MOVB (RO)+,-(SP) ;: PUSH CHARACTER TO BE TYPED ONTO STACK
3601 013556 001005 BNE 4$ ;: BR IF IT ISN'T THE TERMINATOR
3602 013560 005726 TST (SP)+ ;: IF TERMINATOR POP IT OFF THE STACK
3603 013562 012600 60$: MOV (SP)+,RO ;: RESTORE RO
3604 013564 062716 000002 3$: ADD #2,(SP) ;: ADJUST RETURN PC
3605 013570 000002 RTI ;: RETURN
3606 013572 122716 000011 4$: CMPB #HT,(SP) ;: BRANCH IF <HT>
3607 013576 001430 BEQ 8$
3608 013600 122716 000200 CMPB #CRLF,(SP) ;: BRANCH IF NOT <CRLF>
3609 013604 001006 BNE 5$
3610 013606 005726 TST (SP)+ ;: POP <CR><LF> EQUIV
3611 013610 104401 TYPE ;: TYPE A CR AND LF
3612 013612 013765 $CRLF
3613 013614 105037 013750 CLRB $CHARCNT ;: CLEAR CHARACTER COUNT
3614 013620 000755 BR 2$ ;: GET NEXT CHARACTER
3615 013622 004737 013704 5$: JSR PC,$TYPEC ;: GO TYPE THIS CHARACTER
3616 013626 123726 013762 6$: CMPB $FILLC,(SP)+ ;: IS IT TIME FOR FILLER CHARS.?
3617 013632 001350 BNE 2$ ;: IF NO GO GET NEXT CHAR.
3618 013634 013746 013760 MOV $NULL,-(SP) ;: GET # OF FILLER CHARS. NEEDED
3619 ;: AND THE NULL CHAR.
3620 013640 105366 000001 7$: DECB 1(SP) ;: DOES A NULL NEED TO BE TYPED?
3621 013644 002770 BLT 6$ ;: BR IF NO--GO POP THE NULL OFF OF STACK
3622 013646 004737 013704 JSR PC,$TYPEC ;: GO TYPE A NULL
3623 013652 105337 013750 DECB $CHARCNT ;: DO NOT COUNT AS A COUNT
3624 013656 000770 BR 7$ ;: LOOP
3625
3626 ;HORIZONTAL TAB PROCESSOR
3627
3628 013660 112716 000040 8$: MOVB #' ,(SP) ;: REPLACE TAB WITH SPACE
3629 013664 004737 013704 9$: JSR PC,$TYPEC ;: TYPE A SPACE
3630 013670 132737 000007 013750 BITB #7,$CHARCNT ;: BRANCH IF NOT A^
3631 013676 001372 BNE 9$ ;: TAB STOP
3632 013700 005726 TST (SP)+ ;: POP SPACE OFF STACK
```

3633	013702	000724			BR	2\$::GET NEXT CHARACTER
3634	013704	105777	000044		\$TYPEC: TSTB	@STPS	::WAIT UNTIL PRINTER IS READY
3635	013710	100375			BPL	\$TYPEC	
3636	013712	116677	000002	000036	MOVB	2(SP),@STPB	::LOAD CHAR TO BE TYPED INTO DATA REG.
3637	013720	122766	000015	000002	CMPB	#CR,2(SP)	::IS CHARACTER A CARRIAGE RETURN?
3638	013726	0C1003			BNE	1\$::BRANCH IF NO
3639	013730	105037	013750		CLRB	\$CHARCNT	::YES--CLEAR CHARACTER COUNT
3640	013734	000406			BR	\$TYPEX	::EXIT
3641	013736	122766	000012	000002	1\$: CMPB	#LF,2(SP)	::IS CHARACTER A LINE FEED?
3642	013744	001402			BEQ	\$TYPEX	::BRANCH IF YES
3643	013746	105227			INCB	(PC)+	::COUNT THE CHARACTER
3644	013750	000000			\$CHARCNT: .WORD	0	::CHARACTER COUNT STORAGE
3645	013752	000207			\$TYPEX: RTS	PC	
3646							
3647	013754	177564			\$TPS: .WORD	177564	::TTY PRINTER STATUS REG. ADDRESS
3648	013756	177566			\$TPB: .WORD	177566	::TTY PRINTER BUFFER REG. ADDRESS
3649	013760	000			\$NULL: .BYTE	0	::CONTAINS NULL CHARACTER FOR FILLS
3650	013761	002			\$FILLS: .BYTE	2	::CONTAINS # OF FILLER CHARACTERS REQUIRED
3651	013762	012			\$FILLC: .BYTE	12	::INSERT FILL CHARS. AFTER A "LINE FEED"
3652	013763	000			\$TPFLG: .BYTE	0	::"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
3653	013764	077			\$QUES: .ASCII	"?"	::QUESTION MARK
3654	013765	015			\$CRLF: .ASCII	<15>	::CARRIAGE RETURN
3655	013766	000012			\$LF: .ASCII	<12>	::LINEFEED

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOS    ;;CALL FOR TYPEOUT
*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE   M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS

```

```

*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC

```

```

*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON    ;;CALL FOR TYPEOUT

```

```

*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

```

```

*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC    ;;CALL FOR TYPEOUT

```

3681	013770	017646	000000		\$TYPOS:	MOV	@(SP),-(SP)	;;PICKUP THE MODE
3682	013774	116637	000001	014213		MOVB	1(SP),%OFILL	;;LOAD ZERO FILL SWITCH
3683	014002	112637	014215			MOVB	(SP)+,%SOMODE+1	;;NUMBER OF DIGITS TO TYPE
3684	014006	062716	000002			ADD	#2,(SP)	;;ADJUST RETURN ADDRESS
3685	014012	000406				BR	\$TYPON	
3686	014014	112737	000001	014213	\$TYPOC:	MOVB	#1,%OFILL	;;SET THE ZERO FILL SWITCH
3687	014022	112737	000006	014215		MOVB	#6,%SOMODE+1	;;SET FOR SIX(6) DIGITS
3688	014030	112737	000005	014212	\$TYPON:	MOVB	#5,%SOCNT	;;SET THE ITERATION COUNT
3689	014036	010346				MOV	R3,-(SP)	;;SAVE R3
3690	014040	010446				MOV	R4,-(SP)	;;SAVE R4
3691	014042	010546				MOV	R5,-(SP)	;;SAVE R5
3692	014044	113704	014215			MOVB	%SOMODE+1,R4	;;GET THE NUMBER OF DIGITS TO TYPE
3693	014050	005404				NEG	R4	
3694	014052	062704	000006			ADD	#6,R4	;;SUBTRACT IT FOR MAX. ALLOWED
3695	014056	110437	014214			MOVB	R4,%SOMODE	;;SAVE IT FOR USE
3696	014062	113704	014213			MOVB	%OFILL,R4	;;GET THE ZERO FILL SWITCH
3697	014066	016605	000012			MOV	12(SP),R5	;;PICKUP THE INPUT NUMBER
3698	014072	005003				CLR	R3	;;CLEAR THE OUTPUT WORD
3699	014074	006105			1\$:	ROL	R5	;;ROTATE MSB INTO 'C'
3700	014076	000404				BR	3\$;;GO DO MSB
3701	014100	006105			2\$:	ROL	R5	;;FORM THIS DIGIT
3702	014102	006105				ROL	R5	
3703	014104	006105				ROL	R5	
3704	014106	010503				MOV	R5,R3	
3705	014110	006103			3\$:	ROL	R3	;;GET LSB OF THIS DIGIT
3706	014112	105337	014214			DECB	%SOMODE	;;TYPE THIS DIGIT?
3707	014116	100016				BPL	7\$;;BR IF NO
3708	014120	042703	177770			BIC	#177770,R3	;;GET RID OF JUNK
3709	014124	001002				BNE	4\$;;TEST FOR 0
3710	014126	005704				TST	R4	;;SUPPRESS THIS 0?
3711	014130	001403				BEQ	5\$;;BR IF YES

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

;*SAVE R0-R5
;*CALL:
;* SAVREG
;*UPON RETURN FROM \$SAVREG THE STACK WILL LOOK LIKE:
;*
;*TOP---(+16)
;* +2---(+18)
;* +4---R5
;* +6---R4
;* +8---R3
;*+10---R2
;*+12---R1
;*+14---R0

\$SAVREG:

MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PS OF CALL
MOV 22(SP),-(SP) ;;SAVE PC OF CALL
RTI

;*RESTORE R0-R5

;*CALL

;* RESREG

\$RESREG:

MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
MOV (SP)+,22(SP) ;;RESTORE PS OF CALL
MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW
MOV (SP)+,R5 ;;POP STACK INTO R5
MOV (SP)+,R4 ;;POP STACK INTO R4
MOV (SP)+,R3 ;;POP STACK INTO R3
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTI

3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750 014216
3751 014216 010046
3752 014220 010146
3753 014222 010246
3754 014224 010346
3755 014226 010446
3756 014230 010546
3757 014232 016646 000022
3758 014236 016646 000022
3759 014242 016646 000022
3760 014246 016646 000022
3761 014252 000002
3762
3763
3764
3765
3766 014254
3767 014254 012666 000022
3768 014260 012666 000022
3769 014264 012666 000022
3770 014270 012666 000022
3771 014274 012605
3772 014276 012604
3773 014300 012603
3774 014302 012602
3775 014304 012601
3776 014306 012600
3777 014310 000002

3834 014520 004737 013704
3835 014524 021627 000060
3836 014530 002420
3837 014532 021627 000067
3838 014536 003015
3839 014540 042726 000060
3840 014544 005766 000002
3841 014550 001403
3842 014552 006316
3843 014554 006316
3844 014556 006316
3845 014560 005266 000002
3846 014564 056616 177776
3847 014570 000707
3848 014572 104401 013764
3849 014576 000720

16\$: JSR PC,\$TYPEC ;;ECHO CHAR
CMP (SP),#60 ;;CHAR < 0?
BLT 18\$;;BRANCH IF YES
CMP (SP),#67 ;;CHAR > 7?
BGT 18\$;;BRANCH IF YES
BIC #60,(SP)+ ;;STRIP-OFF ASCII
TST 2(SP) ;;IS THIS THE FIRST CHAR
BEQ 17\$;;BRANCH IF YES
ASL (SP) ;;NO, SHIFT PRESENT
ASL (SP) ;; CHAR OVER TO MAKE
ASL (SP) ;; ROOM FOR NEW ONE.
17\$: INC 2(SP) ;;KEEP COUNT OF CHAR
BIS -2(SP),(SP) ;;SET IN NEW CHAR
BR 7\$;;GET THE NEXT ONE
18\$: TYPE ,SQUES ;;TYPE ?<CR><LF>
BR 20\$;;SIMULATE CONTROL-U
.DSABL LSB

*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

*CALL:

* R0CHR ;;INPUT A SINGLE CHARACTER FROM THE TTY
* RETURN HERE ;;CHARACTER IS ON THE STACK
* ;;WITH PARITY BIT STRIPPED OFF

3861 014600 011646
3862 014602 016666 000004 000002
3863 014610 105777 177476
3864 014614 100375
3865 014616 117766 177472 000004
3866 014624 042766 177600 000004
3867 014632 026627 000004 000023
3868 014640 001013
3869 014642 105777 177444
3870 014646 100375
3871 014650 117746 177440
3872 014654 042716 177600
3873 014660 022627 000021
3874 014664 001366
3875 014666 000750
3876 014670 026627 000004 000140
3877 014676 002407
3878 014700 026627 000004 000175
3879 014706 003003
3880 014710 042766 000040 000004
3881 014716 000002

SRDCHR: MOV (SP),-(SP) ;;PUSH DOWN THE PC
MOV 4(SP),2(SP) ;;SAVE THE PS
1\$: TST @STKS ;;WAIT FOR
BPL 1\$;;A CHARACTER
MOVB @STKB,4(SP) ;;READ THE TTY
BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
CMP 4(SP),#23 ;;IS IT A CONTROL-S?
BNE 3\$;;BRANCH IF NO
2\$: TST @STKS ;;WAIT FOR A CHARACTER
BPL 2\$;;LOOP UNTIL ITS THERE
MOVB @STKB,-(SP) ;;GET CHARACTER
BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
BNE 2\$;;IF NOT DISCARD IT
BR 1\$;;YES, RESUME
3\$: CMP 4(SP),#140 ;;IS IT UPPER CASE?
BLT 4\$;;BRANCH IF YES
CMP 4(SP),#175 ;;IS IT A SPECIAL CHAR?
BGT 4\$;;BRANCH IF YES
BIC #40,4(SP) ;;MAKE IT UPPER CASE
4\$: RTI ;;GO BACK TO USER

*THIS ROUTINE WILL INPUT A STRING FROM THE TTY

*CALL:

* RDLIN ;;INPUT A STRING FROM THE TTY
* RETURN HERE ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
* ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

3889 014720 010346

SRDLIN: MOV R3,-(SP) ;;SAVE R3

CZ
CZ
CP
CR
CR
CR
DA
DA
DA
DA
DA
DA
DBI
DBI
DDI
DDI
DD
DEJ
DEJ
DII
DIS
DIS
DOP
DOP
DRV
DSW
DTE
DOL
EBI
EMP
EMP
EMP
EMP
EMT
EPC
ERC
ERM
ERR
ERR
EXI
FAS
FBE
FBI
FIL
FIL
FIL
FIL
FIR
FIR
FLO
FLO
FUN
GET
GET
GET
GET
GNS

3890	014722	012703	015026	18:	MOV	#\$TTYIN,R3	::GET ADDRESS
3891	014726	022703	015036	28:	CMP	#\$TTYIN+8.,R3	::BUFFER FULL?
3892	014732	101405			BLOS	48	::BR IF YES
3893	014734	104407			RDCHR		::GO READ ONE CHARACTER FROM THE TTY
3894	014736	112613			MOVB	(SP)+,(R3)	::GET CHARACTER
3895	014740	122713	000177	108:	CMPB	#177,(R3)	::IS IT A RUBOUT
3896	014744	001003			BNE	38	::SKIP IF NOT
3897	014746	104401	013764	48:	TYPE	,8QUES	::TYPE A '?'
3898	014752	000763			BR	18	::CLEAR THE BUFFER AND LOOP
3899	014754	111337	015024	38:	MOVB	(R3),98	::ECHO THE CHARACTER
3900	014760	104401	015024		TYPE	,98	
3901	014764	122723	000015		CMPB	#15,(R3)+	::CHECK FOR RETURN
3902	014770	001356			BNE	28	::LOOP IF NOT RETURN
3903	014772	105063	177777		CLRB	-1(R3)	::CLEAR RETURN (THE 15)
3904	014776	104401	013766		TYPE	,8LF	::TYPE A LINE FEED
3905	015002	012603			MOV	(SP)+,R3	::RESTORE R3
3906	015004	011646			MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
3907	015006	016666	000004	000002	MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT
3908	015014	012766	015026	000004	MOV	#\$TTYIN,4(SP)	
3909	015022	000002			RTI		::RETURN
3910	015024	000		98:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
3911	015025	000			.BYTE	0	::TERMINATOR
3912	015026	000010			STTYIN:	.BLKB 8.	::RESERVE 8 BYTES FOR TTY INPUT
3913	015036	052536	005015	000	\$CNTLU:	.ASCIZ /^U/<15><12>	::CONTROL 'U'
3914	015043	136	006507	000012	\$CNTLG:	.ASCIZ /^G/<15><12>	::CONTROL 'G'
3915	015050	005015	053523	020122	\$MSWR:	.ASCIZ <15><12>/SWR = /	
3916	015056	020075	000				
3917	015061	040	047040	053505	\$MNEW:	.ASCIZ / NEW = /	
3918	015066	036440	000040				
3919	015072	000			\$AUTOB:	.BYTE 0	::AUTO MODE FLAG
3920	015073	000			\$INTAG:	.BYTE 0	::INTERRUPT MODE FLAG

CZ
CZ
GC
GT
HA
HA
HE
HL
HL
HL
HL
HL
HO
HT
HU
ID
IL
IN
IN
IN
IN
IO
KR
LA
LF
LI
LI
LO
LO
LO
LO
MAE
MAS
MBA
MBA
MBS
MCC
MCR
MCR
MDC
MDE
MDT
MEM
MER
MER
MFI
MFI
MGO
MHU
MID
MIL
MIN
MLA

```

3921      .SBTTL TRAP DECODER
3922
3923      ;*****
3924      ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3925      ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3926      ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3927      ;*GO TO THAT ROUTINE.
3928
3929 015074 010046      $TRAP:  MOV    RO,-(SP)          ;;SAVE RO
3930 015076 016600 000002  MOV    2(SP),RO      ;;GET TRAP ADDRESS
3931 015102 005740      TST    -(RO)         ;;BACKUP BY 2
3932 015104 111000      MOVB   (RO),RO      ;;GET RIGHT BYTE OF TRAP
3933 015106 006300      ASL    RO           ;;POSITION FOR INDEXING
3934 015110 016000 015130  MOV    $TRPAD(RO),RO ;;INDEX TO TABLE
3935 015114 000200      RTS    PO           ;;GO TO ROUTINE
3936
3937
3938      ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3939
3940 015116 011646      $TRAP2: MOV   (SP),-(SP) ;;MOVE THE PC DOWN
3941 015120 016666 000004 000002  MOV   4(SP),2(SP) ;;MOVE THE PSW DOWN
3942 015126 000002      RTI                ;;RESTORE THE PSW
3943
3944      .SBTTL TRAP TABLE
3945
3946      ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3947      ;*BY THE "TRAP" INSTRUCTION.
3948
3949      :
3950      : ROUTINE
3951      : -----
3951 015130 015116      $TRPAD: .WORD  $TRAP2
3952 015132 013534      $TYPE  ;;CALL=TYPE   TRAP+1(104401) TTY TYPEOUT ROUTINE
3953 015134 014014      $TYPOC ;;CALL=TYPOC  TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3954 015136 013770      $TYPOS ;;CALL=TYPOS   TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
3955 015140 014030      $TYPON ;;CALL=TYPON   TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
3956
3957 015142 014366      $GTSWR ;;CALL=GTSWR   TRAP+5(104405) GET SOFT-SWR SETTING
3958
3959 015144 014316      $CKSWR ;;CALL=CKSWR   TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
3960 015146 014600      $RDCHR ;;CALL=RDCHR  TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
3961 015150 014720      $RDLIN ;;CALL=RDLIN   TRAP+10(104410) TTY TYPEIN STRING ROUTINE
3962 015152 014216      $SAVREG ;;CALL=SAVREG  TRAP+11(104411) SAVE R0-R5 ROUTINE
3963 015154 014254      $RESREG ;;CALL=RESREG TRAP+12(104412) RESTORE R0-R5 ROUTINE
3964 015156 006544      XSUBSCOPE ;;CALL=SUBSCOPE TRAP+13(104413)

```



```
4010 .SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
4011
4012
4013 *****
4014 *THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
4015 *UNSIGNED DECIMAL ASCIZ NUMBER.
4016 *CALL
4017 *      MOV      NUMBER,-(SP)      ;;PUT BINARY NUMBER ON THE STACK
4018 *      JSR      PC,@#SSB2D      ;;CALL
4019 *      RETURN     ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK
4020
4021 015342 016637 000002 015372 $SB2D: MOV      2(SP),1$      ;;SAVE BINARY NUMBER
4022 015350 012746 015372      MOV      #1$,-(SP)      ;;SET POINTER
4023 015354 004737 015376      JSR      PC,@#$DB2D      ;;CALL DOUBLE LENGTH CONVERT
4024 015360 062716 000005      ADD      #5,(SP)      ;;ONLY ALLOW FIVE CHARACTERS
4025 015364 012666 000002      MOV      (SP)+,2(SP)    ;;PICKUP POINTER
4026 015370 000207      RTS      PC      ;;RETURN
4027 015372 000000 000000 1$:      .WORD 0,0
4028 .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
4029
4030 *****
4031 *THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
4032 *DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
4033 *POSITIVE.
4034 *CALL
4035 *      MOV      #PNTR,-(SP)      ;;POINTER TO LOW WORD OF BINARY NUMBER
4036 *      JSR      PC,@#$DB2D
4037 *      RETURN     ;;THE FIRST ADDRESS OF ASCIZ
4038 *                ;;IS ON THE STACK
4039
4040
4041 015376 104411 $DR2D: SAVREG      ;;SAVE REGISTERS
4042 015400 016602 000002      MOV      2(SP),R2      ;;PICKUP THE DATA POINTER
4043 015404 012700 015556      MOV      #$DECVL,R0    ;;GET ADDRESS OF "$DECVL" STRING
4044 015410 010066 000002      MOV      R0,2(SP)      ;;PUT ADDRESS OF ASCIZ STRING ON STACK
4045 015414 012201      MOV      (R2)+,R1      ;;PICKUP THE BINARY NUMBER
4046 015416 012202      MOV      (R2)+,R2
4047 015420 012737 000012 015474      MOV      #10.,4$      ;;SET UP TO DO 10 CONVERSIONS
4048 015426 012704 015506      MOV      #STNPWR,R4    ;;ADDRESS OF TEN POWER
4049 015432 012705 015510      MOV      #STNPWR+2,R5
4050 015436 005003 1$:      CLR      R3      ;;CLEAR PARTIAL
4051 015440 161401 2$:      SUB      (R4),R1      ;;SUBTRACT TEN POWER
4052 015442 005602      SBC      R2
4053 015444 161502      SUB      (R5),R2
4054 015446 002402      BLT      3$      ;;BR IF TEN POWER TOO LARGE
4055 015450 005203      INC      R3      ;;ADD 1 TO PARTIAL
4056 015452 000772      BR      2$      ;;LOOP
4057 015454 062401 3$:      ADD      (R4)+,R1      ;;RESTORE SUBTRACTED VALUE
4058 015456 005502      ADC      R2
4059 015460 062402      ADD      (R4)+,R2
4060 015462 022525      CMP      (R5)+,(R5)+  ;;MOVE TO NEXT TEN POWER
4061 015464 052703 000060      BIS      #'0,R3      ;;CHANGE PARTIAL TO ASCII
4062 015470 110320      MOVB     R3,(R0)+     ;;SAVE IT
4063 015472 005327      DEC      (PC)+      ;;DONE?
4064 015474 000000 4$:      .WORD 0
4065 015476 001357      BNE     1$      ;;BR IF NO
```

CZRXBFO RX11 INTERFACE TEST
CZRXBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 PAGE 93
DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

E 10

SEQ 0121

4066	015500	105020			
4067	015502	104412			
4068	015504	000207			
4069	015506	145000			
4070	015510	035632	\$TNPWR:	145000	
4071	015512	160400		35632	
4072	015514	002765		160400	::1.0E08
4073	015516	113200		2765	
4074	015520	000230		113200	::1.0E07
4075	015522	041100		230	
4076	015524	000017		041100	::1.0E06
4077	015526	103240		17	
4078	015530	000001		103240	::1.0E05
4079	015532	023420		1	
4080	015534	000000		23420	::1.0E04
4081	015536	001750		0	
4082	015540	000000		1750	::1.0E03
4083	015542	000144		0	
4084	015544	000000		144	::1.0E02
4085	015546	000012		0	
4086	015550	000000		12	::1.0E01
4087	015552	000001		0	
4088	015554	000000		1	::1.0E00
4089	015556	000014	\$DECVL:	0	
4090				.BLKB 12.	::RESERVE STORAGE FOR ASCII STRING

CZ
CZ

TR
TR
TR
TS

TY
TY
TY

TY
TY
TY
TY

TO
T1
T10
T11
T12
T13
T14
T15
T16
T17

T2
T20
T21
T21
T22
T23
T24
T24
T25
T26

T3
T4
T5
T6
T6F
T7
T7E

UNI

URI
URT
URT
URT
URT
UTC

4091
4092
4093
4094
4095
4096
4097
4098 015572 010046
4099 015574 016600 000004
4100 015600 010037 015632
4101 015604 105710
4102 015606 001406
4103 015610 122710 000060
4104 015614 001005
4105 015616 112720 000040
4106 015622 000770
4107 015624 112740 000060
4108 015630 104401
4109 015632 000000
4110 015634 012600
4111 015636 012616
4112 015640 000207
4113
4114
4115
4116 015642 012546
4117 015644 004737 015342
4118 015650 004737 015572
4119 015654 000205

```
*****  
:TYPE NUMERICAL ASCII STRING,RIGHT JUSTIFIED  
:REPLACING LEADING ZEROS WITH SPACES.  
:FIRST ADDRESS OF ASCII STRING MUST BE ON TOP OF THE STACK  
RTJUST:      MOV R0,-(SP)          ;SAVE R0  
              MOV 4(SP),R0       ;PICK JP ADDRESS OF ASCII STRING  
              MOV R0,3$         ;SAVE ADDRESS FOR TYPE OUT  
1$:          TSTB (R0)           ;IS THIS THE TERMINATOR  
              BEQ 2$             ;IF YES TYPE IT OUT  
              CMPB #'0,(R0)      ;IS IT A ZERO  
              BNE 4$             ;IF NO GO PRINT IT  
              MOVB #' ,(R0)+     ;IF YES REPLACE IT WITH A SPACE  
              E? 1$             ;TEST NEXT CHAR.  
2$:          M,,2 #'0,-(R0)      ;STRING OFF ALL ZEROS,PUT BACK THE LAST ONE  
4$:          TYPE                ;TYPE THE STRING  
3$:          OPEN  
              MOV (SP)+,R0       ;RESTORE R0  
              MOV (SP)+,(SP)     ;RESTORE THE STACK  
              RTS PC            ;RETURN  
:TYPES 16 BIT WORD IN DECIMAL  
SGLDEC:      MOV (R5)+,-(SP)     ;PUT NUMBER TO BE TYPED ON STACK  
              JSR PC,@#$$SB20    ;CONVERT NUMBER TO DECIMAL  
              JSR PC,RTJUST      ;TYPE THE DECIMAL NUMBER  
              RTS R5
```

CZ
CZ
WT
XEI
XFI
XMC
XIE
XOI
XPI
XRE
XSA
XSC
XSE
XSL
XWR
XWT
XXP
SAU
SCH
SCK
SCH
SCN
SCR
SDB
SDE
SFI
SFI
SGT
SHD
SIL
SIN
SLF
SHA
SHNI
SHSI
SHUI
SOCI
SOM
SPO
SPUI
SPUI
SPUI
SPUI
SQUE
SRD
SRDE
SRDL
SRDC
SRDS
SRES
SR2A
SSAV
SSAV
SSB2
SSE1
BSTL
SSWF
STKE

CZRXBFO RX11 INTERFACE TEST
CZRXBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 M 10
MESSAGES PAGE 96

SEQ 0124

4176	016200	044504	044524	047117		
4177	016206	035123	020040	042524		
4178	016214	052123	050040	020103		
4179	016222	020075	000			
4180						
4181	016225	125	044516	020124	MUNIT0:	.ASCIZ 'UNIT 0 ''
4182	016232	020060	000			
4183						
4184	016235	125	044516	020124	MUNIT1:	.ASCIZ 'UNIT 1 ''
4185	016242	020061	000			
4186						
4187	016245	117	046116	020131	MONLY:	.ASCIZ 'ONLY ''
4188	016252	000040				
4189						
4190	016254	054122	051503	036440	MRXCS:	.ASCIZ 'RXCS = ''
4191	016262	000040				
4192						
4193	016264	052123	052101	051525	MASTAT:	.ASCIZ 'STATUS A = ''
4194	016272	040440	036440	000040		
4195						
4196	016300	052123	052101	051525	MBSTAT:	.ASCIZ 'STATUS B - ''
4197	016306	041040	036440	000040		
4198						
4199	016314	005015	000012		DBLLF:	.ASCIZ <15><12><12>
4200						
4201	016320	047516	044440	052116	MINTER:	.ASCIZ 'NO INTERRUPT AT DONE ERROR''
4202	016326	051105	052522	052120		
4203	016334	040440	020124	047504		
4204	016342	042516	042440	051122		
4205	016350	051117	000			
4206						
4207	016353	125	045516	047516	MUKNINT:	.ASCIZ 'UNKNOWN INTERRUPT''
4208	016360	047127	044440	052116		
4209	016366	051105	052522	052120		
4210	016374	000				
4211						
4212	016375	124	052117	046101	MERCT:	.ASCIZ 'TOTAL READ CHECK ERRORS - ''
4213	016402	051040	040505	020104		
4214	016410	044103	041505	020113		
4215	016416	051105	047522	051522		
4216	016424	036440	000040			
4217						
4218	016430	044506	046114	052502	MFIL:	.ASCIZ 'FILLBUFFER ''
4219	016436	043106	051105	000040		
4220						
4221	016444	046505	052120	041131	MEMPTY:	.ASCIZ 'EMPTYBUFFER ''
4222	016452	043125	042506	020122		
4223	016460	000				
4224						
4225	016461	040	051124	041501	MLIMTRK:	.ASCIZ ' TRACKS 52,53,114,0 ''
4226	016466	051513	032440	026062		
4227	016474	031465	030454	032061		
4228	016502	030054	020040	000		
4229						
4230	016507	117	036504	000	MOD:	.ASCIZ '00-''
4231						

CZ
CZI
CO
ENI
ERI
ES
GE1
GE1
MUL
M77
M78
NEW
POP
PUS
REP
SCO
SET
SET
SET
SKI
SLA
SPA
STA
SWR
TRM
TYP
TYP
TYPI
TYPI
TYPI
TYPI
TYPI
TYP
SSE
SSNI
SSSE
SSSI
.EQU
.HEA
.KT
.SE
.SWP
.SAC
.SAF
.SAF
.SAF
.SAC
.SCA
.SCP
.SDE
.SDE

SEQ 0125

4232	016513	040	044440	036504	MID:	.ASCIZ " ID="
4233	016520	000				
4234						
4235	016521	040	043040	051111	MFIRST:	.ASCIZ " FIRST="
4236	016526	052123	000075			
4237						
4238	016532	020040	040514	052123	MLAST:	.ASCIZ " LAST="
4239	016540	000075				
4240						
4241	016542	051103	020103	051105	MBADCRC:	.ASCIZ "CRC ERROR NO DATA ERROR"
4242	016550	047522	020122	047516		
4243	016556	042040	052101	020101		
4244	016564	051105	047522	000122		
4245						
4246	016572	042522	042101	000040	MREAD:	.ASCIZ "READ "
4247						
4248	016600	040504	040524	041440	MCRC:	.ASCIZ "DATA CRC ERROR"
4249	016606	041522	042440	051122		
4250	016614	051117	000			
4251						
4252	016617	123	042505	020113	MSEEK:	.ASCIZ "SEEK ERROR"
4253	016624	051105	047522	000122		
4254						
4255	016632	051127	052111	020105	MWRITE:	.ASCIZ "WRITE "
4256	016640	000				
4257						
4258	016641	120	051101	052111	MPAR:	.ASCIZ "PARITY ERROR"
4259	016646	020131	051105	047522		
4260	016654	000122				
4261						
4262	016656	051105	047522	020122	MNOFLAG:	.ASCIZ "ERROR FLAG ERROR"
4263	016664	046106	043501	042440		
4264	016672	051122	051117	000		
4265						
4266	016677	102	042101	000	MBAD:	.ASCIZ "BAD"
4267						
4268	016703	040	000		SPACE:	.ASCIZ <40>
4269						
4270	016705	107	047517	000104	MGOOD:	.ASCIZ "GOOD"
4271						
4272	016712	020040	044103	041505	MSUM:	.ASCIZ " CHECK SUM "
4273	016720	020113	052523	020115		
4274	016726	000				
4275						
4276	016727	015	051012	030530	MRX11:	.ASCIZ <15><12>"RX11 / RXV11"
4277	016734	020061	020057	054122		
4278	016742	030526	000061			
4279						
4280	016746	005015	041412	051132	MREV:	.ASCIZ <15><12><12> "CZRXBFO RX11 INTERFACE TEST" <15><12>
4281	016754	041130	030106	051040		
4282	016762	030530	020061	047111		
4283	016770	042524	043122	041501		
4284	016776	020105	042524	052123		
4285	017004	005015	000			
4286						
4287	017007	015	052412	042516	LOC4M:	.ASCIZ <15><12>"UNEXPECTED TRAP TO LOC. 4 OCCURRED"

CZI
 CZI
 .SI
 .SE
 .SE
 .SE
 .SM
 .SP
 .SR
 .SR
 .SR
 .SR
 .SS
 .SS
 .SS
 .SS
 .SS
 .ST
 .ST
 .ST
 .ST
 .S4
 .11
 .AI
 .ERI
 .DS
 .RUP
 .RUP
 .COR

LZRXBFO RX11 INTERFACE TEST
CZRXBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 J 10
MESSAGES PAGE 98

SEQ 0126

4288	017014	050130	041505	042524	
4289	017022	020104	051124	050101	
4290	017030	052040	020117	047514	
4291	017036	027103	032040	047440	
4292	017044	041503	051125	042522	
4293	017052	000104			
4294					
4295	017054	005015	047125	054105	LOC10M: .ASCIZ <15><12>'UNEXPECTED TRAP TO LOC. 10 OCCURRED''
4296	017062	042520	052103	042105	
4297	017070	052040	040522	020120	
4298	017076	047524	046040	041517	
4299	017104	020056	030061	047440	
4300	017112	041503	051125	042522	
4301	017120	000104			
4302					
4303	017122	050075	000103		PCM: .ASCIZ ''=PC''
4304					
4305	017126	005015	051124	041501	OD2BIG: .ASCII <15><12>'TRACK LIMITS SELECTED OUT OF RANGE''
4306	017134	020113	044514	044515	
4307	017142	051524	051440	046105	
4308	017150	041505	042524	020104	
4309	017156	052517	020124	043117	
4310	017164	051040	047101	042507	
4311	017172	005015	042504	040506	.ASCIZ <15><12>'DEFAULTING TO ''
4312	017200	046125	044524	043516	
4313	017206	052040	020117	000	
4314					
4315	017213	015	051412	041505	S2BIG: .ASCII <15><12>'SECTOR LIMITS SELECTED OUT OF RANGE''
4316	017220	047524	020122	044514	
4317	017226	044515	051524	051440	
4318	017234	046105	041505	042524	
4319	017242	020104	052517	020124	
4320	017250	043117	051040	047101	
4321	017256	042507			
4322	017260	005015	042504	040506	.ASCIZ <15><12>'DEFAULTING TO ''
4323	017266	046125	044524	043516	
4324	017274	052040	020117	000	
4325					
4326	017301	015	041412	052501	DOLOAD: .ASCII <15><12>'CAUTION - IF YOU DESIRE TO TEST UNIT 0''
4327	017306	044524	047117	026440	
4328	017314	044440	020106	047531	
4329	017322	020125	042504	044523	
4330	017330	042522	052040	020117	
4331	017336	042524	052123	052440	
4332	017344	044516	020124	060	
4333	017351	015	051012	050105	.ASCII <15><12>'REPLACE LOAD MEDIUM WITH A SCRATCH DISKETTE''
4334	017356	040514	042503	046040	
4335	017364	040517	020104	042515	
4336	017372	044504	046525	053440	
4337	017400	052111	020110	020101	
4338	017406	041523	040522	041524	
4339	017414	020110	044504	045523	
4340	017422	052105	042524		
4341	017426	005015	044124	047105	.ASCIZ <15><12>'THEN PRESS CONTINUE''<15><12>
4342	017434	050040	042522	051523	
4343	017442	041440	047117	044524	

CZRXBFO RX11 INTERFACE TEST
CZRXBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 K 10
MESSAGES PAGE 99

SEQ 0127

4344 017450 052516 006505 000012

4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355

.EVEN

::*****

;THE FOLLOWING LOCATIONS ARE USED FOR DATA STORAGE,RETRY COUNTERS
;ACCESS COUNTERS ETC.

017456 000200
000001

BUFADR: .BLKB 200

.END

WTRDCK	007114	2453	2736#				
XERROR	006260	221	2518#				
XFRBYT	007236	2761#					
XHOME	010032	2871	2883#				
XID	013162	3469*	3470*	3473	3496#	3525	
XOD	013160	3471*	3472*	3474	3495#		
XPATGE	012404	3318#	3336				
XREAD	010046	2890#	2913	3131			
XSA202	001612	444	457	462	464	468	470#
XSCOPE	006524	217	2587#				
XSDN	006764	2638	2694#				
XSUBSC	006544	2602#	3964				
XWRITE	007166	2750#	2767	2805			
XWTRDC	007120	2737#	2746				
XXPATG	012454	3345#	3347	3356			
\$AUTOB	015072	428*	3798	3919#			
\$CHARC	013750	3613*	3623*	3630	3639*	3644#	
\$CKSWR	014316	3790#	3959				
\$CNTLG	015043	3801	3914#				
\$CNTLU	015036	3818	3913#				
\$CRLF	013765	3612	3654#	3829	3913		
\$DB2D	015376	4023	4041#				
\$DECLV	015556	4043	4089#				
\$FILLC	013762	3616	3651#				
\$FILLS	013761	3650#					
\$GTSWR	014366	3802#	3957				
\$HD =	000003	18	19				
\$ILLUP	015324	3969	3985	4004#			
\$INTAG	015073	3830	3920#				
\$LF	013766	3655#	3904	3913			
\$MAIL =	***** U	424	3600				
\$MNEW	015061	3805	3917#				
\$MSWR	015050	3802	3915#				
\$NULL	013760	3618	3649#				
\$OCNT	014212	3688*	3717*	3730#			
\$OMODE	014214	3683*	3687*	3692	3695*	3706*	3732#
\$POWER	015332	3210	4000	4007#			
\$PWAD	015320	4002#					
\$PWADN	015160	219	3969#	3997			
\$PWARMG	015314	4000#					
\$PWUP	015232	3979	3985#				
\$QUES	013764	3653#	3848	3897	3913		
\$RDCHR	014600	3861#	3960				
\$RDDEC =	***** U	3962					
\$RDLIN	014720	3889#	3961				
\$RDOCT =	***** U	3962					
\$RDSZ =	000010	3882#					
\$RESRE	014254	3766#	3963				
\$R2A =	***** U	3964					
\$SAVRE	014216	3750#	3962				
\$SAVR6	015330	3978*	3986	3987*	3988*	4006#	
\$SB2D	015342	4021#	4117				
\$SETUP =	000114	241#	421	3785	3919		
\$STUP =	177777	241#					
\$SWR =	160000	18	19#	4003			
\$TKB	014314	3782#	3794	3811	3865	3871	

CZRxBF0 R11 INTERFACE TEST
CZRxBF.P11 08-MAY-79 14:22

MACY11 30A(1052) 29-MAY-79 08:23 PAGE 112
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0138

.SDIV	1#		
.SEOP	1#		
.SERRO	1#		
.SERRT	1#		
.SMULT	1#		
.SPOWE	1#	5#	3965
.SRAND	1#		
.SRDDE	1#		
.SRDOC	1#		
.SREAD	1#	5#	3778
.SR2AZ	1#		
.SSAVE	1#	5#	3733
.SSB2D	1#	5#	4010
.SSB2O	1#		
.SSCOP	1#		
.SSIZE	1#		
.SSUPR	1#		
.STRAP	1#	5#	3921
.STYPB	1#		
.STYPD	1#		
.STYPE	1#	5#	3577
.STYPO	1#	5#	3656
.S4OCA	1#		
.1170	1#		

.ABS. 017056 000

ERRORS DETECTED: 0

DSKZ: CZRxBF, DSKZ: CZRxBF.SEQ/CRF/SDL-[400,1066]SYSMAC.SML,[400,2465]CZRxBF.P11
RUN-TIME: 36 49 3 SECONDS
RUN-TIME RATIO: 363/90=4.0
CORE USED: 33K (65 PAGES)