Tektronix®
COMMITTED TO EXCELLENCE

PLEASE CHECK FOR CHANGE INFORMATION AT THE REAR OF THIS MANUAL.

067-1070-03
4041 TEST FIXTURE
SUPPORT KIT

USER'S MANUAL

INSTRUCTION MANUAL

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

Serial Number _____

070-4152-02
Product Group 76

First Printing JUL 1983
Revised MAR 1985

INSTRUMENT SERIAL NUMBERS

Each instrument has a serial number on a panel insert, tag, or stamped on the chassis. The first number or letter designates the country of manufacture. The last five digits of the serial number are assigned sequentially and are unique to each instrument. Those manufactured in the United States have six unique digits. The country of manufacture is identified as follows:

B000000	Tektronix, Inc., Beaverton, Oregon, USA
100000	Tektronix Guernsey, Ltd., Channel Islands
200000	Tektronix United Kingdom, Ltd., London
300000	Sony/Tektronix, Japan
700000	Tektronix Holland, NV, Heerenveen, The Netherlands

CONTENTS

ILLUSTRATION

TABLES

Section 1

GENERAL INFORMATION

This manual provides instructions for using the 4041 Test Fixture Support Tape and related test fixture components to test and troubleshoot the 4041 Computer/Controller.

The information contained in this manual is intended to be used by qualified service persons who have a good understanding of 4041 circuit function and who know how to operate the 4041.

The following documents contain information related to using the 4041 Test Fixture Support Tape and test fixture:

- o 4041 Service Manual
- o 4041 Operator's Manual
- o 4041 Programmer's Reference Manual
- o 68000 Personality Module Service Manual

PRODUCT DESCRIPTION

The 067-1070-03 4041 Test Fixture Support Kit consists of the 4041 Test Fixture Support Tape and this User's manual. The 4041 Test Fixture Support Tape is a DC-100 Tape Cartridge containing programs that drive a test fixture consisting of a 4041 Computer/Controller and a 68000 Personality Module. The programs on the tape are written in 4041 Extended BASIC and are designed specifically for testing and troubleshooting the 4041. The Tape Cartridge is inserted into the tape drive of the test fixture's 4041 and the programs are loaded into its RAM. The test fixture's 4041 then exercises the 4041 under test by means of a 68000 Personality Module.

Under control of the 4041 Test Fixture Support Tape, the test fixture can do the following:

- o Initialize the system under test.
- o Read and modify memory and register contents within the system under test.
- o Halt, run, and step the CPU within the system under test.

- Set address and data break points (the system halts when the selected address or data appears on the bus.)
- Perform predefined tests designed to isolate circuit problems to the component level.

EQUIPMENT REQUIRED

A complete test fixture consists of the following equipment:

- A 4041 Computer/Controller with Option 2 (TTL Interface) installed and with one of the Read/Write Memory expansion options: Option 20, 21, 22, or 23.
- A 68000 Personality Module (067-1047-00).
- An RS-232 compatible terminal, such as the TEKTRONIX 4025 Computer Display Terminal.
- Interconnecting cables, as follows:

  1. A cable to connect the controlling 4041 to the 68000 Personality Module. (This cable is supplied as an optional accessory to the 4041 - Part No. 012-0432-02).

  2. Two ribbon cables to connect the 68000 Personality Module to the system under test. (These cables are supplied with the 68000 Personality Module.)

- The 4041 Test Fixture Support Tape.
- This User's Manual.

In addition to the equipment listed above, an extender board (Part No. 067-7054-01) and extender cables (Part No. 175-3388-00) are necessary to gain access to circuitry contained on the circuit boards that plug into the 4041's MIPS board. These items are available as optional accessories to the 4041.

Section 2

INSTALLATION

SETTING UP THE EQUIPMENT

Figures 2-1 and 2-2 show how to connect the test fixture components to the 4041 to be tested. In connecting up these components, proceed as follows:

1. Make sure that the power switches on the 4041s are turned off. Disconnect the power cord of the 4041 to be tested from the AC outlets.

WARNING

When operating the 4041 with its covers removed, you may be at risk of lethal electrical shock. Be especially careful when working near the switching power supply on the MIPS board. Always refer to the 4041 Service Manual for disassembly and troubleshooting procedures, giving due regard to any warnings and cautions therein.

2. Remove the top and right side panels from the 4041 to be tested (refer to the 4041 Service Manual).

3. Referring to Figure 2-1, dismount the Front Panel Control Board, set its top mounting bracket aside, and move the board aside far enough to allow access to the CPU board.

4. Using the two ribbon cables supplied with the Personality Module, connect J801 on the personality module to J41 on the 4041's CPU board. Connect J802 on the personality module to J31 on the CPU board. J31 and J41 are located near the top edge of the CPU board. (See Figure 2-2).

5. Connect one end of the large cable labeled "INTER-CONNECTING CABLE TEK 012-0432-02" to the connector labeled "TTL INTERFACE" on the rear panel of the controlling 4041.

6. Connect the other end of the large cable to J800 on the end of the personality module.

7. Connect the terminal to the RS-232 connector on the rear panel of the controlling 4041. Connect the terminal to the AC power outlet and turn it on.

8. Connect the 4041 to be tested to the AC power outlets -- but leave the power switch off.

9. Proceed to the "Initial Checkout" procedure, which follows.

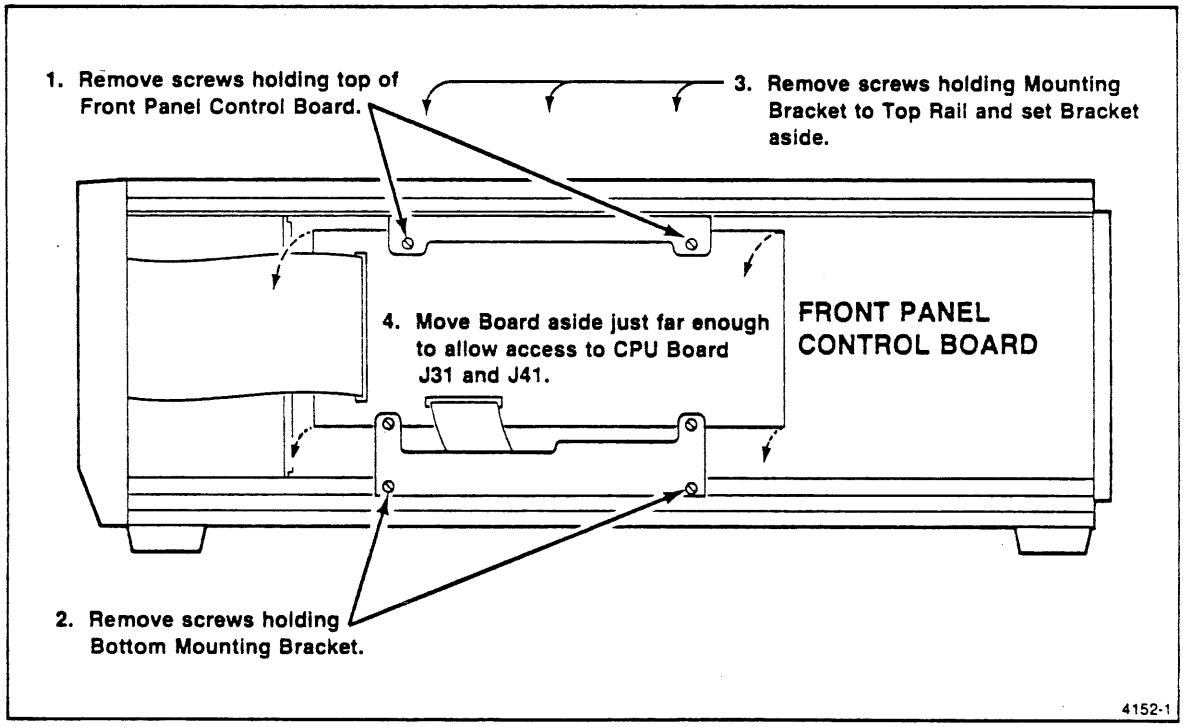

Figure 2-1. Gaining Access to the CPU Board.

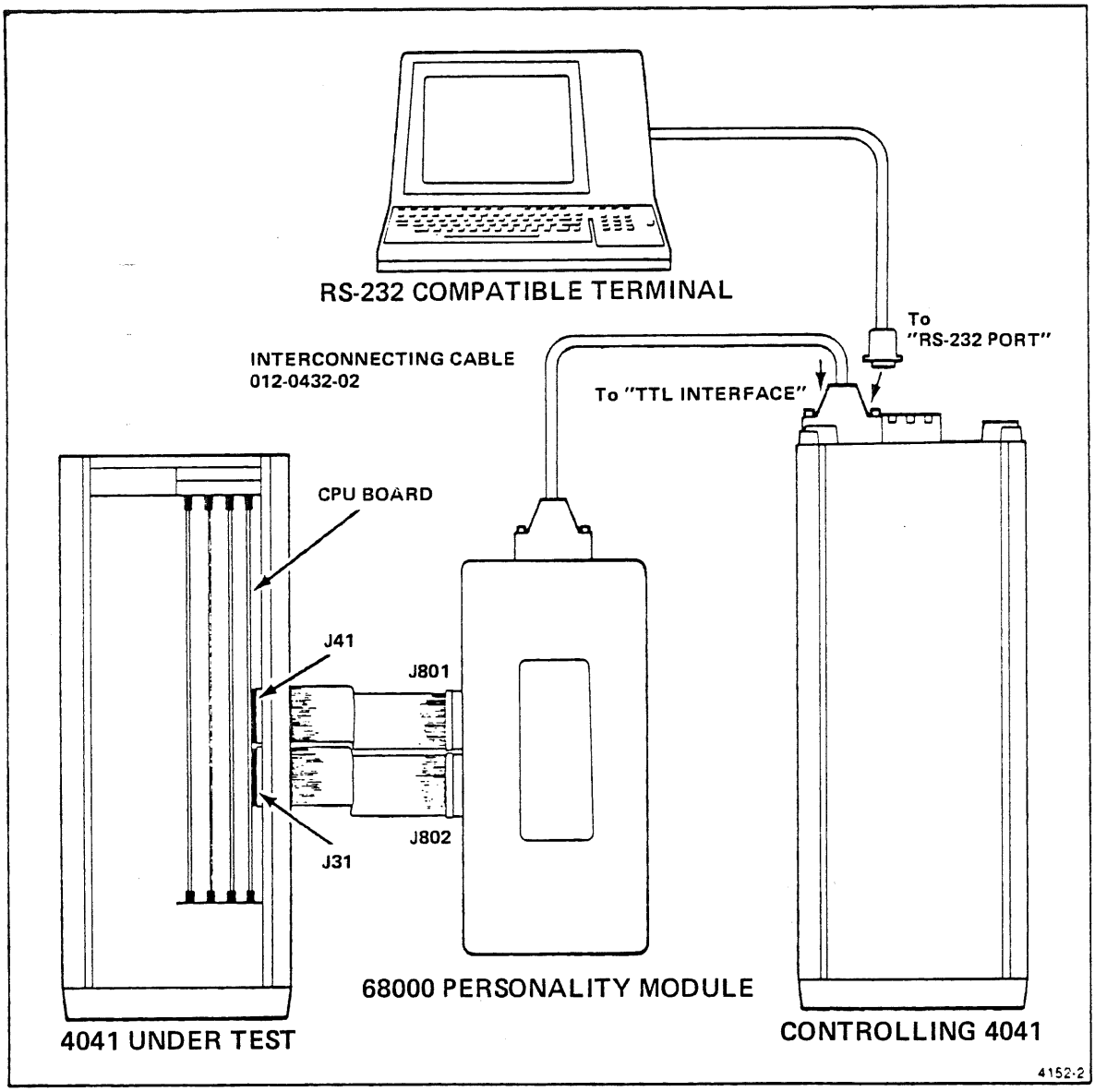

Figure 2-2. Connecting the Test Fixture.

INITIAL CHECKOUT

1. Insert the 4041 Test Fixture Support Tape into the controlling 4041's Tape Drive and turn on power to that 4041 only. At this point, the software on the tape is automatically loaded into RAM. After about 90 seconds, the following message should appear on the terminal's screen:

```
4041 System Test Fixture    067-1070-03
Copyright (c) Tektronix, Inc.  1984
Version 2.3

Initializing
>>>_
```

The triple right angle bracket ">>>" is the prompt character used by the test fixture software. When this character appears, the system is ready for input from the keyboard.

2. Enter the characters **HELP** immediately following the prompt, then press <CR>. A list of commands is displayed, as follows:

```
------------------------------------------
          LEGAL COMMANDS
          ----- --------

GO
HALT
STEP [number_of_steps]
DM [address [number_of_bytes]]
OPEN address
BA address      Break on Address compare
BD data         Break on Data compare
REG             Dump all 68000 registers
INIT            Restart system under test
EXIT            Leave test fixture program
HELP            This message
nnn             Execute test number 'nnn'
------------------------------------------
```

This list contains all of the "high-level" commands provided by the test fixture software.

3. If everything appears to be working properly so far, turn on the system under test, type in INIT and press <CR>. The message **SELF TEST** should appear in the front panel display of the 4041 under test.

4. Enter **GO <CR>** on the terminal; this causes the 4041 under test to execute its self-test program (assuming that the 4041 is working well enough to run.) Proceed to the "COMMANDS" and "FUNCTION TESTS" sections for detailed information on the test fixture's capabilities.

NOTE

If any question arises about whether the test fixture is operating properly, check its operation by testing a 4041 that is known to be working properly.

Section 3

COMMANDS

This section contains descriptions of the high level commands provided by the 4041 Test Fixture Support Tape. The descriptions take the following form:

- o Command Name.
- o Brief Description.
- o Syntax form of the command, including the mnemonic and an explanation of any parameters, enclosed in a box.
- o Examples and further explanation.

Whenever the prompt character >>> appears, it indicates that the test fixture is ready to receive commands. Enter the command following the prompt character (It is alright to have spaces between the prompt character and the command mnemonic). The test fixture executes the command upon receiving <CR>. (Any time <CR> appears, it means "press RETURN"). Command mnemonics must be separated from their parameters by at least one space; the first parameter must be separated from the second by at least one space. Parameters are entered in hexadecimal. If you enter more than one command before <CR>, only the first one is executed. Items enclosed in brackets ([]) are optional; they may be entered or left out, at your discretion (or as noted).

BREAK AT ADDRESS

The BREAK AT ADDRESS command causes the system under test (SUT) to halt when a specified address is reached during program execution.

```
BA [address]

address:     location where the SUT halts; defaults to 0.
```

The SUT starts running at the location currently in its program counter and continues until the break address is reached.

Example:

BA FD0262 stops the SUT at X 'FD0262'.

BREAK ON DATA

The BREAK ON DATA command causes the system under test (SUT) to halt when a specified data word appears on the data bus.

BD [data]	
data:	bus data which causes the SUT to halt; defaults to 0.

The SUT starts running at the location currently in its program counter and continues until the specified data value is transferred on the bus.

Example:

BD 40 causes the SUT to run, then halts it when X'0040' appears on the data bus.

DISPLAY MEMORY

The DISPLAY MEMORY command allows memory content within the system under test to be displayed.

DM [start [number]]	
start:	first address whose content is displayed; defaults to X'000000'.
number:	number of locations whose content is displayed; defaults to 16 (X'10'); the minimum increment is 16 bytes.

Examples:

DM displays the contents of 16 bytes beginning at location X'000000'.

DM 7000 20 displays the contents of 32 (X'20) bytes beginning at location X'007000'.

Sixteen bytes are displayed on each line, as follows: First, the beginning address of the line is shown. Second, the data are shown in hexadecimal numerals. Third, the data are represented in their ASCII equivalents (those values outside the range of printable characters are represented by a period). Shown below is a typical display produced by the DISPLAY MEMORY command.

```
dm fc0716 40
FC0716 2A 08 02 85 00 FF 80 00 55 85 24 45 4E 73 41 FA *.......U.$ENsA.
FC0726 00 2E 45 F9 00 00 00 64 24 C8 24 C8 24 C8 24 C8 ..E....d$.$.$.$.
FC0736 24 C8 42 80 3E 3C FF FF 41 FA 00 16 21 C8 00 80 $.B.><..A...!...
FC0746 4E 40 4E 75 53 86 66 FC 46 D7 0F C0 60 02 5C 8F N@NuS.f.F...`.\.

>>>_
```

EXIT

The EXIT command causes the controlling 4041 to exit from the test fixture program and return to its system monitor.

```
EXIT
```

GO

The GO command causes the system under test to begin executing instructions from the current program counter location.

```
GO
```

HALT

The HALT command causes the system under test to stop program execution.

```
HA[LT]
```

INIT

The INIT command forces a restart operation in the system under test.

```
IN[IT]
```

OPEN

The OPEN command allows memory and register content within the system under test (SUT) to be examined and changed.

```
OP[EN] address

address:     location to be opened; must be specified.
```

NOTE

If access to an undefined or unresponsive memory location is attempted, the test fixture places an asterisk between the address field and the data field displayed on the terminal.

When the OPEN command is given, the test fixture displays the address and the content of that address. At this point, the user has the following options:

1. Entering a space, followed by <CR>, causes the same location to be read again.
2. Entering <CR> causes the next location to be read.
3. Entering <CR> causes the previous location to be read.
4. Entering data, followed by <CR>, causes the data to be written at the specified location. The location is then read again to display the results of the write operation.
5. Entering .<CR> causes the test fixture to exit from the OPEN command and return the prompt character (>>>).

REGISTERS

The REGISTERS command causes the contents of the 68000 microprocessor's registers (A0-A6 and D0-D7) to be displayed.

```
RE[GISTERS]
```

A typical REGISTERS display is shown below. Register content is discussed in Section 4 under "SELF TEST ERROR CODING".

```
>>>re
D0: 00000000
D1: 00002042
D2: 00002042
D3: 312E3120
D4: 00000B98
D5: 00000AB4
D6: 00000B10
D7: 00000A48
A0: 000020E8
A1: 00003768
A2: 00027500
A3: 00080001
A4: 00000000
A5: 000018B0
A6: 0000160A
A7: 0000177E
```

STEP

The STEP command causes the system under test (SUT) to execute a specified number of bus cycles.

ST[EP] [number]
number: number of bus cycles to be executed; defaults to 1; maximum value is 7FFFF.

Examples:

ST 100 steps the SUT through 256 steps.

ST F steps the SUT through 15 steps.

Section 4

FUNCTION TESTS

Software on the 4041 Test Fixture Support Tape provides a number of predefined tests that permit component-level testing of the 4041's circuitry. The tests are divided into four classes according to the level of testing provided. Each test is assigned a number: Classes 1, 2, 3, and 4 are assigned numbers 100 through 199, 200 through 299, 300 through 399, and 400 through 499, respectively. To initiate a particular test, simply enter its number after the prompt character and press <CR>.

Class 1 Tests

Class 1 tests exercise the data bus and address bus circuitry and the address decoding logic. The test fixture takes control of the bus and places a value, or series of values, on the bus. The user can then check to see that the correct logic levels appear in the bus and address decoder circuitry. To abort any of these tests, press CONTROL-C. Table 4-1 summarizes the Class 1 tests.

Table 4-1

CLASS 1 TESTS

Test	Function Tested	Description
100 (2)	system address bus	Each test walks a bit up its respective bus, from the low-order to the high-order system data bus bit position.
102 (2)	system data bus	
110 (1)	U480-12	These tests check address decoding on the Standard I/O board. In each case, a series of addresses is placed on the bus; the specified output pin should toggle with a duty cycle of about 50%.
111 (1)	U480-6U	
112 (1)	U480-8	
113 (1)	U276-12	
114 (1)	U276-6	
115 (1)	U276-8	
116 (1)	U360-6	
130 (1)	U141-5	These tests check address decoding on the CPU board. In each case, the specified output should toggle with a duty cycle of about 50%.
131 (1)	U141-6	
132 (1)	U241-6	
133 (1)	U235-8	
134 (1)	U241-5	
135 (1)	U131-8	
136 (1)	U125-8	
137 (1)	U135-6	

(1) Type 1 test -- See "Adding Tests" at the end of this section.
(2) Type 7 test.

Table 4-1 (cont)

CLASS 1 TESTS

Test	Function Tested	Description
138 (1)	U121-8	These tests check address decoding on the CPU board. In each case, the specified output should toggle with a duty cycle of about 50%.
139 (1)	U125-6	
140 (1)	U221-3	
141 (1)	U131-6	
142 (1)	U235-6	
143 (1)	U225-8	
144 (1)	U221-8	
145 (1)	U221-11	
146 (1)	U121-11	
150 (1)	U285-6	These tests apply to the R/W Memory board. In each case, the specified output should toggle with a duty cycle of about 50%.
151 (1)	U281-1	
152 (1)	U281-2	
153 (1)	U281-3	

(1) Type 1 test -- See "Adding Tests" at the end of this section.
(2) Type 7 test.

Class 2 Tests

Class 2 tests are designed to check the next higher level of circuitry above that checked by Class 1 Tests. By repeatedly reading or writing specific locations, Class 2 Tests check the various sequencers and timing generators involved in bus operations. To abort a Class 2 Test, press CONTROL-C. Table 4-2 summarizes the Class 2 Tests available.

Table 4-2

CLASS 2 TESTS

Test	Function Checked	Description
200 (1)	8250 ACE (read R0)	These tests apply to the Standard I/O board. A register in the specified circuit is repeatedly read or written, providing a "scope loop" for testing the circuitry involved.
201 (1)	8250 ACE (write R0)	
202 (1)	9914 GPIB (read R0)	
203 (2)	9914 GPIB (write R0)	
204 (1)	6821 PIA (read R0)	
205 (2)	6821 PIA (write R0)	
206 (1)	6840 PTM (read R0)	
207 (2)	6840 PTM (write R0)	
208 (1)	6850 ACIA (read R0)	
209 (2)	6850 ACIA (write R0)	
210 (1)	Tape Drive (read R0)	
211 (2)	Tape Drive (write)	
214 (1)	System RAM (read)	Address X'000100' is repeatedly read or written.
215 (2)	System RAM (write)	
220 (1)	8250 ACE (read R0)	These tests apply to the Optional I/O board. A register is repeatedly read or written, providing a "scope loop" for testing the circuits involved.
221 (2)	8250 ACE (write R0)	
222 (1)	9914 GPIB (read R0)	
223 (2)	9914 GPIB (write R0)	

(1) Type 2 Test -- see "Adding Tests" at the end of this section.
(2) Type 3 Test.

Table 4-2 (cont)

CLASS 2 TESTS

Test	Function Checked	Description
224 (1)	6821 PIA (read R0)	These tests apply to the Optional I/O board. A register is repeatedly read or written, providing a "scope loop" for testing the circuits involved.
225 (2)	6821 PIA (write R0)	
226 (1)	DMA Status Reg. (read)	
227 (2)	DMA Compare Lat. (wrt)	
228 (1)	DMA Commnd Reg. (read)	
229 (2)	DMA Commnd Reg. (wrt)	
230 (1)	2940 high (read R0)	
231 (2)	2940 high (write R0)	
232 (1)	2940 middle (read R0)	
233 (2)	2940 middle (wrt R0)	
234 (1)	2940 low (read R0)	
235 (1)	2940 low (write R0)	
236 (1)	ROM (read X'F00000')	
240 (1)	System ROM	Read X'FD0008' CPU board
242 (2)	Self-Test LEDs	Write X'F68000' CPU board
250 (1)	Optional RAM (read)	Read/Write Memory board location X'008000'
251 (2)	Optional RAM (write)	
260 (1)	Option 3 ROM	Read X'D80000' on SCSI
261 (1)	Opt 3 DMA Controller (read)	These tests apply to the Opt 3 SCSI board
262 (2)	Opt 3 DMA Controller (write)	
263 (1)	Opt 3 SCSI Controller (read)	

Table 4-2 (cont)

CLASS 2 TESTS

Test	Function Checked	Description
264 (2)	Opt 3 SCSI Controller (write)	A register is repeatedly read or written, providing a "scope loop" for testing the circuits involved
265 (1)	Opt 3 RS232 Controller (read)	
266 (2)	Opt 3 RS232 Controller (write)	
267 (1)	Opt 3 Interrupt Control (read)	
268 (2)	Opt 3 Interrupt Control (write)	
269 (1)	Read U22	
270 (2)	Write U12	
271 (1)	Read U15	

(1) Type 2 test -- see "Adding Tests" at the end of this section.
(2) Type 3 test.

Class 3 Tests

Class 3 tests are functional tests of circuit blocks on the Standard I/O, Optional I/O, and CPU boards. Table 4-3 summarizes the Class 3 tests available.

Table 4-3

CLASS 3 TESTS

Test	Function Tested	Description
300 (2)	RS-232 Port, Standard I/O bd.	Write, read, and verify values of 00,FF,55, and AA in the Line Control Register of the 8250 ACE.
301 (3)	RS-232 Port, Standard I/O bd.	Execute 4041 self-test code for this circuit block.
302 (3)	GPIB Port, Standard I/O bd.	Error conditions are flagged using one or more of the 68000's registers, A0 thru A7 and D0 thru D7; the registers' contents are displayed at the end of the test. (1)
303 (3)	6840 Timer, Standard I/O bd.	
304(3)	Front Panel Comm. Port, Standard I/O bd.	
310 (2)	RS-232 Port, Optional I/O bd.	Same as test 300
311 (4)	RS-232 Port, Optional I/O bd.	Execute 4041 self-test code.
312 (4)	DMA Command Register, Optional I/O bd.	Error conditions are reported as in tests 301 thru 304. (1)

(1) To interpret the registers' contents, refer to "SELF TEST ERROR CODING" at the end of this section.
(2) Type 4 test -- see "Adding Tests" at the end of this section.
(3) Type 5 test.
(4) Type 7 test.

Table 4-3 (cont)

CLASS 3 TESTS

Test	Function Tested	Description
313 (2)	DMA Command Register, Optional I/O bd.	Write, read, and verify four values: 0, 12, 13, 8.
320 (4)	System ROM, CPU board	18 locations are read and the data pattern is verified.
330 (4)	Self-Test LEDs, CPU board	The following sequence is displayed in the LEDs: 000, 100, 000, 010, 000, 001, 000, 111, 011, 111, 101, 111, 110, 111, 000, read front to back.
340 (2)	Opt 3 5385 SCSI Controller	Error Conditions are reported as in tests 301 thru 304. (1)
341 (2)	Opt 3 9516 DMA Controller	
342 (2)	Opt 3 8250 RS232 Controller	
343 (3)	Opt 3 SCSI Self Test	Performs the SCSI part of the power up self tests. (1)
344 (3)	Opt 3 RS232 Self Test	Performs the RS232 part of the power up self tests. (1)

(1) To interpret the registers' contents, refer to "SELF TEST ERROR CODING" at the end of this section.
(2) Type 4 test -- see "Adding Tests" at the end of this section.
(3) Type 5 test.
(4) Type 7 test.

Class 4 Tests

Class 4 tests are board-level functional tests. In some cases, they are sequential combinations of Class 3 tests; in other cases, they invoke portions of the 4041's self-test code. Table 4-4 summarizes the Class 4 tests available.

Table 4-4

CLASS 4 TESTS

Test	Function Tested	Description
401 (2)	Standard I/O board	Executes tests 300 - 304 (1)
402 (2)	Optional I/O board	Executes tests 310 - 313 (1)
403 (3)	Front Panel LEDs and Printer	LED message: THIS IS A TEST****** Printer message: PRINTER TEST 1234567
404 (3)	All ROM	These tests execute self-test code. (1)
405 (4)	RAM, Standard I/O bd.	
406 (4)	RAM, Optional I/O bd.	
407 (4)	Bottom 16K of RAM	
408 (4)	2nd 16K of RAM	
410 (2)	Opt 3 tests	Executes 340 thru 344 (1)

(1) Errors are reported using the 68000's internal registers, whose contents are displayed at the end of each test. To interpret the registers' content, refer to "SELF TEST ERROR CODING" at the end of this section.
(2) Type 6 test -- see "Adding Tests" at the end of this section.
(3) Type 7 test.
(4) Type 5 test.

ADDING TESTS

Those users familiar with 4041 BASIC programming can add new tests to the 4041 Test Fixture Support Tape program. The simplest method of adding a test is to insert a DATA statement into the program on the tape; this takes advantage of utility subprograms and other functions already on the tape for performing low-level I/O and control functions. A listing of the tape program is provided in Appendix A for reference.

In general, the DATA statement contains a series of data values that define the test, as follows:

[line #] DATA nnn,t,value 1[,value 2,...,value n][,terminator]

where nnn is the test number; t is the test type (there are seven types, 1 through 7); values 1 through n may be data, addresses, or other test numbers; and the terminator is a fixed value (specific to the type of test) that delimits the list of values.

As in the previously described Function Tests, these user-defined tests are invoked by typing in the test number immediately after the prompt character. The seven types of tests that may be added by using DATA statements are decribed in the following paragraphs.

Type 1 Tests

Type 1 tests place a series of addresses on the address bus without allowing any data transfer to occur. These tests provide a means of checking the bus drivers and the address decoding logic. For example,

[line #] DATA 999,1,0,2,4,6,-1

defines a Type 1 test, numbered 999, which sequentially and repeatedly places the values 0,2,4, and 6 on the address bus. The value -1 is the terminator.

Type 2 Tests

Type 2 tests cause a specified address to be read repeatedly. This provides a "scope loop" for checking bus timing and sequencing. For example,

[line #] DATA 998,2,[F4043Dh],0

defines a Type 2 test, numbered 998, which repeatedly reads address X'F4043D' (The small h after the address indicates that it is a hexadecimal number). The value 0 is the terminator.

Type 3 Tests

Type 3 tests cause a specified data byte to be repeatedly written at a specified address. This provides a "scope loop" for testing bus timing and sequencing for write operations. For example,

[line #] DATA 997,3,[F4043Dh],[55h]

defines a Type 3 test, numbered 997, which repeatedly writes X'55' at location X'F4043D'.

Type 4 Tests

Type 4 tests write a value, or a series of values, at a specified location. After each write operation, the same location is read back to verify that the value written there is correct. For example,

[line #] DATA 996,4,[F20202h],0,1,2,-1

defines a Type 4 test where X'00' is written at location X'F20202' then read back and verified. The same is done for 01 and 02. The value -1 serves as the terminator. If any of the values come back incorrect, an error message is displayed and the fixture's prompt character returns, as shown below.

```
Data Written F0 Data Read FF
>>>
```

Type 5 Tests

Type 5 tests provide a mechanism for executing parts of the self-test code residing within the system under test (SUT). Several things occur when a Type 5 is invoked.

1. The SUT is halted and an interrupt vector is forced onto its address bus; this vector forces the CPU to a small monitor program residing in the SUT.

2. A breakpoint is set in the SUT.

3. The registers in the SUT are saved.

4. A series of values is loaded into the SUT which set it up to do the test; this includes an address to begin running.

5. The SUT is allowed to run to the breakpoint.

The data statement for a Type 5 test consists of a test number followed by a series of 30 values. The series of values is loaded into registers D0 through D7 and A0 through A6 within the SUT (in the order given -- high byte first, then low byte) the value loaded into A6 is the address where the SUT begins running; the values in the other registers are parameters which set up the particular portion of the SUT to be tested (the COMM Port, for example).

Most of the Class 3 Tests defined under "FUNCTION TESTS" are Type 5 Tests. Refer to the test fixture software listing in Appendix A for examples of DATA Statements for Type 5 Tests. To understand the purposes of the data values loaded into the various registers, refer to the self-test code listing in Appendix B; see, for example, the CHPTST routine.

Type 6 Tests

Type 6 tests provide a means of executing a series of previously defined numbered tests. For example,

[line #] DATA 995,6,300,301,302,-1

defines test number 995, a type 6 test which executes tests numbered 300, 301, and 302. The value -1 is the terminator.

Type 7 Tests

Type 7 tests provide a means of adding functions not provided by the other test types. The DATA statement for a Type 7 test is structured as follows:

[line #] DATA nnn,7,[index],

where nnn is the test number, 7 is the test type, and the index is the number that directs the test fixture program execution to a user-written subprogram.

Implementing a Type 7 Test requires more than just a DATA Statement. Within the BASIC program, at the line labeled DT7, there is a GO TO Statement containing a series of pointers, or labels. Each label is associated with the beginning line of specific subprogram block.

When the DATA Statement for a new Type 7 test is added, an additional printer must be added at DT7 as well. This new pointer has to be [index] positioned to the right in the series of labels. For example,

[line #] DATA 994,7,9

defines a type 7 test, number 994, which is indexed to position 9 in the GO TO Statement at DT7. If the label DT79, for example, were added at position 9 in the GO TO Statement, program execution would jump to the statement labeled DT79.

SELF-TEST ERROR CODING

Error conditions encountered during execution of the 4041's self-test firmware are reported by way of the 68000's internal registers. Ordinarily, when one of the numbered Function Tests which makes use of the self-test code is executed, the status of the registers is displayed at the end of the test; this allows the error status to be determined. In almost all cases, errors are flagged using D0 only. Information on the contents of the registers can be found in the listing in Appendix B. Tables 4-5 through 4-11 relate the register contents to the error conditions.

Table 4-5

ROM TEST ERROR CODES

Register	Content	Description
D0	0	no error
	1	EXOR test failed
	2	checksum error

Table 4-6

RAM TEST ERROR CODES

Register	Bit	Description (1)
D0	0	walking bit error
	1	pattern error
	2	counter error
	3	refresh of counter error
	4	memory complement error
	5	refresh of memory complement error
	6	memory clear error
	31	parity error
A0	0-23	address where error occurred

(1) logic 1 = error, logic 0 = no error.

Table 4-7

TIMER TEST ERROR CODING

Register	Bit	Description (1)
D0	0	no Timer 1 interrupt @ 2ms
	1	bad interrupt status
	2	no Timer 1 interrupt @ 4ms
	3	bad interrupt status
	4	no Timer 3 interrupt @ 5ms
	5	bad interrupt status
	6	no Timer 2 interrupt @ 10ms
	7	bad interrupt status

(1) logic 1 = error, logic 0 = no error.

Table 4-8

TAPE TEST ERROR CODING

Register	Bit	Description (1)
D0	0	no tape interrupt
	1	bad status
	2	bad status after servo fail flag acknowledge
	3	bad status after servo fail flag reset

(1) logic 1 = error, logic 0 = no error

Table 4-9
(RS232C)
Standard Comm and Opt.1 Error Coding

Register	Bit	Description (1)
D0	0	no XBE interrupt
	1	invalid IIR contents
	2	no XBE interrupt after xmit
	3	invalid IIR contents
	4	no RBF interrupt
	5	invalid IIR contents
	6	bad data received, byte 1
	7	no XBE interrupt, byte 2
	8	invalid IIR contents
	9	no RBF interrupt
	10	bad IIR contents
	11	bad data received, byte 2

(1) logic 1 = error, logic 0 = no error

Table 4-10

FRONT PANEL CONTROLLER TEST ERROR CODING

Register	Bit	Description (1)
D0	0	no TX interrupt
	1	bad status register
	2	no RX interrupt
	3	bad status register
	4	Front Panel detected failure

(1) logic 1 = error, logic 0 = no error

Table 4-11

GPIB TEST ERROR CODING

Register	Bit	Description (1)
D0	0	no IFC interrupt
	1	bad interrupt status
	2	listener handshake error
	3	no MAC, MA interrupt
	4	bad interrupt status
	5	listener handshake error
	6	listener handshake error
	7	no B0 interrupt
	8	bad interrupt status
	9	talker handshake error
	10	bad data detected
	11	talker handshake error
	12	no B0 interrupt
	13	no B0 interrupt
	14	bad interrupt status
	15	CONT not asserted
	16	bad data detected
	17	TE not asserted
	18	bad state: ATN, IFC
	19	bad state: ATN

(1) logic 1 = error, logic 0 = no error

Table 4-12

Option 1 GPIB Test
Error Coding

Register	Bit	Description
D0	(0-19)	(Same description as table 4-11)
	20	bad DMA control register data
	21	bad Address/Word count data
	22	bad data, DMA read from memory
	23	no DMA done interrupt or bad status
	24	bad Byte Match Test
	25	no DMA done interrupt or bad status write to memory test
	26	bad data, DMA write to memory
	28	bad data, LSB 2940 read back
	29	bad data, LSB 2940 read back
	30	bad data, MSB 2940 read back
	31	no hardware available

Table 4-13

Disk Test Error Coding

Bit Number	Description
0	Address decoder failure (On = failure)
1	9519 Interrupt controller status failure (On = failure)
2	9516 DMA pointer register failure (On = failure)
3	9516 DMA data register failure (On = failure)
4 - 7	Always zero
8 - 13	5385 SCSI Diagnostic information (see chart below)
14	Always zero
15	5385 SCSI Diagnostic complete bit (On = failure)

File: tss2 01-NOV-84 08:28:04

```
  100   Rem
  110   Rem              TEST TABLES FOR THE 4041 COMPUTER/CONTROLLER
  120   Rem
  130 Bdata:   rem              STRUCTURE OF DATA STATEMENTS
  140   Rem
  150   Rem                  <DATA  XX,YY,DD,DD,DD,DD>
  160   Rem
  170   Rem                  WHERE XX=TEST NUMBER
  180   Rem                        YY=TEST TYPE
  190   Rem                        DD=DATA FOR THE TEST
  200   Rem
  210   Rem
  220     Data 401,6,300,301,302,303,304,-1 !SIO Group Test
  230     Data 402,6,310,313,-1 !OIO Group Test
  240     Data 403,7,1 !Front Panel/Display Test
  250     Data 404,7,2 ! Test all ROMs
  260   !
  270   ! Test all of Standard RAM
  280   !
  290     Data 405,5,0,0,[5555h],[5555h],[AAAAh],[AAAAh],0,0,0,[FFFFh],0,0
  300     Data 0,0,[2h],[49F0h],0,0,0,[700h],0,[7FFEh],0,[700h],0,0,-5,-6,-3,-4,0,[7
  **           FF0h]
  310   !
  320   ! Test all of Optional RAM
  330   !
  340     Data 406,5,0,0,[5555h],[5555h],[AAAAh],[AAAAh],0,0,0,[FFFFh],0,0
  350     Data 0,0,[2h],[49F0h],0,0,0,[8000h],[10h],[0h],0,[8000h],0,0,-5,-6,-3,-4,0
  **           ,[7FF0h]
  360   !
  370   ! Test addresses  x'000008 through x'004000
  380   !
  390     Data 407,5,0,0,[5555h],[5555h],[AAAAh],[AAAAh],0,0,0,[FFFFh],0,0,0,0
  400     Data 2,[49F0h],0,0,0,8,0,[4000h],0,[100h],0,0,-5,-6,-3,-4,0,[7FF0h]
  410   !
  420   ! Test addresses x'004000 through x'007fff
  430   !
  440     Data 408,5,0,0,[5555h],[5555h],[AAAAh],[AAAAh],0,0,0,[FFFFh],0,0,0,0
  450     Data 2,[29F0h],0,0,0,[4000h],0,[7FFEh],0,[6000h],0,0,-5,-6,-3,-4,0,[3FF0h]
  460   !
  470     Data 410,6,340,341,342,343,344,-1 ! Opt 3 Group tests
  480   !
  490   !      300 Level Tests.
  500   !
  510     Data 300,4,[F40E07h],0,[FFh],[55h],[AAh],-1 !Std. RS-232  ACE chip
  520     Data 301,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,[F4h],0,0,0,0,0,-13,-14,-5,
  **           -6,-9,-10,0,[7FF0h] !Std. Comm Self Test
  530     Data 302,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,[F4h],0,0,0,0,0,-17,-18,-5,
  **           -6,-9,-10,0,[7FF0h] !Std. GPIB Self Test
  540     Data 303,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,[F4h],0,0,0,0,0,-15,-16,-5,
  **           -6,-9,-10,0,[7FF0h] !Timer Self Test
  550     Data 304,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,[F4h],0,0,0,0,0,-19,-20,-5,
  **           -6,-9,-10,0,[7FF0h] !FPC Self Test
  560     Data 310,4,[F202F7h],0,[55h],[AAh],[FFh],-1 !Option 1 ACE chip
  570     Data 311,7,3 !Option 1 Comm Self Test
```

PROGRAM LISTING

File: tss2 01-NOV-84 08:29:43

```
 580     Data 312,7,4 !Option 1 GPIB Self Test
 590     Data 313,4,[F202A1h],[F0h],[FCh],[FDh],[F8h],-1 !DMA Command Register
 600     Data 320,7,5 !CPU ROM Pattern Test
 610     Data 330,7,6 !CPU Self Test LEDs
 620     Data 340,4,[D88021h],0,[FFh],[55h],[AAh],-1 ! Opt 3 5385
 630     Data 341,4,[D88063h],0,[7Eh],[54h],[6Ah],-1 ! Opt 3 9516
 640     Data 342,4,[D88087h],0,[FFh],[55h],[AAh],-1 ! Opt 3 8250
 650     Data 343,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,[7000h],0,0,0,0,0,0,0,0,0,[
 **              D8h],[30h],0,[7FF0h] ! Opt 3 SCSI selftest
 660     Data 344,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,[D8h],[
 **              34h],0,[7FF0h] ! Opt 3 RS232C selftest
 670    !
 680    !       200 Level Tests.
 690    !
 700    !       Tests 200-219 are for Standard I/O
 710    !
 720     Data 200,2,[F40E01h],0 !Read ACE chip R0
 730     Data 201,3,[F40E01h],[55h] !Write ACE chip R0
 740     Data 202,2,[F40A01h],0 !Read 9914, R0
 750     Data 203,3,[F40A01h],[55h] !Write 9914, R0
 760     Data 204,2,[F40001h],0 !Read 6821, R0
 770     Data 205,3,[F40001h],[55h] !Write 6821, R0
 780     Data 206,2,[F40C01h],0 !Read 6840, R0
 790     Data 207,3,[F40C01h],[55h] !Write 6840, R0
 800     Data 208,2,[F40801h],0 !Read 6850, R0
 810     Data 209,3,[F40801h],[55h] !Write 6850, R0
 820     Data 210,2,[F40601h],0 !Read 223B, R0
 830     Data 211,3,[F40601h],[55h] !Write 223b, R0
 840     Data 214,2,[100h],0 !Read Standard RAM, Address x'100
 850     Data 215,3,[100h],[55h] !Write Standard RAM, Address x'100
 860    !
 870    !        Tests 220-239 are for Option 1
 880    !
 890     Data 220,2,[F202F1h],0 !Read 8250, R0
 900     Data 221,3,[F202F1h],[55h] !Write 8250, R0
 910     Data 222,2,[F202D1h],0 !Read 9914, R0
 920     Data 223,3,[F202D1h],[55h] !Write 9914, R0
 930     Data 224,2,[F20281h],0 !Read 6821, R0
 940     Data 225,3,[F20281h],[55h] !Write 6821, R0
 950     Data 226,2,[F202B1h],0 !Read DMA Status Register
 960     Data 227,3,[F20291h],[55h] !Write DMA Compare Latch
 970     Data 228,2,[F202A1h],0 !Read DMA command register
 980     Data 229,3,[F202A1h],[5h] !Write DMA command register
 990     Data 230,2,[F20231h],0 !Read High Byte 2940
1000     Data 231,3,[F20231h],[55h] !Write High Byte 2940
1010     Data 232,2,[F20221h],0 !Read Middle Byte 2940
1020     Data 233,3,[F20221h],[55h] !Write Middle Byte 2940
1030     Data 234,2,[F20211h],0 !Read Low Byte 2940
1040     Data 235,3,[F20211h],[55h] !Write Low Byte 2940
1050     Data 236,2,[F00000h],0 !Read ROM Address x'F00000
1060    !
1070    !         Tests 240-249 are for CPU
1080    !
1090     Data 240,2,[FD0008h],0 !Read CPU ROM address x'FD0008
```

File: tss2 01-NOV-84 08:31:30

```
1100     Data 242,3,[F68000h],0 !Write Self Test Display
1110   !
1120   !       Tests 250-259 are for Optional RAM
1130   !
1140     Data 250,2,[8000h],0 !Read Optional RAM address x'8000
1150     Data 251,3,[8000h],[55h] !Write Optional RAM address x'8000
1160   !
1170   !       Tests 260-271 are for Option 3
1180   !
1190     Data 260,2,[D80000h],0 !Read Option 3 ROM
1200     Data 261,2,[D88060h],0 !Read 9516 (DMA)
1210     Data 262,3,[D88060h],[55h] !Write 9516 (DMA)
1220     Data 263,2,[D88021h],0 !Read 5385 (SCSI)
1230     Data 264,3,[D88039h],[55h] !Write 5385 (SCSI)
1240     Data 265,2,[D88081h],0 !Read 8250A (RS232C)
1250     Data 266,3,[D88081h],[55h] !Write 8250A (RS232C)
1260     Data 267,2,[D880A1h],0 !Read 9519A (IRQ)
1270     Data 268,3,[D880A1h],[55h] !Write 9519A (IRQ)
1280     Data 269,2,[D880E1h],0 !Read U22
1290     Data 270,3,[D880E1h],1 !Write U12
1300     Data 271,2,[D880E9h],0 !Read U15
1310   !
1320   !       100 Level Tests
1330   !
1340     Data 100,7,7 !Walk a bit up the address bus
1350     Data 102,7,8 !Walk a bit up the data bus
1360   !
1370   !       Tests 110-129 are for the Standard I/O
1380   !
1390     Data 110,1,0,[800000h],0,[400000h],0,[200000h],-1 !U480-12
1400     Data 111,1,0,[100000h],0,[80000h],0,[40000h],-1 !U480-6
1410     Data 112,1,0,[20000h],0,[10000h],0,[8000h],-1 !U480-8
1420     Data 113,1,0,[4000h],0,[2000h],0,[1000h],-1 !U276-12
1430     Data 114,1,0,[800h],0,[400h],0,[200h],-1 !U276-6
1440     Data 115,1,0,[100h],0,[80h],0,[40h],-1 !U276-8
1450     Data 116,1,0,[20h],0,[10h],0,8,-1 !U360-6
1460   !
1470   !        Tests 130-149 are for CPU
1480   !
1490     Data 130,1,0,[800000h],0,[400000h],0,[200000h],0,[100000h],0,[80000h],-1 !
 **            U141-5
1500     Data 131,1,0,[40000h],0,[20000h],0,[10000h],0,[8000h],-1 !U141-6
1510     Data 132,1,0,[4000h],0,[2000h],0,[1000h],0,[40h],0,[20h],-1 !U241-6
1520     Data 133,1,0,[10h],0,[8h],-1 !U235-8
1530     Data 134,1,0,[800h],0,[400h],0,[200h],0,[100h],0,[80h],-1 !U241-5
1540     Data 135,1,[F00000h],[700000h],[F00000h],[B00000h],[F00000h],[D00000h],[F0
 **            0000h],[E00000h],-1 !U131-8
1550     Data 136,1,[68000h],[48000h],[68000h],[28000h],[68000h],[60000h],-1 !U125-
 **            8
1560     Data 137,1,0,[80000h],0,[10000h],0,[4000h],0,[2000h],0,[1000h],-1 !U135-6
1570     Data 138,1,0,[80000h],0,[40000h],-1 !U121-8
1580     Data 139,1,0,[4000h],0,[10h],0,[800h],-1 !U125-6
1590     Data 140,1,0,[8000h],0,[10h],-1 !U221-3
1600     Data 141,1,[C0000h],[40000h],[C0000h],[80000h],[C0000h],[E0000h],-1 !U131-
```

```
 **            6
1610      Data 142,1,[80000h],0,[80000h],[C0000h],[80000h],[A0000h],-1 !U235-6
1620      Data 143,1,[F68000h],0,[F68000h],[80h],[F68000h],0,[F68000h],[10000h],-1 !
 **            U225-8
1630      Data 144,1,[FC0000h],0,-1 !U221-8
1640      Data 145,1,[F80000h],0,-1 !U221-11
1650      Data 146,1,[8000h],0,[8000h],[FC8000h],-1 !U121-11
1660    !
1670    !         Tests 150-159 are for Optional RAM
1680    !
1690      Data 150,1,0,[800000h],0,[400000h],0,[200000h],0,[100000h],0,[80000h],-1 !
 **            U285-6
1700      Data 151,1,[8000h],0,[10000h],0,[18000h],0,[20000h],0,-1 !U281-1
1710      Data 152,1,[18000h],0,[20000h],0,-1 !U281-2
1720      Data 153,1,[10000h],0,[20000h],0,-1 !u281-3
1730      Data 160,1,0,[800000h],0,[400000h],0,[200000h],0,[100000h],-1 !U285-6 Opt
 **            24-25
1740      Data 161,1,0,[8000h],0,[10000h],0,[20000h],0,[40000h],0,[80000h],-1 !U285-
 **            5
1750      Data 162,1,0,[8000h],[80000h],[10000h],[C0000h],[40000h],[A0000h],[20000h]
 **            ,[E0000h],[60000h],-1 !U285-1
1760      Data 163,1,0,[20000h],[4000h],[60000h],[8000h],[60000h],[40000h],[20000h],
 **            -1 !U281-2
1770      Data 164,1,0,[40000h],[4000h],[60000h],[8000h],[40000h],[C0000h],[40000h],
 **            [E0000h],[60000h],-1 !U281-3
1780      Data -1,-1 !End of Table
2000    !.page
2010    ! Romtst,Ramtst,Int7,0,Chptst,0,Acetst,Ptmtst,Gpitst,Aciats
2020      Data [312Eh],[3041h] !Version 1.0A
2030      Data [FCh],[484h],[FCh],[4C6h],0,0,0,0,[FCh],[724h],0,0
2040      Data [FCh],[89Ch],[FCh],[7E6h],[FCh],[92Eh],[FCh],[A82h]
2050    !
2060      Data [312Eh],[3120h] !Version 1.1
2070      Data [FCh],[484h],[FCh],[4C6h],0,0,0,0,[FCh],[724h],0,0
2080      Data [FCh],[89Ch],[FCh],[7E6h],[FCh],[92Eh],[FCh],[A82h]
2090    !
2100      Data [322Eh],[3020h] !Version 2.0
2110      Data [FCh],[44Ch],[FCh],[48Eh],[FCh],[3Ah],0,0,[FCh],[698h],0,0
2120      Data [FCh],[810h],[FCh],[75Ah],[FCh],[8A2h],[FCh],[9FEh]
2130    !
2140      Data [322Eh],[3120h] !Version 2.1
2150      Data [FCh],[458h],[FCh],[49Ah],[FCh],[3Ah],0,0,[FCh],[6A4h],0,0
2160      Data [FCh],[81Ch],[FCh],[766h],[FCh],[8AEh],[FCh],[A0Ah]
2170    !
2180      Data -1,-1
2190      Data -1,-1
3000    !.page
3010 Dt72_d:     data [312Eh],[3041h] !Version 1.0A
3020      Data [FC0000h],[FCh],[5Eh],8192 !OS
3030      Data [FC4000h],[4D36h],[384Bh],8192 !IO PAIR 1
3040      Data [FC8000h],[48E7h],[40C0h],8192 !IO PAIR 2
3050      Data [FCC000h],[4D36h],[384Bh],8192 !XO PAIR 1
3060      Data [FD0000h],[3602h],[243h],8192 !XO PAIR 2
3070      Data [FD4000h],[4E55h],[FFEEh],8192 !XO PAIR 3
```

File: tss2 01-NOV-84 08:34:58

```
3080     Data [F80000h],[4D36h],[384Bh],8192 !PD PAIR 1
3090     Data [F84000h],[7234h],[B22Eh],8192 !PD PAIR 2
3100     Data [F00000h],[4D36h],[384Bh],4096 !OPT. 1
3110     Data [E80000h],[4D36h],[384Bh],2048 !OPT. 2
3120     Data -1,-1,-1,-1
3130   !
3140     Data [312Eh],[3120h] !Version 1.1
3150     Data [FC0000h],[FCh],[5Eh],8192 !OS
3160     Data [FC4000h],[4D36h],[384Bh],8192 !IO PAIR 1
3170     Data [FC8000h],[48E7h],[40C0h],8192 !IO PAIR 2
3180     Data [FCC000h],[4D36h],[384Bh],8192 !XO PAIR 1
3190     Data [FD0000h],[3602h],[243h],8192 !XO PAIR 2
3200     Data [FD4000h],[4E55h],[FFEEh],8192 !XO PAIR 3
3210     Data [F80000h],[4D36h],[384Bh],8192 !PD PAIR 1
3220     Data [F84000h],[7234h],[B22Eh],8192 !PD PAIR 2
3230     Data [F00000h],[4D36h],[384Bh],4096 !OPT. 1
3240     Data [E80000h],[4D36h],[384Bh],2048 !OPT. 2
3250     Data -1,-1,-1,-1
3260   !
3270     Data [322Eh],[3020h] !Version 2.0
3280     Data [FC0000h],[6000h],[6Ch],8192 !OS
3290     Data [FC4000h],[4D36h],[384Bh],8192 !IO PAIR 1
3300     Data [FC8000h],[C288h],[1B82h],8192 !IO PAIR 2
3310     Data [FCC000h],[4D36h],[384Bh],8192 !XO PAIR 1
3320     Data [FD0000h],[6100h],[B42h],8192 !XO PAIR 2
3330     Data [FD4000h],[6100h],[CB4Ah],8192 !XO PAIR 3
3340     Data [FDC000h],[4D36h],[384Bh],2048 !XO PAIR 4
3350     Data [F80000h],[4D36h],[384Bh],8192 !PD PAIR 1
3360     Data [F84000h],[4E5Dh],[4E75h],8192 !PD PAIR 2
3370     Data [F84000h],[4D36h],[384Bh],8192 !MISC. ROMPACK
3380     Data [F88000h],[4D36h],[384Bh],8192 !MISC. ROMPACK
3390     Data [F8C000h],[4D36h],[384Bh],8192 !MISC. ROMPACK
3400     Data [F90000h],[4D36h],[384Bh],8192 !MISC. ROMPACK
3410     Data [F94000h],[4D36h],[384Bh],8192 !MISC. ROMPACK
3420     Data [F00000h],[4D36h],[384Bh],4096 !OPTION 1
3430     Data [E80000h],[4D36h],[384Bh],2048 !OPTION 2
3435     Data [D80000h],[4D36h],[384Bh],8192 !OPTION 3
3440     Data -1,-1,-1,-1
3450   !
3460     Data [322Eh],[3120h] !Version 2.1
3470     Data [FC0000h],[6000h],[6Ch],8192 !OS
3480     Data [FC4000h],[4D36h],[384Bh],8192 !IO PAIR 1
3490     Data [FC8000h],[202Dh],[1Ah],8192 !IO PAIR 2
3500     Data [FCC000h],[4D36h],[384Bh],8192 !XO PAIR 1
3510     Data [FD0000h],[4278h],[A06h],8192 !XO PAIR 2
3520     Data [FD4000h],[4E51h],[FFDAh],8192 !XO PAIR 3
3530     Data [FDC000h],[4D36h],[384Bh],2048 !XO PAIR 4
3540     Data [F80000h],[4D36h],[384Bh],8192 !PD PAIR 1
3550     Data [F84000h],[4E5Dh],[4E75h],8192 !PD PAIR 2
3560     Data [F84000h],[4D36h],[384Bh],8192 !MISC. ROMPACK
3570     Data [F88000h],[4D36h],[384Bh],8192 !MISC. ROMPACK
3580     Data [F8C000h],[4D36h],[384Bh],8192 !MISC. ROMPACK
3590     Data [F90000h],[4D36h],[384Bh],8192 !MISC. ROMPACK
3600     Data [F94000h],[4D36h],[384Bh],8192 !MISC. ROMPACK
```

File: tss2 01-NOV-84 08:36:49

```
 3610     Data [F00000h],[4D36h],[384Bh],4096 !OPTION 1
 3620     Data [E80000h],[4D36h],[384Bh],2048 !OPTION 2
 3625     Data [D80000h],[4D36h],[384Bh],8192 !OPTION 3
 3630     Data -1,-1,-1,-1
 3640     Data -1,-1 !END OF DT72 TEST TABLE
 5000   !.page
 5010   Rem
 5020   Rem              TEST FIXTURE CODE FOR THE 4041
 5030   Rem
 5040   Rem              DEFINE THE DATA BUFFERS
 5050   Rem
 5060     Dim testnum(120),testtyp(120),testndx(120),t6list(50)
 5070     Dim dinbuf(32)
 5080     Dim t5adr(10,22)
 5090     Long t1list(256),t23tab(50,2),t4list(50),t5list(400)
 5100     Long regbuf(16),reghbuf(16),reglbuf(16),ldata
 5110     Dim cmnd$ to 50,svcmnd$ to 50,line$ to 80,msg1$ to 30
 5120     Valnonum=101
 5130   Rem
 5140   Rem               DEFINE SOME BUFFER INDEX VALUES
 5150   Rem
 5160     T1ndx=1
 5170     T23ndx=1
 5180     T4ndx=1
 5190     T5ndx=1
 5200     T6ndx=1
 5210     Irq7=96
 5220   Rem
 5230   Rem  set the baud rate in the following statement to the correct
 5240   Rem  value for your system
 5250   Rem
 5260     Set driver "comm(baud=2400):"
 5270     Set console "COMM:"
 5280     Print
 5290     Print "4041 System Test Fixture    067-1070-03"
 5300     Print "Copyright (c) Tektronix, Inc.  1984"
 5310     Print "Version 2.3"
 5320     Rcall "OPT2IN",1,1,pmtyp
 5330     If pmtyp=1 then goto 5370
 5340     Print "Error.....Illegal Personailty ID Code.^G^G"
 5350     Stop
 5360     Goto 5280
 5370     Print
 5380     Print "Initializing"
 5390   Rem
 5400   Rem              BUILD ALL THE TEST TABLES
 5410   Rem
 5420     Call tfinit !Initialize the personality module and set up variables
 5430     Restore
 5440     Tti=1
 5450 Btloop:     read tnum,ttyp !Get a test number and class
 5460     If tnum<0 then goto pue !end of table?
 5470   Rem              GO TO THE PROPER HANDLER FOR THIS TYPE OF TEST
 5480     If ttyp=0 then goto typ0
```

File: tss2 01-NOV-84 08:38:07

```
5490     If ttyp=1 then goto typ1
5500     If ttyp=2 then goto typ2
5510     If ttyp=3 then goto typ3
5520     If ttyp=4 then goto typ4
5530     If ttyp=5 then goto typ5
5540     If ttyp=6 then goto typ6
5550     If ttyp=7 then goto typ7
5560     Print "TTYP ERROR......TTYP=";ttyp
5570 Typ0:     print "ERROR IN TEST TABLE INITIALIZER"
5580     Stop
5590     Goto 5580 !hang him up if he tries to continue
5600   !
5610   !  Type 1: Set up the test tables to point to the first address in
5620   !          <T1LIST>.  Then fill up <T1LIST> with all the appropriate
5630   !          addresses.
5640   !
5650 Typ1:     testnum(tti)=tnum
5660     Testtyp(tti)=ttyp
5670     Testndx(tti)=t1ndx
5680     Tti=tti+1
5690   Rem               GET THE ADDRESS LIST
5700 T1loop:     read address
5710     T1list(t1ndx)=address
5720     T1ndx=t1ndx+1
5730     If address<0 then goto btloop
5740     Goto t1loop
5750   !
5760   !  Type 2 and Type 3: These tests share a common table.
5770   !                     <T23NDX(x,1)> is address to be accessed.
5780   !                     <T23NDX(x,2)> is data to be written in type 3.
5790   !                     <T23NDX(x,2)> is unused in type 2.
5800   !                     <X>-----| is the value of <TESTNDX> for this test.
5810   !
5820 Typ2:   rem
5830 Typ3:     testnum(tti)=tnum
5840     Testtyp(tti)=ttyp
5850     Testndx(tti)=t23ndx
5860     Tti=tti+1
5870     Read address,sutdat
5880     T23tab(t23ndx,1)=address
5890     T23tab(t23ndx,2)=sutdat
5900     T23ndx=t23ndx+1
5910     Goto btloop
5920   !
5930   !  Type 4: First Element of <T4LIST> is address to be accessed.
5940   !          Subsequent entries are the data items to be written, read
5950   !          and compared for correctness.  Any errors are reported.
5960   !          Data list for a given test ends with a -1.
5970   !
5980 Typ4:     testnum(tti)=tnum
5990     Testtyp(tti)=ttyp
6000     Testndx(tti)=t4ndx
6010     Tti=tti+1
6020 T4loop:     read address
```

```
6030     T4list(t4ndx)=address
6040     T4ndx=t4ndx+1
6050     If address<0 then goto btloop
6060     Goto t4loop
6070   !
6080   !  Type 5: Each test occupies a 32 entry block of <T5LIST>, which
6090   !          correspond to the high and low bytes of the 16 registers
6100   !          in the 68000.
6110   !
6120 Typ5:     testnum(tti)=tnum
6130     Testtyp(tti)=ttyp
6140     Testndx(tti)=t5ndx
6150     Tti=tti+1
6160     For q=1 to 32
6170       Read t5list(t5ndx)
6180       T5ndx=t5ndx+1
6190       Next q
6200     Goto btloop
6210   !
6220   !  Type 6: <T6LIST> entries are the test numbers of the tests to
6230   !          executed by this macro test.  Each list segment terminates
6240   !          with a -1.
6250   !
6260 Typ6:     testnum(tti)=tnum
6270     Testtyp(tti)=ttyp
6280     Testndx(tti)=t6ndx
6290     Tti=tti+1
6300 T6loop:     read t6list(t6ndx)
6310     T6ndx=t6ndx+1
6320     If t6list(t6ndx-1)=-1 then goto btloop else goto t6loop
6330   !
6340   !  Type 7:  The <TESTNDX> entry for each of these tests is an index
6350   !           to be used by the dispatcher to get to the appropriate
6360   !           special function test.
6370   !
6380 Typ7:     testnum(tti)=tnum
6390     Testtyp(tti)=ttyp
6400     Read testndx(tti)
6410     Tti=tti+1
6420     Goto btloop
6430   !
6440   !    End of table detected.  Clean up
6450   !
6460 Pue:     testnum(tti)=-1
6470     Testtyp(tti)=-1
6480     Testndx(tti)=-1
6490     Tnmax=tti
6500     On abort then call absub
6510   !Allow the abort function
6520     Enable abort
6530     On error(valnonum) then call alers
6540     Call tfinit
6550     Tmode=0 !Set <TMODE> to 0 ===> no executing a type 6 test sequence.
6560     Index=1
```

File: tss2 01-NOV-84 08:40:42

```
6570   !
6580   !    Read all the index structures for the type 5 tests.  When a
6590   !    type 5 test is executed, the structure built here will be
6600   !    accessed to locate the entry points for routines in the
6610   !    target 4041 which will be used.
6620   !
6630 Pue5:     for 1=1 to 22
6640       Read t5adr(index,1)
6650       If 1=1 and t5adr(index,1)=-1 then exit to idle
6660       Next 1
6670     Index=index+1
6680     Goto pue5
6690   !
6700   !    label <IDLE> is the point to which all tests return upon completion
6710   !
6720 Idle:     if tmode=1 then goto dt6i !If executing a type 6 test, go back and g
 **            et a new one
6730   Rem
6740     Call getfcn ! Get a test number to execute in <TNUM>
6750     Tti=1
6760   !
6770   !   Search for the correct test and dispatch to proper handler
6780   !
6790 Tisrch:     if testnum(tti)=tnum then goto testtyp(tti) of dt1,dt2,dt3,dt4,dt5
 **             ,dt6,dt7
6800     Tti=tti+1
6810     If tti<tnmax then goto tisrch
6820     Print "Error....Test does not exist"
6830     Goto idle
6840   !
6850   !  Type 1: Set up and output addresses in <T1LIST> until he hits Abort
6860   !
6870 Dt1:     t1ndx=testndx(tti)
6880     Call stopit
6890 Dt1a:     index=t1ndx
6900 Dt1l:     address=t1list(index)
6910     If address<0 then goto dt1a
6920     Outadr=address bor 1
6930     Rcall "opt2out",waddr,4,outadr
6940     Rcall "opt2out",bufstb,1,dummy
6950     Index=index+1
6960     Goto dt1l
6970   !  Type 2: Set up one read operation is <READB>, the loop forever
6980   !          strobing additional read operations.
6990   !
7000 Dt2:     call stopit
7010     Address=t23tab(testndx(tti),1)
7020     Sutdat=0
7030     Call readb
7040 Dt2l:     rcall "opt2out",bufstb,1,0
7050     Rcall "opt2out",memstb,1,0
7060     Goto dt2l
7070   !
7080   !  Type 3: Set up one write operation with <WRITEW>, then loop forever
```

File: tss2 01-NOV-84 08:42:01

```
 7090   !          strobing in additional write operations.
 7100   !
 7110 Dt3:     call stopit
 7120     Address=t23tab(testndx(tti),1)
 7130     Sutdat=t23tab(testndx(tti),2)
 7140     Call writew
 7150 Dt31:     rcall "opt2out",bufstb,1,0
 7160     Rcall "opt2out",memstb,1,0
 7170     Goto dt31
 7180   !
 7190   !  Type 4:  Get the address from the list, and then each data item in
 7200   !           turn, doing a write, read, check sequence on each.
 7210   !
 7220 Dt4:     call stopit
 7230     Index=testndx(tti)
 7240     T4ndx=index+1
 7250     Address=t4list(index)
 7260     Index=index+1
 7270 Dt41:     sutdat=t4list(index)
 7280     If sutdat<0 then goto idle
 7290     Savdat=sutdat
 7300     Call writeb
 7310     Call readb
 7320     If savdat<>sutdat then goto der4
 7330     Index=index+1
 7340     Goto dt41
 7350 Der4:     em1$="Data Written"
 7360     Em2$="Data Read"
 7370     Print using "12ax2hx9ax2h":em1$;savdat;em2$;sutdat
 7380     Goto idle
 7390   !
 7400   !  Type 5: Start by setting up the interupt vector to point to our
 7410   !          monitor code in the target 4041.  Then force the machine
 7420   !          to execute to a point midway through that block of code
 7430   !          and stop it.
 7440   !
 7450 Dt5:     call stopit
 7460     Address=irq7
 7470     Sutdat=(tfin7 band [FF0000h])/[10000h]
 7480     Call writew
 7490     Sutdat=tfin7 band [FFFFh]
 7500     Address=address+2
 7510     Call writew
 7520     Brkad=(tfin7+12) band [FFFFFFh]
 7530     Rcall "opt2out",adrcmp,4,brkad
 7540     Cmdmap=cmdmap bor [80h]
 7550     Rcall "opt2out",commnd,1,cmdmap
 7560     Rcall "opt2out",ip7en,1,dummy
 7570     Rcall "opt2out",ssdis,1,dummy
 7580     Rcall "opt2out",start,1,dummy
 7590 Dt51:     rcall "opt2in",rstat1,1,status
 7600     If (status band [2h])=1 then goto dt51
 7610     If (status band [20h])=1 then goto dt51
 7620   !
```

File: tss2 01-NOV-84 08:56:18

```
7630   !  Read the firmware version from the target machine and find a matching
7640   !  entry in <T5LIST>.
7650   !
7660     Address=tfvers
7670     Call readw
7680     Tfv1=sutdat
7690     Address=address+2
7700     Call readw
7710     Index=1
7720 Dt50:     if t5adr(index,1)=tfv1 and t5adr(index,2)=sutdat then goto dt52
7730     If t5adr(index,1)<>-1 then goto dt51
7740     Print "This firmware version not supported"
7750     Goto idle
7760 Dt51:     index=index+1
7770     Goto dt50
7780   !
7790   !  Load up the communication RAM with the new register set.
7800   !
7810 Dt52:     address=[300h]
7820     Ind=testndx(tti)
7830     For q=0 to 31
7840       Sutdat=t5list(ind+q)
7850       If sutdat<0 then sutdat=t5adr(index,abs(sutdat)+2)
7860       Call writew
7870       Address=address+2
7880       Next q
7890   !
7900   !  Shut off the interupt and let the test execute.
7910   !
7920     Rcall "opt2out",ip7dis,1,dummy
7930     Call bkadr
7940     Call getreg !get the register file from the communication RAM.
7950   !
7960   !  Compare the registers with what we originally loaded.  If there
7970   !  are no differences, the test probably did not run.  This would mean
7980   !  that the 68000 was in a wierd state and could not run.  In that
7990   !  case, notify the user that the test may not have executed.
8000   !
8010     Testrun=0
8020     For q=0 to 31 step 2
8030       Sutdat=t5list(ind+q)
8040       If sutdat<0 then sutdat=t5adr(index,abs(sutdat)+2)
8050       Savdat=t5list(ind+q+1)
8060       If savdat<0 then savdat=t5adr(index,abs(savdat)+2)
8070       If sutdat<>reghbuf((q+1)/2) or savdat<>reglbuf((q+1)/2) then testrun=1
8080       Next q
8090     Call prntreg
8100     If testrun=0 then print "^G^G^G*******Register data unaltered."
8110     If testrun=0 then print "^G^G^G*******Test may have NOT EXECUTED."
8120     Goto idle
8130   !
8140   !  Type 6: Set <TMODE> to 1 and run the tests specified in sequence.
8150   !
8160 Dt6:      call stopit
```

```
 8170     T6ndx=testndx(tti)
 8180     Tmode=1
 8190 Dt6l:     tnum=t6list(t6ndx)
 8200     If tnum=-1 then goto dt6out
 8210     Print "--------------------------"
 8220     Print using "21AX3D":"Executing test number";tnum
 8230     Tti=1
 8240     Goto tisrch
 8250 Dt6i:     t6ndx=t6ndx+1
 8260     Goto dt6l
 8270 Dt6out:     tmode=0
 8280     Print using "/":
 8290     Goto idle
 8300   !
 8310   !  Type 7: Dispatch off to the appropriate special test.
 8320   !
 8330 Dt7:     goto testndx(tti) of dt71,dt72,dt73,dt74,dt75,dt76,dt77,dt78
 8340     Print "ERROR IN DATA STRUCTURE"
 8350     Stop
 8360   !
 8370   !  Test #403:  Output two messages to the FPC.  The first should cause
 8380   !              the printer to rattle, the second will place a message
 8390   !              in the LED display.
 8400   !
 8410 Dt71:     msg1$="Printer Test 1234567"
 8420     Msg1$=chr$([F1h])&chr$([F8h])&chr$([F2h])&msg1$&chr$([B2h])
 8430     For l=1 to len(msg1$)
 8440       Sutdat=asc(seg$(msg1$,l,1))
 8450       Address=[F40803h]
 8460       Call writeb
 8470       Next l
 8480     Msg1$="THIS IS A TEST******"
 8490     Msg1$=chr$([F1h])&chr$([A0h])&chr$([F8h])&chr$([81h])&msg1$
 8500     For l=1 to len(msg1$)
 8510       Sutdat=asc(seg$(msg1$,l,1))
 8520       Address=[F40803h]
 8530       Call writeb
 8540       Next l
 8550     Goto idle
 8560   !
 8570   !  Test #404:  Test all the ROMs in the machine.  Do basically the
 8580   !              same thing a the type 5 tests did.  That is, take
 8590   !              control of the machine and force it to execute the
 8600   !              self test for the <ROMTST> sequence.
 8610   !
 8620 Dt72:     call stopit
 8630     Address=tfvers
 8640     Call readw
 8650     Savdat=sutdat
 8660     Address=address+2
 8670     Call readw
 8680   !
 8690   !  First, find the table entry in <DT72_D> for this version of firmware.
 8700   !
```

File: tss2 01-NOV-84 08:58:59

```
 8710     Restore dt72_d
 8720     Read version,version1
 8730     If version=savdat and version1=sutdat then goto dt72_s
 8740     If version<>-1 then goto dt72_1
 8750     Print "This firmware version not supported"
 8760     Goto idle
 8770 Dt72_1:     read version,version,version,version
 8780     If version<>-1 then goto dt72_1 else goto 8720
 8790   !
 8800   !   We've found the correct table entries, start testing the ROMs
 8810   !
 8820 Dt72_s:     incon=0
 8830     Count=0
 8840 Dt721:     read adr,vdata1,vdata2,romsiz
 8850     If adr=-1 then goto dt72done !end of list?
 8860   !
 8870   !   Check to see if we have a valid ROM pair at this address
 8880   !
 8890     Address=adr
 8900     Call readw
 8910     If sutdat<>vdata1 then goto dt721
 8920     Address=address+2
 8930     Call readw
 8940     If sutdat<>vdata2 then goto dt721
 8950   !
 8960   !   Check to see if we already have control, if not take control and
 8970   !   do the first test.
 8980   !
 8990     If incon=1 then goto dt72c
 9000     Address=irq7
 9010     Sutdat=(tfin7 band [FF0000h])/[10000h]
 9020     Call writew
 9030     Sutdat=tfin7 band [FFFFh]
 9040     Address=address+2
 9050     Call writew
 9060     Brkad=tfin7+12
 9070     Rcall "opt2out",adrcmp,4,brkad
 9080     Cmdmap=cmdmap bor [80h]
 9090     Rcall "opt2out",commnd,1,cmdmap
 9100     Rcall "opt2out",ip7en,1,dummy
 9110     Call bkwait
 9120     Address=tfvers
 9130     Call readw
 9140     Tfv1=sutdat
 9150     Address=address+2
 9160     Call readw
 9170     Index=1
 9180   !
 9190   !  Given the firmware version, find out where is <ROMTST> code is.
 9200   !
 9210 Dt72a:     if t5adr(index,1)=tfv1 and t5adr(index,2)=sutdat then goto dt72c
 9220     If t5adr(index,1)<>-1 then goto dt72b
 9230     Print "This firmware version not supported"
 9240     Goto idle
```

```
9250 Dt72b:     index=index+1
9260     Goto dt72a
9270   !
9280   !   Load up the required data for <ROMTST>
9290   !
9300   !         Address x'304 ==> D1: ROM size
9310   !         Address x'324 ==> A1: Base address of ROM
9320   !         Address x'334 ==> A5: Return Pointer
9330   !         Address x'338 ==> A6: ROMTST entry
9340   !         Address x'33C ==> A7: Stack pointer
9350   !
9360 Dt72c:     address=[304h]
9370     Incon=1
9380     Sutdat=(romsiz band [FF0000h])/[10000h]
9390     Call writew
9400     Address=address+2
9410     Sutdat=romsiz band [FFFFh]
9420     Call writew
9430     Address=[324h]
9440     Sutdat=(adr band [FF0000h])/[10000h]
9450     Call writew
9460     Address=address+2
9470     Sutdat=adr band [FFFFh]
9480     Call writew
9490     Address=[334h]
9500     Sutdat=t5adr(index,7)
9510     Call writew
9520     Address=address+2
9530     Sutdat=t5adr(index,8)
9540     Call writew
9550     Address=address+2
9560     Sutdat=t5adr(index,3)
9570     Call writew
9580     Address=address+2
9590     Sutdat=t5adr(index,4)
9600     Call writew
9610     Address=address+2 !set A7
9620     Sutdat=0
9630     Call writew
9640     Address=address+2
9650     Sutdat=[7FF0h]
9660     Call writew
9670     Rcall "opt2out",ip7dis,1,dummy
9680     Call bkadr
9690     Call getreg
9700     If reglbuf(1) band [3h]<>0 then result$="Failed" else result$="Passed"
9710     Print using 9730:"Rom at address";adr;result$
9720     Count=count+1
9730     Image                "14ax6hx6a"
9740   !
9750   !  If we've done both bytes, get next entry.  Else do odd byte.
9760   !
9770     If (adr band 1)=1 then goto dt721
9780     Adr=adr bor 1
```

File: tss2 01-NOV-84 09:01:26

```
 9790     Goto dt72c
 9800 Dt72e:     result$="Failed"
 9810     Return
 9820 Dt72done:     print using 9830:"Number of ROMs tested is";count
 9830     Image"/24ax2d/"
 9840     Goto idle
 9850 Dt73:   rem
 9860 Dt74:     print "Error....Test does not exist"
 9870     Goto idle
 9880   !
 9890   !      Test 320: CPU ROM data pattern verification.
 9900   !                Read 18 words from ROM and verify the correct data
 9910   !
 9920 Dt75:     data 0,1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32768.
  **            0,65535.0
 9930     Restore dt75
 9940     Address=tfbase+[10h]
 9950     For l=1 to 18
 9960       Call readw
 9970       Read savdat
 9980       If sutdat<>savdat then gosub dt75e
 9990       Address=address+2
10000       Next l
10010     Goto idle
10020 Dt75e:     em1$="Data Read"
10030     Em2$="Data Expected"
10040     Print using "9ax4hx13ax4h":em1$,sutdat,em2$,savdat
10050     Return
10060   !
10070   !      Test 330: Blink the Selftest LEDS in sequence.
10080   !
10090 Dt76:     data 0,12,0,16,0,8,0,[1Ch],[4Ch],[1Ch],[50h],[1Ch],[48h],[1Ch],0,[1C
  **            h],0
10100     Restore dt76
10110     For l=1 to 16
10120       Read address
10130       Address=address+[F68000h]
10140       Call readb
10150       Wait 0.300000
10160       Next l
10170     Goto idle
10180   !
10190   !      Test 100: Walk a bit up the address bus.
10200   !
10210 Dt77:     address=0
10220     Call putadr
10230     For l=0 to 23
10240       Address=2^l
10250       Call putadr
10260       Next l
10270     Goto dt77
10280   !
10290   !       Test 101: Walk a bit up the data bus.
10300   !
```

```
10310 Dt78:     sutdat=0
10320     Call putdat
10330    -For l=0 to 15
10340       Sutdat=2^l
10350       Call putdat
10360       Next l
10370     Goto dt78
10380     End


10400 Sub tfinit
10410   !
10420   !   Initialize the test fixture personality module.
10430   !
10440     Integer id,busreq,ip7dis,ip7en,ssdis,ssen,stopp,start,bufstb,memstb,reset
10450     Integer raddr,adrcmp,waddr,rdata,wdata,dtacmp,ivect1,ivect2,rstat0,rstat1,
  **            commnd
10460     Integer byte,status,dummy,cmdmap,inc
10470     Long address,outadr,onum,brkad,gval
10480     Long ad1,l,tfbase,tfin7,tfvers,adr
10490     Tfbase=[FC0000h]
10500     Tfin7=tfbase+[3Ah]
10510     Tfvers=tfbase+[34h]
10520     Bhtxt$="0123456789ABCDEF"
10530     Id=[0h]
10540     Busreq=[1h]
10550     Ip7dis=[2h]
10560     Ip7en=[3h]
10570     Ssdis=[8h]
10580     Ssen=[9h]
10590     Stopp=[Ah]
10600     Start=[Bh]
10610     Bufstb=[Ch]
10620     Memstb=[Dh]
10630     Reset=[Eh]
10640     Raddr=[10h]
10650     Adrcmp=[18h]
10660     Rcall "opt2out",adrcmp,4,0
10670     Waddr=[1Ch]
10680     Rcall "opt2out",waddr,4,0
10690     Rdata=[20h]
10700     Wdata=[28h]
10710     Rcall "opt2out",wdata,2,0
10720     Dtacmp=[2Ah]
10730     Rcall "opt2out",dtacmp,2,0
10740     Ivect1=[2Ch]
10750     Ivect2=[2Eh]
10760     Rstat0=[30h]
10770     Rstat1=[31h]
10780     Commnd=[39h]
10790     Rcall "opt2out",commnd,1,0
10800     Rcall "opt2out",ip7dis,1,dummy
10810     Dummy=0
10820     Cmdmap=0
```

File: tss2 01-NOV-84 09:03:35

```
10830     Return
10840     End


10900 Sub readb
10910   !
10920   !     Read a byte of data from the UUT.  <ADDRESS> contains the address
10930   !     to be read.  Data will be placed in <SUTDAT>.
10940   !
10950     Byte=(address band [1h])
10960     Call readw
10970     If byte=0 then sutdat=int(sutdat/[100h]) else sutdat=(sutdat band [FFh])
10980     Return
10990     End


11100 Sub readw
11110   !
11120   !     Read a word of data from the UUT.  <ADDRESS> contains the address
11130   !     to be read.  Data will be place in <SUTDAT>.
11140   !
11150     Rcall "OPT2IN",rstat1,1,status
11160     If (status band [2h])<>0 then goto noread
11170     Rcall "OPT2OUT",busreq,1,dummy
11180     Outadr=address bor [1h]
11190     Rcall "OPT2OUT",waddr,4,outadr
11200     Rcall "OPT2OUT",bufstb,1,dummy
11210     Rcall "OPT2OUT",memstb,1,dummy
11220     Rcall "OPT2IN",rdata,2,sutdat
11230     Sutdat=sutdat band [FFFFh]
11240 Noread:     return
11250     End


11300 Sub writeb
11310   !
11320   !     Write a byte of data to the UUT.  <ADDRESS> contains the address
11330   !     to be written.  Data to be written is in <SUTDAT>.  We first
11340   !     read a word, the replace the effected byte and the write a word.
11350   !
11360     Byte=(address band [1h])
11370     Olddat=sutdat
11380     Call readw
11390     If byte=0 then goto evenbyte
11400     Sutdat=(sutdat band [FF00h])
11410     Sutdat=sutdat+olddat
11420     Goto dowrt
11430 Evenbyte:     sutdat=(sutdat band [FFh])
11440     Sutdat=sutdat bor (olddat*[100h])
11450 Dowrt:     call writew
11460     Return
11470     End
```

```
11500 Sub writew
11510   !
11520   !     Write a word of data to the UUT,  <ADDRESS> contains the address
11530   !     to be written.  Data to be written is in <SUTDAT>.
11540   !
11550     Rcall "OPT2IN",rstat1,1,status
11560     If (status band [2h])<>0 then goto nowrite
11570     Rcall "OPT2OUT",busreq,1,dummy
11580     Outadr=address band [FFFFFEh]
11590     Rcall "OPT2OUT",waddr,4,outadr
11600     Rcall "OPT2OUT",wdata,2,sutdat
11610     Rcall "OPT2OUT",bufstb,1,dummy
11620     Rcall "OPT2OUT",memstb,1,dummy
11630 Nowrite:     return
11640     End


11700 Sub runit
11710   !
11720   !     Allow the 68000 in the UUT to run.
11730   !
11740     Rcall "OPT2IN",rstat1,1,status
11750     If (status band [1h])=0 then goto runexit
11760     If (status band [2h])<>0 then goto runexit
11770     Cmdmap=(cmdmap band [3Fh])
11780     Rcall "OPT2OUT",commnd,1,cmdmap
11790     Rcall "OPT2OUT",ssdis,1,dummy
11800     Rcall "OPT2OUT",start,1,dummy
11810 Runexit:     return
11820     End


11900 Sub stopit local count
11910   !
11920   !     Stop the 68000 in the UUT.
11930   !
11940     Count=1
11950 Halt:     rcall "OPT2OUT",stopp,1,dummy
11960     Rcall "OPT2IN",rstat1,1,status
11970     If (status band [2h])<>0 then goto halt
11980     Rcall "OPT2OUT",busreq,1,dummy
11990     If (status band [80h])=0 then return
12000     Count=count+1
12010     If count<5 then goto halt
12020     Print "Buss access error"
12030     Return
12040     Return
12050     End


12100 Sub stepit
12110   !
12120   !     Cause <NSTEPS> bus cycles to occur in the 68000.
12130   !
```

File: tss2 01-NOV-84 09:05:46

```
12140     Call stopit
12150     For inc=1 to nsteps
12160       Rcall "OPT2IN",rstat1,1,status
12170       If (status band [1h])=0 then goto stepexit
12180       If (status band [2h])<>0 then goto stepexit
12190       Rcall "OPT2OUT",ssen,1,dummy
12200       Rcall "OPT2OUT",start,1,dummy
12210       Rcall "opt2in",rdata,2,sutdat
12220       Rcall "opt2in",raddr,4,address
12230       Address=address band [FFFFFFh]
12240       Print using "8a6h6a4h":"Address:";address;" Data:";sutdat
12250       Next inc
12260 Stepexit:     inc=0
12270     Return
12280     End


12300 Sub bkadr
12310   !
12320   !    Allow the 68000  to run until address <BKADR> is accessed.
12330   !
12340     Brkad=brkad band [FFFFFEh]
12350     Rcall "OPT2OUT",adrcmp,4,brkad
12360     Cmdmap=(cmdmap bor [80h])
12370     Rcall "OPT2OUT",commnd,1,cmdmap
12380     Call bkwait
12390     Return
12400     End


12500 Sub bkdta
12510   !
12520   !    Allow the 68000 to run until data item <BRKDAT> is transferred
12530   !    on the bus.
12540   !
12550     Rcall "OPT2OUT",dtacmp,2,brkdat
12560     Cmdmap=(cmdmap bor [40h])
12570     Rcall "OPT2OUT",commnd,1,cmdmap
12580     Call bkwait
12590     Return
12600     End


12700 Sub putadr
12710   !
12720   !     Place address <ADDRESS> on the 68000 bus.
12730   !
12740     Rcall "OPT2IN",rstat1,1,status
12750     If (status band [2h])<>0 then goto paexit
12760     Outadr=address bor [1h]
12770     Rcall "OPT2OUT",waddr,4,outadr
12780     Rcall "OPT2OUT",bufstb,1,dummy
12790 Paexit:     return
12800     End
```

```
12900 Sub putdat
12910   !
12920   !     Put <SUTDAT> on the data bus.
12930   !
12940     Rcall "OPT2IN",rstat1,1,status
12950     If (status band [2h])<>0 then goto pdexit
12960     Outadr=address band [FFFFFEh]
12970     Rcall "OPT2OUT",waddr,4,outadr
12980     Rcall "OPT2OUT",wdata,2,sutdat
12990     Rcall "OPT2OUT",bufstb,1,dummy
13000 Pdexit:     return
13010     End


13100 Sub bkwait
13110   !
13120   !     Wait for a break to occur.
13130   !
13140     Rcall "OPT2OUT",ssdis,1,dummy
13150     Rcall "OPT2OUT",start,1,dummy
13160 Bktest:     rcall "OPT2IN",rstat1,1,status
13170     If (status band [1h])=0 then goto brkexit
13180     If (status band [2h])<>0 then goto bktest
13190 Brkexit:     rcall "OPT2IN",rdata,2,sutdat
13200     Rcall "OPT2IN",raddr,4,address
13210     Address=address band [FFFFFFh]
13220     Sutdat=sutdat band [FFFFh]
13230     Cmdmap=(cmdmap band [3Fh])
13240     Rcall "OPT2OUT",commnd,1,cmdmap
13250     Return
13260     End


13300 Sub normrst
13310   !
13320   !    Restart the system under test.
13330   !
13340     Cmdmap=(cmdmap band [DFh])
13350     Rcall "OPT2OUT",commnd,1,cmdmap
13360     Call restart
13370     Return
13380     End


13400 Sub vectrst
13410     Rcall "OPT2OUT",ivect1,4,address
13420     Cmdmap=(cmdmap bor [20h])
13430     Rcall "OPT2OUT",commnd,1,cmdmap
13440     Call restart
13450     Return
13460     End
```

File: tss2 01-NOV-84 09:07:47

```
13500 Sub restart
13510   !
13520   !     Issue the reset pulse.
13530   !
13540     Rcall "OPT2OUT",reset,1,dummy
13550     Wait 0.1
13560     Return
13570     End


13600 Sub getfcn
13610   !
13620   !     Main command interpretter.  Prompts the user for input.  If
13630   !     the input command is a numbered test, the value is placed in
13640   !     <TNUM> and we return to the main to execute it.  If it is
13650   !     one of the commands we recognize, we process it locally.
13660   !
13670     Set upcase 1
13680 Gfin:     input prompt ">>>":cmnd$
13690     Call tfinit !reset the personality module
13700     If cmnd$="EXIT" then stop
13710     If len(cmnd$)=0 then cmnd$=svcmnd$
13720     Svcmnd$=cmnd$
13730     Cmnd$=trim$(cmnd$)
13740     Tnum=0
13750     Call stopit
13760     Dp$=seg$(cmnd$,1,2)
13770     If dp$="GO" then tnum=1
13780     If dp$="ST" then tnum=2
13790     If dp$="DM" then tnum=3
13800     If dp$="OP" then tnum=4
13810     If dp$="BA" then tnum=5
13820     If dp$="IN" then tnum=6
13830     If dp$="BD" then tnum=7
13840     If dp$="RE" then tnum=8
13850     If dp$="HE" then tnum=9
13860     If dp$="HA" then tnum=10
13870     If tnum>0 then goto gfdsm
13880     If len(cmnd$)=3 then tnum=val(cmnd$)
13890     If tnum<>0 then goto gfdsm
13900     Print "ILLEGAL COMMAND"
13910     Svcmnd$=""
13920     Goto gfin
13930 Gfdsm:     if tnum>50 then return
13940     Goto tnum of g1,g2,g3,g4,g5,g6,g7,g8,g9,g10
13950     Print "DISPATCHER ERROR"
13960     Goto gfin
13970 G1:   ! "go" command
13980 G1run:     call runit
13990     Goto gfin
14000 G2:   rem             "STEP"...MAY HAVE A NUMBER OF STEPS TO TAKE
14010     Call getnum(1)
14020     Nsteps=gval
14030     Call stepit
```

```
14040     Goto gfin
14050 G3:   rem              DISPLAY MEMORY
14060     Call getnum(1)
14070     Address=gval
14080     Nsteps=16
14090     Call getnum(0)
14100     If gval<>0 then nsteps=gval
14110 G3loop:     for l=1 to 16
14120       Call readb
14130       Dinbuf(l)=sutdat
14140       Address=address+1
14150       Next l
14160     Address=address-16
14170     Print using "6HXS":address
14180     For l=1 to 16
14190       Print using "2HXS":dinbuf(l)
14200       Next l
14210     For l=1 to 16
14220       C$="."
14230       If dinbuf(l)>=32 and dinbuf(l)<127 then c$=chr$(dinbuf(l))
14240       Print c$;
14250       Next l
14260     Print
14270     Address=address+16
14280     Nsteps=nsteps-16
14290     If nsteps>0 then goto g3loop
14300     Print
14310     Goto gfin
14320 G4:   rem              Open memory for update
14330     Call getnum(1)
14340 G4a:     address=gval
14350 G4loop:     call readb
14360     For l=1 to 10
14370       Rcall "opt2in",rstat1,1,status
14380       If (status band 1)=0 then exit to g4berr
14390       Next l
14400     Be$=" "
14410     Goto g4prompt
14420 G4berr:     be$="*"
14430 G4prompt:     putmem buffer line$ using "6H:1A2HX":address;be$;sutdat
14440     Input prompt line$:cmnd$
14450     If len(cmnd$)>0 then goto g4b
14460     Address=address+1
14470     Goto g4loop
14480 G4b:     cmnd$=trim$(cmnd$)
14490     If len(cmnd$)=0 then goto g4loop
14500     If cmnd$="^" then goto g4back
14510     If cmnd$="." then goto gfin
14520     Call getnum(0)
14530     If gval>=0 and gval<=255 then goto g4c
14540     Print "ERROR...DATA OUT OF RANGE"
14550     Goto g4loop
14560 G4c:     sutdat=gval
14570     Call writeb
```

File: tss2 01-NOV-84 09:10:07

```
14580     Goto g4loop
14590 G4back:     address=address-1
14600     Goto g4loop
14610 G5:   rem               "BA" SET A BREAK ADDRESS AND RUN
14620     Call getnum(1)
14630     Brkad=gval
14640     Call bkadr
14650     Print using "8a6h6a4h":"Address:";address;" Data:";sutdat
14660     Goto gfin
14670 G6:   rem               "IN" INIT THE SYSTEM
14680     Call restart
14690     Goto gfin
14700 G7:   rem               "BD" SET A DATA BREAKPOINT AND RUN
14710     Call getnum(1)
14720     Brkdat=gval
14730     Call bkdta
14740     Print using "8a6h6a4h":"Address:";address;" Data:";sutdat
14750     Goto gfin
14760 G8:   rem               "REG" DUMP THE SYSTEM REGISTERS
14770     Call getreg
14780     Call prntreg
14790     Goto gfin
14800 G9:   rem               HELP COMMAND...PRINT A LIST OF ALL LEGAL COMMANDS
14810     Print "-----------------------------------------"
14820     Print "           LEGAL COMMANDS"
14830     Print "           ----- --------"
14840     Print
14850     Print "GO"
14860     Print "HALT"
14870     Print "STEP [number_of_steps]"
14880     Print "DM [address [number_of_bytes]]"
14890     Print "OPEN address"
14900     Print "BA address    Break on Address compare"
14910     Print "BD data       Break on Data compare"
14920     Print "REG           Dump all 68000 registers"
14930     Print "INIT          Restart system under test"
14940     Print "EXIT          Leave test fixture program"
14950     Print "HELP          This message"
14960     Print "nnn           Execute test number 'nnn'"
14970     Print "-----------------------------------------"
14980     Print
14990     Goto gfin
15000 G10:   rem               halt command
15010     Call stopit
15020     Goto gfin
15030     End


15100 Sub getnum(schr)
15110   !
15120   !     Find out if the user typed a parameter with the command.
15130   !     If so, return it in <GVAL>.
15140   !
15150     Gval=0
```

```
15160     If schr<>0 then gosub gtst
15170     I=1
15180 Gtloop:     if len(cmnd$)<i then goto gtout
15190     Ch$=seg$(cmnd$,i,1)
15200     C=asc(ch$)
15210     If c>=48 and c<=57 then goto gt1
15220     If c>=65 and c<=70 then goto gt2
15230     If c>=97 and c<=102 then goto gt3
15240     Goto gtout
15250 Gt1:     gval=gval*16+c-48
15260     I=i+1
15270     Goto gtloop
15280 Gt2:     gval=gval*16+c-55
15290     I=i+1
15300     Goto gtloop
15310 Gt3:     gval=gval*16+c-87
15320     I=i+1
15330     Goto gtloop
15340 Gtout:     if len(cmnd$)>i then cmnd$=seg$(cmnd$,i+1,len(cmnd$)-i) else cmnd$=
  **              " "
15350     Return
15360 Gtst:     i=pos(cmnd$," ",1)
15370     Cmnd$=seg$(cmnd$,i+1,len(cmnd$)-i)
15380     Return
15390     End


15500 Sub bhex(value,num)
15510   !
15520   !     Append a <NUM> digit hex number <VALUE> to the current
15530   !     contents of <LINE$>.
15540   !
15550     Onum=value
15560     For x=num to 1 step -1
15570       Digit=0
15580       Digit=int(onum/16^(x-1))
15590       Onum=onum-digit*16^(x-1)
15600 Bh1:        line$=line$&seg$(bhtxt$,digit+1,1)
15610       Next x
15620     Line$=line$&" "
15630     Return
15640     End


15700 Sub putpc
15710     Sutdat=int(gval/2^16)
15720     Call writew
15730     Sutdat=gval-sutdat*2^16
15740     Address=address+2
15750     Call writew
15760     Return
15770     End
```

File: tss2 01-NOV-84 09:12:23

```
15800 Sub absub
15810   !
15820   !      ABORT handler.   Simply branch back to the idle state.
15830   !
15840     Branch idle
15850     End


15900 Sub alers
15910   !
15920   !      Error handler for VAL function.  We didn't recognize anything
15930   !      so report it as an illegal command and go to the idle state.
15940   !
15950     Print "Illegal Command"
15960     Branch idle
15970     End


16000 Sub getreg
16010   !
16020   !      Get the 68000's registers from the communication RAM
16030   !
16040     Address=[300h]
16050     For l=1 to 16
16060       Call readw
16070       Reghbuf(l)=sutdat
16080       Address=address+2
16090       Call readw
16100       Address=address+2
16110       Reglbuf(l)=sutdat
16120       Regbuf(l)=reghbuf(l)*[10000h]+reglbuf(l)
16130       Next l
16140     Return
16150     End


16200 Sub prntreg
16210   !
16220   !      Display the Registers we loaded with <GETREG>.
16230   !
16240     Integer rn
16250     For l=1 to 16
16260       If l<9 then rt$="D" else rt$="A"
16270       If l<9 then rn=l-1 else rn=l-9
16280       Rt$=rt$&trim$(str$(rn))&":"
16290       Print using "3ax4h4h":rt$;reghbuf(l);reglbuf(l)
16300       Next l
16310     Return
16320     End

          Total characters in the file =   24436
```