June 1990
Order Number: 311870-001

iPSC®/2 and iPSC®/860
SYSTEM ACCEPTANCE TEST
USER'S GUIDE

Intel® Corporation

REV.	REVISION HISTORY	DATE
-001	Original Issue	06/90

RESTRICTED RIGHTS

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the rights in Technical Data and Computer Software clause at 52.227-7013. Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051.

PREFACE

This manual tells how to use the System Acceptance Test (SAT) to verify that your iPSC system is working properly.

NOTE

In this manual, the term "iPSC system" refers to both the iPSC/2 family of systems and the iPSC/860 family of systems.

This manual is intended for system administrators, field service representatives, and Intel manufacturing engineers.

ORGANIZATION

Chapter 1	Introduces the iPSC/2 and iPSC/860 System Acceptance Test (SAT) and tells how to invoke the SAT program.
Chapter 2	Tells how to use the SAT menu interface and shows a sample SAT session.
Chapter 3	Describes each test in the test suite, and describes the test configuration file.

APPLICABLE DOCUMENTS

For more information, refer to these iPSC, Intel and other manuals:

iPSC® System Manuals

iPSC®/2 and iPSC®/860 C Language Reference Manual
Describes the C compiler for the iPSC system.

iPSC®/2 and iPSC®/860 FORGE User's Guide
Tells how to use the FORGE tool set to analyze Fortran programs and to port them to a parallel machine.

iPSC®/2 and iPSC®/860 Fortran Language Reference Manual
Describes the Fortran compiler for the iPSC system.

iPSC®/2 and iPSC®/860 LOOCS User's Guide
Tells how to use the LOOCS (Large Out Of Core Solver) software to solve very large matrices.

iPSC®/2 and iPSC®/860 Math Libraries Reference Manual
Describes the math libraries available on the iPSC system.

iPSC®/2 and iPSC®/860 Programmer's Reference Manual
Describes iPSC system commands and system calls (both C and Fortran).

iPSC®/2 and iPSC®/860 System Administrator's Guide
Describes the system administration tasks related to operating and maintaining an iPSC system.

iPSC®/2 and iPSC®/860 User's Guide
Overviews the iPSC system, including hardware and software architectures. Tells how to develop and run programs.

iPSC®/2 and iPSC®/860 VME Interface Reference Manual
Describes the installation and development of software drivers for the VME Interface Adapter board.

iPSC®/2 Ada Program Development Guide
Describes and tells how to use the tools for developing Ada programs for the iPSC/2.

iPSC®/2 Ada Programmer's Reference Manual
Describes all Ada routines and commands for the iPSC/2 system.

iPSC®/2 DECON User's Guide
Tells how to use DECON, the iPSC/2 concurrent debugger.

iPSC®/2 DECON User's Guide RX Beta Change Notice
Describes RX Beta Changes to DECON commands.

iPSC®/2 Lisp Language Reference Manual
Describes the Lisp implementation and its extensions that run on the iPSC nodes.

iPSC®/2 Lisp Programmer's Reference Manual
Describes the iPSC/2 Lisp user-interface and the iPSC/2 Lisp concurrent constructs.

iPSC®/2 Simulator Manual
Tells how to use the iPSC/2 Simulator for software development.

iPSC®/2 VAST2 User's Guide
Tells how to use the iPSC/2-VX version of VAST2 software.

iPSC®/2-VX User's Guide
Describes development of programs for the iPSC/2-VX processing system.

iPSC®/860 Vector User's Guide
Tells how to use the iPSC/860 vector processing features and the iPSC/860 version of VAST2 software.

Intel® Manuals

Intel® NFS for System V/386 Programmer's Guide and Reference
Describes the NFS programming environment and tools.

Intel® NFS for System V/386 User's/System Administrator's Guide and Reference
Describes the NFS programming environment and provides user and system administration information.

Intel® TCP/IP for SYSTEM V/386 Administrator's Guide and Reference
Describes TCP/IP Network administration.

Intel® TCP/IP for SYSTEM V/386 Programmer's Guide and Reference Manual
Describes the TCP/IP Network programming environment and provides information on programming tools.

Intel® TCP/IP for SYSTEM V/386 User's Guide and Reference
Describes the TCP/IP Network programming environment and provides user information.

i860™ 64-Bit Microprocessor Assembler and Linker Reference Manual
Tells how to use the i860 assembler and linker.

i860™ 64-Bit Microprocessor Programmer's Reference Manual
Tells how to use the i860 microprocessor.

i860™ Microprocessor Vector Primitive Library Reference Manual
Provides a complete list of the vector primitives for VAST2 for RX nodes.

SYP301 Installation and User's Guide
Tells how to install and start the System Resource Manager. Also provides hardware technical data.

Other Manuals

C: A Reference Manual - Harbison and Steele
Describes the C programming language.

Common Lisp: The Language - Guy L. Steele Jr.
Describes the Lisp programming language.

Reference Manual For The Ada Programming Language - ANSI/MIL-STD-1815A-1983
Describes the Ada programming language.

The C Programming Language - Kernighan and Ritchie
Describes the C programming language.

UNIX System V Manual Set
Describes UNIX System V.

NOTATIONAL CONVENTIONS

This section describes the following notational conventions:

- Type style usage
- Examples in text
- Command syntax

Type Style Usage

This manual uses the following type style conventions:

- **Bold** type style is used for anything that must be entered exactly as shown, including:

  Command names (including absolute pathname versions)
  Switches, flags, and options
  System Call names
  Routine names (predefined *and* user-defined)
  Reserved words

- *Italic* type style is used for:

  Variables
  File names (including absolute pathname versions)
  Directory names
  Process names
  User names
  Emphasis

- ***Bold-italic*** type style is used for user input described in text (what you enter in response to some prompt).

- `Plain-Monospace` type style is used for:

  Computer output (prompts and messages)
  Code
  Error messages
  Values of variables

- ***`Bold-Italic-Monospace`*** type style is used for user input displayed in an example (what you enter in response to some prompt)

- **Bold-Monospace** type style is used for the names of keyboard keys (which are also enclosed in angle brackets). For example:

  <Alt> <Backspace>
  <Break> <Ctrl>
  <Delete> or **<Del> <Enter>**
  <Esc> <Tab>

  For a key that prints, the result of pressing the key is used. For example, **s** means press the key labeled **<S>**, and **S** means press and hold the **<Shift>** key while pressing the **<S>** key.

  A dash indicates that the key preceding the dash is to be held down *while* the key following the dash is pressed. This has the effect of producing a single keystroke (which is why there is only one set of angle brackets). For example:

  <Ctrl-S>
  <Ctrl-@>
  <Ctrl-Alt-Del>

  Note that there are several ways to reference some keys. For example, **S** and **<Shift-s>** mean the same thing. In such cases, the simplest method (**S**) is usually used.

Examples in Text

This manual uses `monospace` type style for examples.

In examples of interactive sessions, user input appears in ***bold-italic-monospace*** type style to distinguish it from computer output. For example:

```
# getcube -t20                                          (userinput)
getcube successful: cube type 2m8n0 allocated          (computer output)
```

Many examples contain annotations that describe specific parts of the example. These annotations (which are not part of the example code or session) appear in *italic* type style.

Command Syntax

You enter commands at the console. Commands have the following form:

```
command arguments
```

where:

command	Is a keyword (see following section).
arguments	Consist of zero or more optional or required terms. One kind of argument, called a switch, consists of a dash followed by one or more letters. Sometimes a name or value follows the switch.

Command syntax is represented in this manual as follows:

keywords	Keywords appear in bold type. You must use keywords exactly as written. Command names and switches are keywords.
variables	Variables appear in italic type. They represent arguments that you must supply (in the place of the italicized word) when you invoke the command. Do not enter the name of the italicized word itself.
[ ]	Optional command-line arguments appear within square brackets. If the optional argument is a keyword, it appears in bold type. If the argument represents a variable, it appears in italic type.
. . .	Ellipses (three periods in a row) following an argument indicate that you may repeat that argument.
|	A vertical bar separates two or more choices of which you may select only one. For example, **[-c** *cubename* **| -a]** indicates that may select the first option (**-c** *cubename*) or the second option (**-a**).

The following command syntax for the **load** command shows all of the syntax elements:

```
load [ -c cubename | -a ] [ -p pid ] [ -H ] [ node... ]
   filename [ arguments... ]
```

According to the syntax, only the word **load** and a *filename* are required. However, five options are available: four options require a switch, and two options (*node* and *arguments*), allow more than one argument of this kind. The following example of this command uses the *cubename* option, two *node* values, an input file name "input_file," and an *argument* value of 1000 (this example does not use the **-p** and **-H** options):

```
load -c alpha 3 4 input_file 1000
```

TABLE OF CONTENTS

CHAPTER 3
TEST SUITE

INTRODUCTION 1

This chapter introduces the iPSC System Acceptance Test (SAT) program, and tells how to invoke the SAT program.

In this manual, the term "iPSC system" refers to both the iPSC/2 family of systems and the iPSC/860 family of systems. These systems contain two kinds of nodes:

- CX nodes are based on the 386™ microprocessor
- RX nodes are based on the i860™ microprocessor

The SAT program tests both kinds of nodes.

WHAT IS THE SAT PROGRAM?

The SAT program provides a menu-driven, system-level test-suite that lets you verify the proper functioning of your iPSC system.

The SAT program takes a "top-down" approach to the system, assuming that the entire system works, and running tests to prove that assumption. If these tests pass, then the hardware and software are assumed to be acceptable. If these tests fail, then your system needs to be examined by a qualified Intel Customer Support representative.

The SAT program complements the Cube Diagnostic Program (CDP), which takes a "bottom-up" approach. CDP assumes that nothing works, and tests each piece of the system before relying on it in subsequent tests.

The SAT program offers these advantages over CDP:

- **Anyone can run the SAT program.** You do not need to know a lot about your system's hardware, and you do not need to have superuser access.
- **The SAT program runs on the user level.** The SAT program tests the user-level software interface (as well as the hardware). CDP tests only the low-level hardware. It does not use the normal user software interface.
- **The SAT program can run in any size cube.** This lets you test specific nodes without interrupting other users. (CDP requires the entire system.)

The tests that the SAT program uses come from user algorithms that have shown weaknesses missed by standard diagnostic programs. These algorithms (which include random message patterns, heavy I/O usage, and 3D FFT calculations) test the communications, Concurrent I/O, and floating-point operations of the iPSC system.

Note that these tests do not find the cause of an error. They only indicate that a problem exists. If problems are found, please contact iSC Customer Support.

INVOKING THE SAT PROGRAM

Before you invoke the SAT program, use the **getcube** command to get a cube. The SAT program lets you test a cube of any size.

When you invoke the SAT program, it does the following:

1. Prints an identifying header.
2. Probes the attached cube to see what hardware is available. A command-line option lets you suppress the hardware probe.
3. Reads a test configuration file to determine which tests are available. Using the information returned by the probe, the SAT program disables all tests for which no hardware is available. The default test configuration files (*satconf.cx* for CX nodes and *satconf.rx* for RX nodes) reside in the directory */usr/ipsc/diag/satbin*.
4. Displays the Main Menu, and prompts you to select a menu item. Chapter 2 describes each menu item.

The SAT Command

The SAT command has the following syntax;

sat [**-f** *config_file*] [**-F**] [**-h**] [**-l** *log_file*] [**-L**] [**-P**]

where:

-f *config_file*	Tells the SAT program to use the file *config_file* as the test configuration file. (For information on creating a custom test configuration file, refer to Chapter 3.)
-F	Suppresses reading of any test configuration file at startup.
-h	Prints the usage message and exits.
-l *log_file*	Tells the SAT program to use *log_file* as the log file instead of the default log file *sat_log*.
-L	Suppresses error logging.
-P	Suppresses the hardware probe at startup.

Running the SAT Program With All Default Options

The following procedure runs the SAT program with all its default options:

1. Make sure that the cube software is running. If it is not, use the following command to start the proper daemons:

   ```
   rebootcube
   ```

2. Get a cube for the SAT program to test:

   ```
   getcube
   ```

3. Start the System Acceptance Test:

   ```
   sat
   ```

4. When the menu appears, press **<Return>** three times to select the defaults and start the test suite.

Assuming that the SAT program finds no errors:

- Systems without a Concurrent File System™ (CFS) require about 110 minutes to complete all standard tests.
- Systems with CFS require about 140 minutes to complete all standard tests.

MENU INTERFACE 2

The SAT menu interface is how you interact with the SAT program. This chapter tells how to use the SAT menu interface.

OVERVIEW

Each menu item is either an action or another menu:

- If you select an action, the SAT program performs that action and then displays the menu again.
- If you select another menu, the SAT program displays that menu and prompts you to make a selection.

Each menu item is preceded by a number. To select a menu item, enter its number.

The first item in each menu returns you to the previous level:

- The first item in the Main Menu returns you to the invoking shell (i.e., it exits the SAT program).
- The first item in the other three menus (Manage Test Configuration Menu, Manage Log File Menu, and Run Tests Menu) returns you to the Main Menu.

The second item in each menu shows help text for that menu.

Each menu prompt indicates a default selection (enclosed in square brackets). Pressing the **<Return>** key selects the indicated default.

THE MAIN MENU

The SAT program's Main Menu looks like this:

```
Main Menu
   0. Return to UNIX
   1. Show Help
   2. Manage Test Configuration
   3. Manage Log File
   4. Enter Shell
   5. Run Tests

Enter Selection [5] ->
```

where:

`Return to UNIX`
: Terminates the SAT program and returns you to your invoking shell.

`Show Help`
: Shows the help text for the Main Menu.

`Manage Test Configuration`
: Selects the Manage Test Configuration Menu. This menu lets you control which tests are available.

`Manage Log File`
: Selects the Manage Log File Menu. This menu lets you control what is logged and where the log is stored.

`Enter Shell`
: Spawns an interactive shell. Uses */bin/sh* or the shell identified by the *SHELL* environment variable. When you terminate the interactive shell, the SAT program displays the Main Menu again.

`Run Tests`
: Selects the Run Tests Menu. This menu lets you run one or all available tests.

THE MANAGE TEST CONFIGURATION MENU

The Manage Test Configuration Menu looks like this:

```
Manage Test Configuration
          Name        Hardware MaxTime RunTime Errors Comment
   •
   •                                    (Current test configuration appears here)
   •

   0. Return to Main Menu
   1. Show Help
   2. Show Current Test Configuration
   3. Modify MaxTime
   4. Probe Cube
   5. Load New Test Configuration
   6. Save Current Test Configuration

Enter Selection [0] ->
```

where:

`Return to Main Menu`
Returns you to the Main Menu.

`Show Help` Shows the help text for the Manage Test Configuration Menu.

`Show Current Test Configuration`
Shows a table of the tests that are currently available. This is the same table that appears immediately before the Manage Test Configuration Menu selections. The table shows the following information:

- The name of the test.
- The hardware to be tested.
- The maximum running time (in minutes) for the test. When this time limit is exceeded, the SAT program terminates the test. To change this value explicitly, select the "Modify MaxTime" menu item. Setting this time limit to zero disables the test. To disable all tests for which the attached cube has no hardware, select the "Probe Cube" menu item.
- The total time that the test has run during this SAT session.
- The total number of errors seen by the test during this SAT session.
- A brief description of the test.

`Modify MaxTime`
Changes a test's *MaxTime* value.

`Probe Cube` Determines what hardware is in the attached cube, and disables any tests for which no hardware is available.

`Load New Test Configuration`
Prompts you to enter the name of a test configuration file, and then replaces the current test table with the contents of that file.

`Save Current Test Configuration`
Prompts you to enter the name of a test configuration file, and then writes the current test table into that file. If the specified file exists, the SAT program prompts you for permission to overwrite the file.

THE MANAGE LOG FILE MENU

The Manage Log File Menu looks like this:

```
Manage Log File
        logging_message
   0. Return to Main Menu
   1. Show Help
   2. Show Log
   3. Clear Log
   4. Start/Stop Logging to Log file
   5. Change Log File

Enter Selection [0] ->
```

where:

`logging_message`
Identifies the current logging file, or tells you that logging is disabled.

`Return to Main Menu`
Returns you to the Main Menu.

`Show Help` Shows help text for the Manage Log File Menu.

`Show Log` Displays the contents of the current log file. Uses **more** or the utility identified by the *PAGER* environment variable.

`Clear Log` Empties the current log file.

`Start/Stop Logging to Log File`
Turns logging on and off.

`Change Log File`
Prompts you to enter the name of a log file, and then makes that file the new current log file. This item lets you save different SAT runs in different log files.

THE RUN TESTS MENU

The Run Tests Menu looks like this:

```
Run Tests
   0. Return to Main Menu
   1. Show Help
   2. Name_1
   3. Name_2
   4. Name_3
   .
   .
   .
   N+1. Name_N
   N+2. Run All Tests (approx. nn minutes)

Enter Selection [N+2] ->
```

where:

`Return to Main Menu`
Returns you to the Main Menu.

`Show Help` Shows the help text for the Run Tests Menu.

`Name_1 ... Name_N`
Are the names of the tests that are available to run. To run a specific test, enter the number of that test. The SAT program then prompts for the number of times to run the test.

`Run All Tests (approx. nn minutes)`
Prompts you to enter the number of times to run the tests, and then runs all tests in the order shown in the menu.

You can stop the test run at any time by pressing the interrupt key.

SAMPLE SAT SESSION

This section shows a sample SAT session. For this discussion, the session is separated into logical blocks, and each block begins with text that describes what the block does.

You may find it helpful to actually run the SAT program while going through this sample session. Doing so will give you an opportunity to try the other menu items that this sample session does not use.

This sample session takes 20-30 minutes.

Getting a Cube

Before you can begin a SAT session, you must first get a cube. Trying to run the SAT program with no cube attached produces an error message.

This sample session requests the largest available cube consisting of nodes with a 387 coprocessor installed. However, you can run the SAT program with a cube of any size and type.

```
% getcube -tf
getcube successful: cube type 16m8cxn16 allocated
```

Invoking the SAT Program

Now that a cube is attached, you can start the SAT program. When invoked, the SAT program prints an identifying header that shows the version, probes the attached cube to find out what hardware is available, loads the test configuration file, displays the Main Menu, and prompts you to enter a selection.

This sample session invokes the SAT program using the default options.

```
% sat

iSAT - iPSC System Acceptance Test, v1.0
Probing hardware configuration ...
        Detected CX nodes with
            387
            CFS

Reading test configuration file /usr/ipsc/diag/satbin/satconf.cx ...
        5 tests listed in configuration file.

Main Menu
   0. Return to UNIX
   1. Show Help
   2. Manage Test Configuration
   3. Manage Log File
   4. Enter Shell
   5. Run Tests

Enter Selection [5] ->
```

The "Enter Selection" prompt of all menus contains a number in square brackets. This number corresponds to that menu's default selection. For the main menu, the default selection is item number 5, the "Run Tests" menu. To select a default menu item, press the **<Return>** key. Otherwise, enter the number of the item that you want.

Getting Help

Menu item number 1 on each menu is "Show Help." This item provides additional information about the menu selections on the current menu.

The SAT program uses the UNIX **more** command to display the help text. This ensures that you'll be able to see all the text when there is more than will fit on one screen.

After displaying the help text, the Main Menu reappears, and you are again prompted to enter a selection.

This sample session selects item number 1, which shows the help information for the Main Menu.

```
Enter Selection [5] -> 1

Main Menu Help

Return to UNIX
        Terminates the SAT program and returns you to your invoking shell.

Show Help
        Shows the help text for the Main Menu.

Manage Test Configuration
        Selects the Manage Test Configuration Menu.  This menu lets you
        control which tests are available.

Manage Log File
        Selects the Manage Log File Menu.  This menu lets you control
        what is logged and where the log is stored.

Enter Shell
        Spawns an interactive shell.  Uses /bin/sh or the shell
        identified by the SHELL environment variable.  When you
        terminate the interactive shell, SAT displays the Main Menu
        again.

Run Tests
        Selects the Run Tests Menu.  This menu lets you run one or all
        available tests.

SAT Command Line Options
        f config_file   Specify a test configuration file
        h               Print the usage message and exit
        l log_file      Specify a log file
        F               Don't read the test configuration file at startup
        L               Turn off error logging
        P               Don't do a hardware probe at startup


Main Menu
   0. Return to UNIX
   1. Show Help
   2. Manage Test Configuration
   3. Manage Log File
   4. Enter Shell
   5. Run Tests

Enter Selection [5] ->
```

Modifying the Test Configuration

Before running the tests, you can use the "Manage Test Configuration" menu to see what tests are available or to modify one or more of the tests. When you select this menu, a table appears immediately before the menu itself. This table shows information about the current test suite.

```
Enter Selection [5] -> 2

Manage Test Configuration
          Name          Hardware  MaxTime RunTime Errors Comment
   1. Msgsize               HOST     30       0      0   host-node comm
   2. CFT-ncft               CFS     30       0      0   node-ionode general
   3. Async                 NODE     40       0      0   node-to-node comm
   4. Rand-ih               NODE     30       0      0   random isends, hrecvs
   5. 3d-fftcx               387     10       0      0   numerics and comm
        Totals:                     140       0      0

   0. Return to Main Menu
   1. Show Help
   2. Show Current Test Configuration
   3. Modify MaxTime
   4. Probe Cube
   5. Load New Test Configuration
   6. Save Current Test Configuration

Enter Selection [0] ->
```

LOADING A DIFFERENT TEST CONFIGURATION FILE

If you don't like the current test suite, you can load a different test configuration file by selecting menu item number 5.

This sample session loads the file *tests.cx*, a test configuration file that makes all CX tests available. (Chapter 3 tells how to create custom test configuration files that are based on the *tests.cx* and *tests.rx* files.)

```
Enter Selection [0] -> 5

Set configuration to file [/usr/ipsc/diag/satbin/satconf.cx] -> /usr/ipsc/diag/
satbin/tests.cx

Reading test configuration file /usr/ipsc/diag/satbin/tests.cx ...
        15 tests listed in configuration file.

Manage Test Configuration
          Name          Hardware  MaxTime RunTime Errors Comment
   1. Msgsize               HOST     30       0      0   host-node comm
   2. Gssize                NODE     30       0      0   node-node comm
   3. CFT-ncft               CFS     30       0      0   node-ionode general
   4. CFT-rc                 CFS     10       0      0   read cache
   5. CFT-rd                 CFS     30       0      0   read disk
   6. CFT-wrc                CFS     10       0      0   write/read cache
   7. CFT-wrd                CFS     30       0      0   write/read disk
   8. Async                 NODE     40       0      0   node-to-node comm
   9. Rand-c                NODE     30       0      0   random csends
  10. Rand-i                NODE     30       0      0   random isends
  11. Rand-h                NODE     30       0      0   random hsends
  12. Rand-ih               NODE     30       0      0   random isends, hrecvs
  13. 3d-fftcx               387     10       0      0   numerics and comm
  14. 3d-fftsx                SX     10       0      0   numerics and comm
  15. 3d-fftvx                VX     10       0      0   numerics and comm
        Totals:                     360       0      0

   0. Return to Main Menu
   1. Show Help
   2. Show Current Test Configuration
   3. Modify MaxTime
   4. Probe Cube
   5. Load New Test Configuration
   6. Save Current Test Configuration

Enter Selection [0] ->
```

PROBING THE CUBE

After loading a new test configuration file, you should always select item number 4, which probes the attached cube and updates the current configuration according to the current cube. In this sample session, the probe disables tests number 14 and 15 because the attached cube does not include any SX or VX nodes.

```
Enter Selection [0] -> 4
Probing hardware configuration ...
        Detected CX nodes with
            387
            CFS

Manage Test Configuration
          Name          Hardware  MaxTime RunTime Errors Comment
   1. Msgsize               HOST       30       0      0   host-node comm
   2. Gssize                NODE       30       0      0   node-node comm
   3. CFT-ncft               CFS       30       0      0   node-ionode general
   4. CFT-rc                 CFS       10       0      0   read cache
   5. CFT-rd                 CFS       30       0      0   read disk
   6. CFT-wrc                CFS       10       0      0   write/read cache
   7. CFT-wrd                CFS       30       0      0   write/read disk
   8. Async                 NODE       40       0      0   node-to-node comm
   9. Rand-c                NODE       30       0      0   random csends
  10. Rand-i                NODE       30       0      0   random isends
  11. Rand-h                NODE       30       0      0   random hsends
  12. Rand-ih               NODE       30       0      0   random isends, hrecvs
  13. 3d-fftcx               387       10       0      0   numerics and comm
  14. 3d-fftsx                SX disabled       0      0   numerics and comm
  15. 3d-fftvx                VX disabled       0      0   numerics and comm
        Totals:                        340       0      0

   0. Return to Main Menu
   1. Show Help
   2. Show Current Test Configuration
   3. Modify MaxTime
   4. Probe Cube
   5. Load New Test Configuration
   6. Save Current Test Configuration

Enter Selection [0] ->
```

RELOADING THE DEFAULT TEST CONFIGURATION FILE

For the sake of brevity, this sample session reloads *satconf.cx*, the default test configuration file.

```
Enter Selection [0] -> 5

Set configuration to file [/usr/ipsc/diag/satbin/tests.cx] -> /usr/ipsc/diag/
satbin/satconf.cx

Reading test configuration file /usr/ipsc/diag/satbin/satconf.cx ...
        5 tests listed in configuration file.

Manage Test Configuration
          Name          Hardware  MaxTime RunTime Errors Comment
   1. Msgsize               HOST     30       0      0   host-node comm
   2. CFT-ncft               CFS     30       0      0   node-ionode general
   3. Async                 NODE     40       0      0   node-to-node comm
   4. Rand-ih               NODE     30       0      0   random isends, hrecvs
   5. 3d-fftcx               387     10       0      0   numerics and comm
        Totals:                     140       0      0

   0. Return to Main Menu
   1. Show Help
   2. Show Current Test Configuration
   3. Modify MaxTime
   4. Probe Cube
   5. Load New Test Configuration
   6. Save Current Test Configuration

Enter Selection [0] ->
```

Normally, you would want to select item number 4 at this point. However, the previous probe revealed that the attached cube can handle all of the default tests. Therefore, another probe is not needed here (although it wouldn't hurt to do one just to be sure).

MODIFYING EACH TEST'S MAXIMUM RUN TIME

Also for the sake of brevity, this sample session reduces each test's maximum run time to one minute. Note, however, that reducing the maximum run time below the default value means that all nodes in the cube will not be tested. In practice, when you change a test's run time, you'll generally be *increasing* the value (not decreasing it) or setting the value to zero (to disable the test entirely).

```
Enter Selection [0] -> 3

Modify run time for test # -> 1

New run time (in minutes) for Msgsize test [30] -> 1

Manage Test Configuration
          Name         Hardware  MaxTime RunTime Errors Comment
   1. Msgsize              HOST       1       0      0   host-node comm
   2. CFT-ncft              CFS      30       0      0   node-ionode general
   3. Async                NODE      30       0      0   node-to-node comm
   4. Rand-ih              NODE      30       0      0   random isends, hrecvs
   5. 3d-fftcx              387      10       0      0   numerics and comm
        Totals:                      43       0      0

   0. Return to Main Menu
   1. Show Help
   2. Show Current Test Configuration
   3. Modify MaxTime
   4. Probe Cube
   5. Load New Test Configuration
   6. Save Current Test Configuration

Enter Selection [0] -> 3

Modify run time for test # -> 2

New run time (in minutes) for CFT-ncft test [30] -> 1

        .
        .                                          (Shows the updated test table and the menu)
        .

Enter Selection [0] -> 3

Modify run time for test # -> 3

New run time (in minutes) for Async test [40] -> 1
```

```
        •
        •                                          (Shows the updated test table and the menu)
        •

Enter Selection [0] -> 3

Modify run time for test # -> 4

New run time (in minutes) for Rand-ih test [30] -> 1

        •
        •                                          (Shows the updated test table and the menu)
        •

Enter Selection [0] -> 3

Modify run time for test # -> 5

New run time (in minutes) for 3d-fftcx test [10] -> 1

Manage Test Configuration
          Name         Hardware  MaxTime RunTime Errors Comment
   1. Msgsize             HOST       1       0      0   host-node comm
   2. CFT-ncft             CFS       1       0      0   node-ionode general
   3. Async               NODE       1       0      0   node-to-node comm
   4. Rand-ih             NODE       1       0      0   random isends, hrecvs
   5. 3d-fftcx             387       1       0      0   numerics and comm
        Totals:                       5       0      0

   0. Return to Main Menu
   1. Show Help
   2. Show Current Test Configuration
   3. Modify MaxTime
   4. Probe Cube
   5. Load New Test Configuration
   6. Save Current Test Configuration

Enter Selection [0] ->
```

Logging the Session

By default, all test output is saved (logged) in the file *sat_log*. Manage Log file Menu item number 5 lets you name a different log file, which is useful when you want to separate the results of different test runs. If you name a file that does not exist, the SAT program creates it. If you name a file that already exists, the SAT program appends the log to that file.

This sample session saves its log in the file *demo.log*.

First, return to the Main Menu (the default selection for the Manage Test Configuration Menu) and select the Manage Log File Menu (Main Menu item number 3).

```
Enter Selection [0] -> <Return>

Main Menu
   0. Return to UNIX
   1. Show Help
   2. Manage Test Configuration
   3. Manage Log File
   4. Enter Shell
   5. Run Tests

Enter Selection [5] -> 3

Manage Log File
        Logging to file sat_log.
   0. Return to Main Menu
   1. Show Help
   2. Show Log
   3. Clear Log
   4. Start/Stop Logging to Log File
   5. Change Log File

Enter Selection [0] ->
```

Next, select item number 5 and enter "demo.log" as the name of the log file.

```
Enter Selection [0] -> 5
Redirect logging to [sat_log] -> demo.log

Manage Log File
        Logging to file demo.log.
   0. Return to Main Menu
   1. Show Help
   2. Show Log
   3. Clear Log
   4. Start/Stop Logging to Log File
   5. Change Log File

Enter Selection [0] ->
```

Running the Test Suite

Having named a new log file, return to the Main Menu (the default selection for the Manage Log File Menu) and select the Run Tests Menu (the default selection for the Main Menu).

```
Enter Selection [0] -> <Return>

Main Menu
   0. Return to UNIX
   1. Show Help
   2. Manage Test Configuration
   3. Manage Log File
   4. Enter Shell
   5. Run Tests

Enter Selection [5] -> <Return>

System Acceptance Tests
   0. Return to Main Menu
   1. Help
   2. Msgsize Test
   3. CFT-ncft Test
   4. Async Test
   5. Rand-ih Test
   6. 3d-fftcx Test
   7. Run All Tests (approx. 5 minutes)

Enter Selection [7] ->
```

RUNNING ALL AVAILABLE TESTS

To run all available tests (the default selection for the Run Tests Menu) press the **<Return>** key.

```
Enter Selection [7] -> <Return>

Running System Acceptance Tests

How many cycles? (0 for continuous; q for menu) [1] -> <Return>
Starting logging for this run...Done

 -@- 05/10/90 09:27:17  Starting SAT cycle 1 of 1
 -@- 05/10/90 09:27:17  Executing Msgsize                    (The first test)
5/10/90 09:27:21 NODE: 16 PID: 64 : Start MSGSIZE.H test.
NODE: 16 PID: 64 : Message size = 33000, cycles = 100
5/10/90 09:27:27 NODE: 16 PID: 64 : On cycle 1, msg size 0
 -@- msgsize: PASSED -@-
 -@- 05/10/90 09:28:18  Test done, 0 errors.

 -@- 05/10/90 09:28:21  Executing CFT-ncft                   (The second test)
Loading node program "ncft.cx" Done
Cycle 5                                    (Node messages overwrite each other)
HCFT: Abort in progress, please wait.......                  (Test timed out)
 -@- hcft: PASSED -@-
 -@- 05/10/90 09:29:22  Test done, 0 errors.

 -@- 05/10/90 09:29:25  Executing Async                      (The third test)
NODE: 0 : Ready to start sending 100000 length messages
 -@- ASYNC async.cx: PASSED -@-
 -@- 05/10/90 09:30:26  Test done, 0 errors.

 -@- 05/10/90 09:30:32  Executing Rand-ih                    (The fourth test)
Message 0, seed 00000001, Thu May 10 09:30:35 1990
RAND: Random Message Exchange test, using ihrxnode.cx
 -@- Rand ihrxnode.cx: PASSED -@-
 -@- 05/10/90 09:31:33  Test done, 0 errors.
```

```
 -@- 05/10/90 09:31:39  Executing 3d-fftcx                    (The last test)
FFT test, using fft3d.cx
 -@- ffthost fft3d.cx: PASSED -@-
          Name         Hardware  MaxTime RunTime Errors Comment
   1. Msgsize              HOST       1       1      0   host-node comm
   2. CFT-ncft              CFS       1       1      0   node-ionode general
   3. Async                NODE       1       1      0   node-to-node comm
   4. Rand-ih              NODE       1       1      0   random isends, hrecvs
   5. 3d-fftcx              387       1       1      0   numerics and comm
        Totals:                       5       5      0

 -@-    SAT run finished, 0 errors seen in this run.


 -@-            P A S S E D

System Acceptance Tests
   0. Return to Main Menu
   1. Help
   2. Msgsize Test
   3. CFT-ncft Test
   4. Async Test
   5. Rand-ih Test
   6. 3d-fftcx Test
   7. Run All Tests (approx. 5 minutes)

Enter Selection [7] ->
```

RUNNING INDIVIDUAL TESTS AND STOPPING A TEST

To run only one of the available tests, enter the number of the desired test. To stop a test, press the interrupt key.

This sample session starts the Msgsize test, and then stops it.

```
Enter Selection [7] -> 2

Running System Acceptance Test Msgsize

How many cycles? (0 for continuous; q for menu) [1] -> 0
Starting logging for this run...
Done

 -@- 05/10/90 09:37:44  Starting SAT cycle 1 of continuous run.
 -@- 05/10/90 09:37:44  Executing Msgsize
5/10/90 09:37:46 NODE: 16 PID: 64 : Start MSGSIZE.H test.
NODE: 16 PID: 64 : Message size = 33000, cycles = 100
5/10/90 09:37:52 NODE: 16 PID: 64 : On cycle 1, msg size 0
 -@- SAT Interrupted, stand by...
 -@- 03/09/90 09:56:10  Test interrupted by user, 0 errors.
          Name         Hardware  MaxTime RunTime Errors Comment
   1. Msgsize             HOST       1       2      0   host-node comm
   2. CFT-ncft             CFS       1       1      0   node-ionode general
   3. Async               NODE       1       1      0   node-to-node comm
   4. Rand-ih             NODE       1       1      0   random isends, hrecvs
   5. 3d-fftcx             387       1       1      0   numerics and comm
        Totals:                      5       6      0

 -@-    SAT run finished, 0 errors seen in this run.


 -@-             P A S S E D

System Acceptance Tests
   0. Return to Main Menu
   1. Help
   2. Msgsize Test
   3. CFT-ncft Test
   4. Async Test
   5. Rand-ih Test
   6. 3d-fftcx Test
   7. Run All Tests (approx. 5 minutes)

Enter Selection [7] ->
```

Exiting the SAT Program

Return to the Main Menu and exit the SAT program.

```
Enter Selection [7] -> 0

Main Menu
   0. Return to UNIX
   1. Show Help
   2. Manage Test Configuration
   3. Manage Log File
   4. Enter Shell
   5. Run Tests

Enter Selection [5] -> 0

Exiting SAT ...
```

This concludes the sample SAT session.

TEST SUITE 3

The test suite used by the SAT program consists of tests that exercise different parts of the iPSC system.

This chapter describes each test, its arguments, and its error messages.

This chapter also describes the format of the test configuration file, shows the contents of the two default files, and tells how to create custom files.

THE TESTS

The SAT program uses the following tests (located in */usr/ipsc/diag/satbin*):

Host Global Send Test (**msgsize**)
: Exercises global host-to-node and node-to-host communication (message-passing).

Node Global Send Test (**gssize**)
: Exercises global node-to-node communication.

Concurrent File System™ Tests (**hcft**)
: Exercise the Concurrent File System.

Asynchronous Message-Passing Test (**asynchost**)
: Exercises discrete node-to-node communication (by sending large messages in order between all nodes).

Random Message-Passing Tests (**randhost**)
: Exercise discrete node-to-node communication (by sending random-sized messages to random nodes).

3D-FFT Tests (**ffthost**)
: Exercise node-to-node communication, memory, and numeric hardware on each node.

Host Global Send Test

The Host Global Send Test exercises the system's host-to-node and node-to-host message-passing capabilities. This test verifies the long paths from the host to the last nodes in the cube.

DESCRIPTION

In this test:

1. The host program sends a short message to all nodes (using a global send).
2. Each node receives the message, verifies its contents and size, and returns it to the host.
3. The host verifies that the message returned by each node hasn't changed, and then sends a slightly larger message to all nodes.

This cycle continues until the maximum message size is reached.

SYNTAX

The **msgsize** command invokes the Host Global Send Test:

msgsize [**-S**] [**-m** *msg_size*] [**-r**] [*reps*]

where:

-S	Suppresses all **getcube()** and **relcube()** calls. This argument is required when running under the SAT program, which performs all necessary cube control.
-m *msg_size*	Specifies the maximum message size in bytes. The default value for *msg_size* is 32K bytes.
[**-r**]	Specifies the RX version of the test. If this option is omitted, then the CX version is used.
reps	Specifies the number of times the test will repeat. The default value for *reps* is 10.

For example, the following invocation line suppresses all **getcube()** and **relcube()** calls, sets the maximum message size to 33,000 bytes, and sets the number of repetitions to 100:

```
msgsize -S -m 33000 100
```

ERROR MESSAGES

The Host Global Send Test may generate the following error messages:

`incorrect infocount`	The length of the message received was incorrect.
`msg incorrect`	The contents of the received message were incorrect.
`buffer bad beyond msg end`	The message received corrupted data beyond the end of the receiving buffer.

Node Global Send Test

The Node Global Send Test is similar to the Host Global Send Test, except that this test exercises only the nodes.

DESCRIPTION

In this test:

1. One node sends a message to all other nodes (using a global send).
2. Each node receives the message, verifies its contents and size, and returns it to the originating node.
3. The originating node verifies that the message returned by the other nodes hasn't changed, and then sends a slightly larger message to all nodes.

This cycle repeats until the maximum message size is reached, at which time the next node becomes the originating node, and so on. When all nodes have served as the originating node, the test is complete.

SYNTAX

The **gssize** command invokes the Node Global Send Test:

gssize [**-S**] [**-m** *msg_size*] [**-r**] [*reps*]

where:

-S	Suppresses all **getcube()** and **relcube()** calls. This argument is required when running under the SAT program, which performs all necessary cube control.

-m *msg_size*	Specifies the maximum message size in bytes. The default value for *msg_size* is 32K bytes.
[**-r**]	Specifies the RX version of the test. If this option is omitted, then the CX version is used.
reps	Specifies the number of times the test will repeat. The default value for *reps* is 10.

For example, the following invocation line suppresses all **getcube()** and **relcube()** calls, sets the maximum message size to 33,000 bytes, and sets the number of repetitions to 100:

```
gssize -S -m 33000 100
```

ERROR MESSAGES

The Node Global Send Test may generate the following error messages:

`incorrect infocount`	The length of the message received was incorrect.
`msg incorrect`	The contents of the received message were incorrect.
`buffer bad beyond msg end`	The message received corrupted data beyond the end of the receiving buffer.

Concurrent File System Tests

The Concurrent File System Test exercises the following components of the Concurrent File System (CFS):

- Hardware:

  I/O nodes
  SCSI bus
  Disks
  Connection to the I/O nodes

- Software:

  I/O library functions
  diskproc
  nameproc

DESCRIPTION

In this test, the host program loads the node programs and monitors their progress. Each node executes a loop of open, seek, write, read, close, and delete functions, operating on files in the Concurrent File System.

SYNTAX

The **hcft** command invokes the Concurrent File System Tests:

hcft [**-S**] [**-t** *cubetype*] [**-b** *buffer_size*] [**-m** *file_size*] [**-n** *version*]

where:

-S Suppresses all **getcube()** and **relcube()** calls. This argument is required when running under the SAT program, which performs all necessary cube control.

-t *buffer_size* Specifies the buffer size (in bytes) to use in file I/O. The default value for *buffer_size* is 4K bytes.

-m *file_size* Specifies the maximum file size (in bytes) to use in file I/O. The default value for *file_size* is 400K bytes.

-n *version* Specifies the test version to run:

`ncft.cx` (or `ncft.rx`)
Tests CX nodes (or RX nodes) using normal file usage patterns. If you don't specify a test, `ncft.cx` is assumed.

`rc.cx` (or `rc.rx`)
Tests CX nodes (or RX nodes) by creating a one-block file, writing it once, and repeatedly reading it. Exercises the I/O node block cache.

`wrc.cx` (or `wrc.rx`)
Tests CX nodes (or RX nodes) by creating a one-block file, and repeatedly writing and reading it. Exercises the I/O node block cache.

`rd.cx` (or `rd.rx`)
Tests CX nodes (or RX nodes) by creating a large file, writing it once, and repeatedly reading it. Exercises the SCSI bus and the disks.

`wrd.cx` (or `wrd.rx`)
: Tests CX nodes (or RX nodes) by creating a large file, and repeatedly writing and reading it from the disk. Exercises the SCSI bus and the disks.

For example, the following invocation line suppresses all **getcube()** and **relcube()** calls, and specifies the *rc.cx* test version:

```
hcft -S -n rc.cx
```

ERROR MESSAGES

The Concurrent File System Tests may generate the following error messages:

`mismatch error`	The data read from disk was incorrect.
`create error`	There was an error on creating a file.
`unlink error`	There was an error on unlinking a file.
`open error`	There was an error on opening a file.
`read error`	There was an error on reading a file.
`seek error`	There was an error on seeking a file.
`write error`	There was an error on writing a file.

Asynchronous Message-Passing Test

The Asynchronous Message-Passing Test is similar to the Node Global Send Test, except that this test sends many large messages.

DESCRIPTION

Each node sends a message to all of its neighbors while receiving messages from its neighbors. To keep the message traffic high, the messages are not verified.

SYNTAX

The **asynchost** command invokes the Asynchronous Message-Passing Test:

asynchost *version msg_size*

where:

version Specifies the test version to run:

`async.cx` Tests CX nodes.

`async.rx` Tests RX nodes.

msg_size Specifies the message size in bytes.

For example, the following invocation line specifies the *async.cx* test version and sets the maximum message size to 10,000 bytes:

```
asynchost async.cx 10000
```

ERROR MESSAGES

The Asynchronous Message-Passing Test may generate the following error messages:

`Haven't passed msgs in` *nn* `msecs` The node communication may be hung, or traffic may be so high that messages aren't getting through.

Random Message-Passing Tests

The Random Message-Passing Tests exercises node-to-node communications.

DESCRIPTION

This test sends random-sized messages to random nodes. The receiving node verifies that the message arrived at the correct node with the correct type, length, and data.

SYNTAX

The **randhost** command invokes the Random Message-Passing Tests:

randhost *version*

where:

version Specifies the test to run:

`irxnode.cx` (or `irxnode.rx`)
Tests CX nodes (or RX nodes) using **isend()** and **irecv()** calls.

`crxnode.cx` (or `crxnode.rx`)
Tests CX nodes (or RX nodes) using **csend()** and **crecv()** calls.

`hrxnode.cx` (or `hrxnode.rx`)
Tests CX nodes (or RX nodes) using **hsend()** and **hrecv()** calls.

`chrxnode.cx` (or `chrxnode.rx`)
Tests CX nodes (or RX nodes) using **csend()** and **hrecv()** calls.

`ihrxnode.cx` (or `ihrxnode.rx`)
Tests CX nodes (or RX nodes) using **isend()** and **hrecv()** calls.

For example, the following invocation line specifies the *crxnode.cx* test version:

```
randhost crxnode.cx
```

ERROR MESSAGES

The Random Message-Passing Tests may generate the following error messages:

`Duplicate mid xx` The node operating system returned a message identifier (mid) that was already in use.

`mid xx: Wrong type: a should be b` The node received a message whose type was not the same as was sent.

`mid xx: Wrong node: a should be b`
The node received a message that was sent to some other node.

`mid xx: Wrong count: a should be b`
The node received a message whose length was not the same as was sent.

`mid xx: Buffer overrun, a should be b`
The node received a message whose contents was not as expected.

`mid xx: n Mismatches from m, a should be b`
The data in a message from one node to another was corrupted.

3D-FFT Tests

The 3D-FFT Tests exercises node-to-node communication, memory, and the numeric hardware on each node.

DESCRIPTION

The 3D-FFT Tests run an actual application program. These tests verify that an application will run correctly.

In these tests, each node:

1. Performs a two-dimensional FFT on all of its randomly-generated data
2. Swaps its data with other nodes and calculates the third dimension of the three-dimensional FFT
3. Checks its work by doing inverse FFTs and comparing the result with the starting data.

The tests print an estimated MFLOP rating after running each example.

The file */usr/ipsc/diag/satbin/fft3d.dat* provides the sizes of the example matrices.

SYNTAX

The **ffthost** command invokes the 3D-FFT Tests:

ffthost *version*

where:

version Specifies the test version to run:

`fft3d.cx`	Tests CX nodes that have the standard 80387 coprocessor.
`fft3d.sx`	Tests CX nodes that have the optional SX processor.
`fft3d.vx`	Tests CX nodes that have the optional VX processor.
`fft3d.rx`	Tests RX nodes.

For example, the following invocation line specifies the *fft3d.cx* test version:

```
ffthost fft3d.cx
```

ERROR MESSAGES

The 3D-FFT Tests generate no error messages (unless the nodes do not have enough memory to run a specific example).

TEST CONFIGURATION FILE

To determine which tests to run during a session, the SAT program reads a "test configuration file." This file tells the SAT program about the tests that are available in a particular test suite (what the test exercises, how many times to run the test, how long to let the test run, and so on).

To add or delete tests from the test suite, you edit the test configuration file for that test suite.

This section describes the format of the test configuration file, shows the contents of the two default files, and tells how to create custom files.

Format of Test Configuration File

The test configuration file contains one line for each test in the test suite. Each line consists of five colon-delimited fields of the following form:

Name : *Command* : *Hardware* : *MaxTime* : *Description*

where:

Name Is the name by which the SAT program knows the test. This is the name that appears when you select the "Show Test Configuration" item of the Manage Test Configuration Menu, or the "Run Tests" item of the Main Menu.

Command Is the command line that executes the test. This is any valid Bourne Shell command line that does not use background processes or I/O redirection.

Hardware Identifies the hardware that the test covers. Valid values include the following:

`HOST`	A host interface test
`NODE`	A generic node test
`386NODE`	A CX-node test
`860NODE`	An RX-node test
`387`	80387 numeric coprocessor test
`SX`	Weitek numeric coprocessor test
`VX`	Vector numeric coprocessor test
`CFS`	Concurrent File System test

MaxTime Is the maximum running time for the test (in minutes). If a test is still running after this time, SAT terminates the test and starts the next test in the suite. Setting the run time to 0 minutes disables the test.

Description Is a brief description of the test.

For example, the following configuration entry identifies the Msgsize test for CX nodes:

```
Msgsize : msgsize -S -m 10000 100 : HOST : 30 : host-node comm
```

Error Checking

As the SAT program reads the test configuration file, it checks each entry. If an entry contains an empty field or the wrong number of fields (i.e., not five), SAT ignores the invalid entry, and prints the following warning message:

```
Invalid Configuration Entry: line n
```

where *n* is the line number of the invalid entry.

The Default CX Test Configuration File

When you invoke the SAT program (with no command-line options) for an iPSC/2 cube, it reads the default test configuration file (*/usr/ipsc/diag/satbin/satconf.cx*). This file contains entries for testing iPSC/2 cubes.

The *satconf.cx* file contains these configuration entries:

```
#
#   Test configuration file for iPSC/2 cubes
#

Msgsize  : msgsize -S -m 33000 100   : HOST : 30 : host-node comm
CFT-ncft : hcft -S                   : CFS  : 30 : node-ionode general
Async    : asynchost async.cx 100000 : NODE : 40 : node-to-node comm
Rand-irx : randhost irxnode.cx       : NODE : 30 : isend rx
3d-fftcx : ffthost fft3d.cx          : 387  : 10 : numerics & comm
```

The Default RX Test Configuration File

When you invoke the SAT program (with no command-line options) for an iPSC/860 cube, it reads the default test configuration file *satconf.rx*. This file contains entries for testing iPSC/860 cubes.

The *satconf.rx* file contains these configuration entries:

```
#
#   Test configuration file for iPSC/860 cubes
#

Msgsize  : msgsize -S -r -m 33000 100 : HOST    : 30 : host-node comm
CFT-ncft : hcft -S -n ncft.rx         : CFS     : 30 : node-ionode general
Async    : asynchost async.rx 10000   : NODE    : 40 : node-to-node comm
Rand-irx : randhost irxnode.rx        : NODE    : 30 : isend rx
3d-fftcx : ffthost fft3d.rx           : 860NODE : 10 : numerics & comm
```

Creating a Custom Test Configuration File

If you invoke the SAT program with the -f command-line option, it does not read either of the default test configuration files. Instead, it reads a test configuration file that you specify. This lets you create and use custom test configuration files that suit your particular needs.

To create a custom test configuration file:

1. Create a copy of the appropriate test suite file (*tests.cx* for an iPSC/2 system; *tests.rx* for an iPSC/860 system). For example, the following command-line copies the CX test suite file into a file called *mytests.cx*:

```
cp tests.cx mytests.cx
```

2. Edit the new test configuration file (*mytests.cx* in the previous example), removing all tests that you don't want and modifying any other tests as desired.

Now you can use your new test configuration file with the SAT program.

THE CX TEST SUITE FILE

The *tests.cx* file contains these configuration entries:

```
#
#All sat test configuration file for CX cubes
#

Msgsize    : msgsize -S -m 262144 100 : HOST : 30 : host-node comm
Gssize     : gssize  -S -m 262144 100 : NODE : 30 : node-node comm

CFT-ncft   : hcft -S                  : CFS  : 30 : node-ionode general
CFT-rc     : hcft -S -n  rc.cx        : CFS  : 10 : read cache
CFT-rd     : hcft -S -n  rd.cx        : CFS  : 30 : read disk
CFT-wrc    : hcft -S -n wrc.cx        : CFS  : 10 : write/read cache
CFT-wrd    : hcft -S -n wrd.cx        : CFS  : 30 : write/read disk

Async      : asynchost async.cx 10000 : NODE : 30 : node-to-node comm

Rand-crx   : randhost crxnode.cx      : NODE : 30 : csend rx
Rand-irx   : randhost irxnode.cx      : NODE : 30 : isend rx
Rand-hrx   : randhost hrxnode.cx      : NODE : 30 : hsend rx
Rand-chrx  : randhost chrxnode.cx     : NODE : 30 : chsend rx

3d-fftcx   : ffthost fft3d.cx         : 387  : 10 : numerics and comm
3d-fftsx   : ffthost fft3d.sx         : SX   : 10 : numerics and comm
3d-fftvx   : ffthost fft3d.vx         : VX   : 10 : numerics and comm
```

THE RX TEST SUITE FILE

The *tests.rx* file contains these configuration entries:

```
#
#All sat test configuration file for RX cubes
#

Msgsize    : msgsize -S -r -m 262144 100 : HOST : 30 : host-node comm
Gssize     : gssize  -S -r -m 262144 100 : NODE : 30 : node-node comm

CFT-ncft   : hcft -S -n ncft.rx          : CFS  : 30 : node-ionode general
CFT-rc     : hcft -S -n   rc.rx          : CFS  : 10 : read cache
CFT-rd     : hcft -S -n   rd.rx          : CFS  : 30 : read disk
CFT-wrc    : hcft -S -n  wrc.rx          : CFS  : 10 : write/read cache
CFT-wrd    : hcft -S -n  wrd.rx          : CFS  : 30 : write/read disk

Async      : asynchost async.rx 10000    : NODE : 30 : node-to-node comm

Rand-crx   : randhost crxnode.rx         : NODE : 30 : csend rx
Rand-irx   : randhost irxnode.rx         : NODE : 30 : isend rx
Rand-hrx   : randhost hrxnode.rx         : NODE : 30 : hsend rx
Rand-chrx  : randhost chrxnode.rx        : NODE : 30 : chsend rx

3d-fftcx   : ffthost fft3d.rx            : 860NODE : 10 : numerics and comm
```