April, 1989
Order Number: 311702

# iPSC®/2 SYSTEM SOFTWARE

# RELEASE 3.0

# PRODUCT RELEASE NOTES

intel Corporation

## RESTRICTED RIGHTS

# PREFACE

## PURPOSE/SCOPE

These are the release notes for the iPSC®/2 System. They provide the latest information on features, limitations and their workarounds, and installation procedures. These notes assume that you are an application programmer proficient in the C and Fortran languages and the UNIX operating system.

## ORGANIZATION

Chapter 1 describes features, limitations, and associated workarounds for your Release 3.0 iPSC/2 System Software.

Chapter 2 provides installation procedures for your UNIX, TCP/IP, and iPSC/2 System Software.

Chapter 3 provides suggestions for using your new Concurrent File System™.

Chapter 4 provides information on the Remote Host feature of your Release 3.0 iPSC/2 System Software.

Chapter 5 provides miscellaneous user information that will assist you in getting the most out of your Release 3.0 iPSC/2 System Software.

# APPLICABLE DOCUMENTS

For more information, refer to the *Release 3.0 Customer Letter* that accompanied your software.

# NOTATIONAL CONVENTIONS

The notational conventions described in this section are used throughout the release notes.

## Program Elements

Program elements, such as system calls, file names, directory names, and labels are printed in *italics* in the text.

## Interactive Computer Sessions

In the descriptions of interactive sessions, prompts/information displayed on your workstation monitor is printed in Titan 10-point type. For example:

```
File system "old cfs" already exists
```

User input to the system is printed in Titan 10-point bold type. For example:

```
File name?  myfile
```

## Workstation Keys

Workstation keys are described in text using the exact nomenclature printed on the key, surrounded by angle brackets. For example:

To enter text, press <Enter>.

When an instruction involves pressing more than one key simultaneously, all the key nomenclatures involved, separated by commas, are shown in angle brackets. For example:

To reboot the system, press <Ctrl, Alt, Del>.

## Input Entry

Unless otherwise directed, press <Enter> after all commands, and to enter user input.

# TABLE OF CONTENTS

## CHAPTER 1 FEATURES AND LIMITATIONS

# CHAPTER 2 SOFTWARE INSTALLATION



# CHAPTER 3 CONCURRENT FILE SYSTEM™



# CHAPTER 4 REMOTE HOST

# CHAPTER 5 USER INFORMATION

# FEATURES AND LIMITATIONS 1

## INTRODUCTION

### IMPORTANT NOTE

**Before you use your iPSC/2 System or attempt to update it, read all of these Product Release Notes.**

The iPSC/2 Product Release Notes explain the features, known limitations, and workarounds for Release 3.0 of the iPSC/2 system software. They provide procedures for installation of the software for the iPSC/2 System. The release notes also include an explanation of how to run iPSC/1 programs on the iPSC/2 System.

## NEW FEATURES

Release 3.0 of the iPSC/2 system software runs on all hardware configurations of the iPSC/2 System that include the PC586 Ethernet Controller. The following describes some of the new features available in this release:

### UNIX System V/386 Release 3.2 Version 2.1 for the SRM

1. **Better disk and tape support**
   Larger disks can now be supported on the SRM. You can now use the 2K-byte file block option to improve file access performance on any size disk. Bad sector and bad track handling are improved. The cartridge tape driver running under this version of UNIX software supports multiple volume read and write operations that require the issuing of fewer commands.

2. **Streams and sockets**
   This version of the UNIX software uses streams to implement a fully Berkeley-compatible socket interface. This interface is a key to supporting our NFS option.

3. **Improved security**
   This version of UNIX software provides improved security with such features as sticky bit protection of public directories, shadow passwords and security additions to such commands as *uucp*, *login*, *mail*, and *lp*.

4. **International character sets**
   This version of UNIX software now supports international character sets for such commands as *cat*, *ed*, and *ls* using full eight-bit characters.

5. **DOS hooks**
   This version of UNIX software supports DOS and DOS applications. You must have VP/ix software to run DOS under the UNIX operating system. Not all DOS applications are compatible with VP/ix. Check with your software supplier for details.

## New TCP/IP Software and Hardware

The new Lachman software replaces the Excelan TCP/IP software. This new software provides the following improvements:

1. **Runs faster than previous SRM system**
   The new TCP/IP software/hardware enables the local area network to run approximately 60 percent faster than the previous SRM system.

2. **Berkeley style TCP/IP supports NFS**
   The new software provides a full Berkeley-compatible sockets interface. This interface supports NFS.

## Message-Passing Extensions

1. **Enhanced message-passing library**
   The following commands have been added to the node message-passing library: *csendrecv*, *hsendrecv*, *isendrecv*, and *hsend*. The first three system calls enable programmers to send a message and receive a reply with one system call. The *hsend* call sends a message and provides a user-written handler on completion of the send.

2. **Improved remote procedure call mechanism**
   This mechanism supports node-node and node-host messages that cannot be interfered with by other message traffic over the Direct Connect Network, thus implementing a remote call mechanism.

## Concurrent File System™ (CFS) Option Support

1. **Transparent parallel disk access**
   High-speed parallel disk access to the Concurrent I/O System is fully transparent to the user. It is as easy to use as a traditional single-disk UNIX file system.

2. **Both SRM host files and CFS files can be accessed easily**
   You can do file copies between the SRM and CFS as easily as you can do any other UNIX file copy. For example, you can read information from the SRM cartridge tape into a CFS file with a one-line command. This is possible because you can see all the mass storage resources in the iPSC/2 system (both SRM host files and CFS files) simultaneously by running the node shell (*nsh*) command interpreter from a node rather than the SRM.

3. **New I/O routines added**
   CFS provides the following library of fast I/O routines that can be called by programs written in either Fortran or C: *cread, cwrite, iodone, iomode, iowait, iread, iseof, iwrite, lsize, restrictvol,* and *setiomode.*

4. **New I/O routines improve Fortran performance**
   Our *cread, iread, cwrite,* and *iwrite* I/O routines bypass some of the overhead associated with standard compiler-driven Fortran I/O. These routines support both synchronous and asynchronous operations. The *lseek* routine lets each node position its file pointer to a different part of a file. The *iseof* routine tests for the end-of-file (EOF).

5. **Four modes for file access**
   When multiple nodes need to access the same file in parallel, CFS provides the programmer with four modes of parallel access:

   - Mode 0: individual file pointer (IFP). This is the default mode.
   - Mode 1: common file pointer (CFP)
   - Mode 2: synchronized, CFP, variable length buffer
   - Mode 3: synchronized, CFP, fixed length buffer

   These modes provide varying degrees of synchronization to simplify programming.

## Network File System (NFS) Option Support

1. **Transparent file sharing over local area network (LAN)**
   Using NFS on a workstation (or SRM), you can mount file systems on other Ethernet nodes and access those files as if they were local files on your own workstation. This eliminates such time-consuming operations as explicit file copies and logins. It also allows programs to use remotely mounted files without modification.

2. **Easier to edit programs on workstation and compile them on SRM**
   NFS allows you to keep source and object files on the SRM disk and use your workstation text editor and other tools. Using NFS another way, you can keep your program files on your workstation and still compile and load programs on the SRM. This enables you to conserve storage space on the SRM disk.

3. **New LAN software and hardware**
   iSC's version of NFS runs under UNIX System V/386 Release 3.2 Version 2.1 software. The new TCP/IP software provides a full Berkeley-compatible sockets interface. This interface supports NFS. The PC586 Ethernet Controller provides a full-function IEEE 802.3 Ethernet capability over an Ethernet cable. The NFS

option supports the SRM's local disk. Release 3.0 software does not support NFS on CFS. iSC plans to support NFS on CFS later.

### SRM Large Disk Option Support

1. **Allows replacement of a 140M-byte SRM disk with a 380M-byte disk**
   Some iPSC/2 systems with very large programs or a large number of users may require more SRM disk space. Replacing the SRM disk with a 380M-byte disk increases SRM user disk space by a factor of more than 2.5. In addition, the new ESDI controller runs faster for better access times.

2. **Runs under UNIX System V/386 Release 3.2 Version 2.1 software**
   Release 3.2 UNIX software provides better disk and tape support, that can improve file access performance on any size disk.

## EXISTING FEATURES, LIMITATIONS, RESTRICTIONS, AND WORKAROUNDS

### IMPORTANT NOTE

**Read the following section carefully. While using your iPSC/2 system, record any problems you encounter. Report them to iSC Technical Support at 1-800-421-CUBE or (in Oregon) 629-7777.**

Release 3.0 of the iPSC/2 system software runs on all iPSC/2 system configurations. Some features and known limitations of this release are documented below:

### NX/2 Node eXecutive

1. **Message length**
   Check that the *len* parameter in a node send or receive call does not exceed the size of the buffer. Unpredictable results can occur if *len* exceeds the buffer size. (Refer to the *iPSC®/2 Programmer's Reference Manual* for more information on these parameters.)

2. **Invalid buffer pointers**
   Invalid buffer pointers in C node programs will sometimes be accepted as valid and not return an error message.

3. **Node may hang with file I/O**
   If a node's message buffers are filled, but no receives are posted, attempts to do file I/O may fail. To avoid this failure, you must make sure that the node receives pending messages.
4. ***waitone* call completion code error**

When a node process stops due to an overflow exception, the *waitone* call incorrectly returns a completion code of -243. It should return -252. (Refer to the *iPSC®/2 Programmer's Reference Manual* for a description of the completion codes.)

5. **Incorrect error message displayed during *handler* routine use**
   If the *handler* routine is used to catch a general protection violation (exception 13), a stack overflow error message is incorrectly displayed.

6. **"Memory Fault" error results when more than 300 channels opened**
   Using the Fortran node compatibility library *copen* routine to open more than 300 channels causes a "Memory Fault" error message and the program aborts.

7. ***flushmsg* may not flush all messages**
   *flushmsg* may not flush a message that is in transit.

8. ***cubeinfo* command displays system name only**
   The *cubeinfo* routine returns the system and domain name for the *srmname* and *hostname* fields. The *cubeinfo* command truncates the domain name, and displays just the system name.

9. **Invalid file descriptors in node *dup* call**
   Using an invalid file descriptor of -1 in the *dup* call in node applications will not be detected as invalid.

10. **Printed NaNs from SX applications**
    NaNs (not a number) will be formatted incorrectly when printed from an SX node application (e.g. printed as 0.??? instead of the standard format of ?.???E+00.)

11. **Rewinding a file with *lseek***
    Using the C *lseek* call on the nodes to rewind a file may occasionally return an invalid file pointer value, instead of the expected file pointer value of 0. When this occurs, the *lseek* call does rewind the file as expected. So, you can ignore this invalid return value.

12. ***strcmp* on last environment variable**
    In node applications, *strcmp* may generate a "Memory fault" error when given the last environment variable.

13. **Handler parameters after global *hsend***
    If a node does a global *hsend* to *mypid*, the node parameter in the handler routine contains an undefined value instead of the expected value of -1.

    If a node does a global *hsend* to other than *mypid*, the node and pid parameters in the handler routine contain the sender's node and pid values, not the destination node and pid values.

14. **Prints originating within a handler operate incorrectly**
    Prints originating from within a handler routine may overwrite and/or merge with the prints from other nodes.

## Node Shell (nsh)

1. ***startcube* command not available on nodes in this release**
   The node command *startcube*, documented in the *iPSC/2 Programmer's Reference Manual*, is not available in this release. It is available on the SRM.

2. ***pipe, fork,* and *exec* calls not fully supported**
   UNIX-compatible calls *pipe, fork, execl, execv, execle, execve, execlp, execvp, setuid,* and *setgid* for the nodes are only partially implemented. They are used by the node shell (*nsh*) and may be useful in porting programs from the UNIX environment to run under *nsh*. Their use in newly written node programs is **strongly** discouraged.

3. ***ls -s* on the nodes prints incorrectly**
   If the number of bytes is in the multiple gigabyte range, the node command *ls -s* may print the number as negative.

4. **Interrupting node *stream* command during tape read can hang tape driver**
   Pressing the delete key <Del> while using the node *stream* command to read a tape into CFS can cause the tape driver to hang. If this happens, kill the *fserver* process.

5. ***backup* command writes more than necessary**
   Disk space is allocated in clusters that contain 22 blocks each. When disk space is allocated, the affected cluster is marked as "used". However, some of the blocks in the cluster may not contain data. When you execute *backup*, the system writes all the blocks in the used clusters. The *backup* command writes out approximately 1MB per disk more than is present in data files. The extra data does not affect the quality of the backup. (Refer to the "Suggestions for Using CFS" section.)

6. ***nsh* wildcard (*) does not work properly for filenames larger than 14 characters**
   The nsh wildcard character (*) does not expand properly for CFS filenames longer than 14 characters.

7. **Changing directory names**
   The *mv* command does not work on directories, even to change the name of the directory. The only way to do this is to copy the whole directory structure to the new name and remove the old structure. For example, to move */cfs/dirname/progs* to */cfs/dirname/src*, *cd* to */cfs/dirname* and use the following command sequence:

   ```
   tar cf - progs | tar xvf - src
   rm -r progs
   ```

   This can also be used to move or copy directories to other places in the file system.

8. **Loading and killing processes on nodes not assigned to you**
   Under certain circumstances, it is possible to load and kill processes on nodes that are not assigned to you. If you use the iPSC/2 system as documented, this will not occur.

9. **Maximum SRM shell file size**
   The node shell will only execute the first 512 bytes of a shell file if the file is located on the SRM.
   **Workaround:** Copy the shell file to CFS, then run it.

## Concurrent File System™

1. **Cannot use *open* on a directory name**
   You cannot use *open* on a directory name in CFS. Use *opendir* instead.

2. ***umask* only works on CFS**
   The *umask* routine on the nodes only works on CFS. It is not supported for SRM files.

3. ***bootcube* must be executed after *mkcfs***
   After the *mkcfs* command has been executed, *bootcube* must be executed to reset the file system. (Refer to the "Suggestions for Using CFS" section.)

4. **Using CFS commands/routines in a non-CFS system may hang your application**
   If you have a non-CFS system or a cube that was booted as a non-CFS system (using *bootcube* -o), do not use any of the CFS commands or routines. Doing so may cause your application to hang. If this occurs, use the <Del> key to escape from the hang and then issue a *killcube* followed by a *relcube*.

5. ***backup* and *restore* documentation error**
   The *iPSC®/2 Programmer's Reference Manual* incorrectly states that a *getcube* command, specifying a cube of any size, must be executed prior to using the *backup* or *restore* command. You should execute an entire *bootcube* sequence. Then, use *getcube* to reserve the entire cube for the duration of the operation.

6. ***cread* and *cwrite* documentation error**
   The *iPSC®/2 Programmer's Reference Manual*, incorrectly states that the C versions of *cread* and *cwrite* return the number of bytes read or written. There is no return value for either of these calls.

7. ***restrictvol* applies to file block allocation after file initially created**
   The *restrictvol* call applies to file block allocation only after the file is initially created. The file header block is allocated when the file is created. It can reside anywhere in CFS.

8. **Maximum length of file system name is 29 characters**
   The maximum length of a file system name that may be used in *mkcfs* is 29 characters. The *mkcfs* command erroneously says 30 characters.

9. **Number of disks per I/O node**
   *bootcube* will hang on step 14 if every I/O node does not have exactly the same number of disks.

10. ***iread* error**
    The *iread* call may occasionally abort with the error message "iread: Tries to read past end-of-file", when in fact it did not try to do this. If this occurs, you should use some other method to read the file (e.g. *cread*).

11. **Reading past end-of-file**
    Occasionally, reading past end-of-file using *cread* or *iread* may not be detected as an error. You can use *iseof to* detect end-of-file.

## Concurrent Workbench™

1. ***load* command**
   The *load* command does not report how much memory is used by a node process as it did in the iPSC/2 System.

2. ***waitcube* and *killcube***
   The *waitcube* and *killcube* commands occasionally fail to return. To recover, press <Del> to kill the command, clean up unwanted *fservers*, and then do a *bootcube*.

3. **Nodes marked unusable**
   If you get a message saying, "nn node(s) not responding, marked as unusable", where "nn" is the number of nodes, do a *bootcube* to reset the nodes. This message will only appear on the system console. If problems persist, use *cdp* to check for bad node boards. *cdp* is described in the *iPSC®/2 System Administrator's Guide*.

4. **Message length**
   Check that the *len* parameter in a host send or receive call does not exceed the size of the buffer. Unpredictable results can occur if *len* exceeds the buffer size.

5. **Invalid buffer pointers**
   Invalid buffer pointers in C host programs will sometimes be accepted as valid and not return an error message.

6. **Host long messages**
   Host programs cannot use *csend* or *isend* followed by *crecv* to send long messages (greater than 100 bytes) to themselves. Also, a node program sending a message of greater than 100 bytes to the host will not complete the send operation until the host application posts a receive for the message.
   **Workaround:** Use *irecv*, *csend*, *msgdone*, or *msgwait*.

   Example 1 shows a code fragment from a host program:

```
id = irecv ( 1, buf, sizeof(buf) );
     .
     .
     .
csend ( 1, msg, sizeof(msg), myhost(), mypid() );
msgwait(id);
```

   Example 2 shows a code fragment from a host program that works only if the send and receive buffers are different:

```
id = isend ( 1, msg, sizeof(msg), myhost(), mypid() );
     .
     .
     .
crecv ( 1, buf, sizeof(buf) );
msgwait(id);
```

7. **Releasing a cube**

It is recommended that *killcube* be used prior to releasing the cube. *Relcube* alone does not always clean up processes properly, causing subsequent programs to fail. Using *killcube* avoids this problem. Use the *ps -ad* node command to verify that the clean-up is complete.

8. **Number of nodes in cube**
   *Getcube* always allocates a number of nodes that is a power of two. If you specify a number of nodes that is not a power of two, *getcube* attempts to allocate additional nodes to make the number of nodes a power of two. If the cube with the next largest dimension is unavailable, no nodes are allocated. No error message is displayed. *numnodes* reports the number of nodes actually allocated, not necessarily the number of nodes you requested.

9. **Node memory**
   When the memory size is specified in *getcube*, the only guarantee is that nodes containing enough memory to satisfy the request will be allocated. Even when nodes containing the exact amount of memory specified are available, they might not be allocated by *getcube*.

   Example:

   Given that 1M, 4M, 8M, and 16M node boards are resident and available in the cube,

   **getcube -td0m1** may allocate either the 1M, 4M, 8M, or 16M node.
   **getcube -td0m4** may allocate either a 4M, 8M, or 16M node .

10. ***setsyslog***
    Calling *setsyslog* when the host program is already piping output through *syslog* causes the host program to hang. Press <Del> to recover.

11. ***newserver***
    A host program cannot call *newserver* if its output is being piped through *syslog*. Doing so causes the host program to hang or be killed. Press <Del> to recover.

12. ***load* and *killcube***
    Calling *killcube* immediately after calling *load* may cause the *killcube* to hang. To recover, send an interrupt to kill the host process (the default is to press <Del>) then do a *bootcube*.

13. **"Not enough memory" error message**
    *Killcube* and *relcube* may fail to recover node memory, resulting in "not enough memory" error messages. Use *bootcube* to recover.

14. ***killcube* fails if *waitcube* is in the background**
    If a *waitcube* is executing in the background, a *killcube* cannot be executed until the *waitcube* completes. Attempts to run both at the same time fail with the message "(host) setpid: Pid already in use."

15. ***killcube* flushes file server messages**
    If a host program calls *killcube* before nodes complete file I/O, some output may be lost.
    **Workaround:** Call *waitcube* before calling *killcube*.

16. **_getcube_ may allocate one or more SX nodes**
When you have a mixture of SX and 387 nodes, a *getcube* with no *cubetype* specified may allocate a mixture of node types, without telling you what types of nodes it allocated. This can cause floating point exceptions, if you execute 387 code on an SX node, or vice versa.
**Workaround:** Use *getcube* with either the **-tf** or **-tsx** option, to force it to allocate a cube of either all 387 or all SX nodes.

17. **_bootcube_ must complete before you execute _getcube_**
*bootcube* must be complete before attempting to execute *getcube*, otherwise the whole cube may be marked as bad.
**Workaround:** To recover, do another *bootcube*.

18. **Shutting off cube occasionally causes SRM panic**
Shutting off the cube may cause the SRM to panic. To recover, reset the SRM.

19. **Outstanding _crecv_ or _msgwait_ will block any _hrecv_**
An outstanding *crecv* or *msgwait* blocks any *hrecv* from being serviced.

20. **Cube allocated by _bootcube_**
When *bootcube* is executed on an SRM, a 0 node cube named "iocube" is allocated. This is necessary for all systems. It reduces the number of cubes that users can allocate from 10 to 9.

21. **Maximum of 11 outstanding _irecv_'s may be posted on an SRM**
A maximum of 11 outstanding *irecv*'s per cube may be posted on an SRM by a host program. If you attempt to post more than this, an error message displays, and the process aborts. The maximum for the remote host is 12.

22. **Multiple _getcube_ calls**
If you call *getcube* more than once in a host program, it will attach you to all cubes allocated in that host program, rather than just the last one allocated.

23. **iPSC error numbers**
The iPSC error numbers, used to set *errno*, start at 150 in this release. They are now located in */usr/include/sys/errno.h* instead of *cube.h*, to follow the standard UNIX model more closely. These numbers started at 110 in Release 2.4.

24. **_rebootcube_ command added**
Nonroot users can use *rebootcube* to reload the NX/2 operating system on all the nodes of a cube.

25. **Loading vector applications on nonvector nodes**
Loading a vector application into a nonvector node will cause *load* to hang. To recover, press the <Del> key and do a *bootcube*.

26. **Running multiple processes on vector boards**
Only one process at a time can use a vector board. Attempting to load a second process onto the vector board will cause an error message, and may also corrupt the first process.

27. **Changes to *gopf***
Changes were made to the *gopf* routine to allow all data types to be used with it. The *n* parameter, that represented the number of elements in the vector *x*, has been replaced by the *xlen* parameter. This parameter represents the number of bytes in the vector *x*. In addition, the length parameter *n* is no longer passed to the function *f*, which now takes only two parameters. Refer to the *iPSC®/2 Progammer's Reference Manual* for more information on using *gopf*.

28. **Force types in message passing calls**
Some message type values used in the message passing calls have special functionality. Types 0 to 999,999,999 are normal message types, that function as they did in previous software releases. Types 1,073,741,824 to 1,999,999,999 are special force types, that bypass the normal flow control mechanisms, do not match the -1 wildcard, and are discarded if no receive is posted. Other types in the positive integer range are used by the system and should not be used in user applications. See the *iPSC®/2 Programmer's Reference Manual* for more information on force types.

29. **Bad vector board**
If the cube contains a bad vector board, *bootcube* may do one or both of the following:

- Not detect the board as being bad
- Indicate that it is using */tmp/cubeconf* instead of */usr/ipsc/conf/cubeconf*, even though these two files are identical.

## Concurrent Workbench™ Assembler and Linker

1. **Directive incorrectly used**
In assembling data objects declared with the *.data* instruction, the assembler creates a large temporary file on the SRM. This file can exhaust all available file space on the SRM and abort the assembly.
**Workaround:** Such data objects will not create large files during assembly if they are declared using the equivalent *.bss* instruction.

2. **Unsupported flag**
The **-Y** flag is not supported in the assembler even though it is documented in the UNIX System V/386 manuals.

3. **The assembler is undocumented**
*as* is only documented as *as(1)* in the UNIX System V/386 manuals. The actual assembly instructions are undocumented.

4. **Data alignment by the compiler and linker**
The **-vx** switch on the compiler may not correctly align all variables on their natural boundaries for the vector board. This can cause unpredictable results, such as wrong answers or parity errors.
**Workaround:** You can use the tool *nm* to make sure that all variables used on the vector board are aligned correctly.

If you use the vector board and have misaligned data, you may get a parity error under certain circumstances. This is due to a hardware problem with the 386

node's iLBX-II interface and the Vortex iLBX-II interface. To avoid this problem, you must ensure that all variables start on their natural boundaries, and that all strings start on a 32-bit boundary. The Fortran compiler does this for all variables except character strings.
**Workaround:** When sending and receiving character strings, you should ensure that these strings are multiples of four bytes. Also, the string's starting address should be on a 32-bit word.

## Concurrent Workbench™ C

The default C compiler has been changed from the AT&T version of the *pcc* compiler to the Green Hills C compiler, *gcc*. This was done for performance and compatibility with Green Hills Fortran.

UNIX software on the SRM is built with the AT&T *pcc* compiler rather than the Green Hills compiler. The support libraries for the Green Hills compiler have been remade with the AT&T include files for compatibility with UNIX software. These libraries are now the same as those used by the *pcc* compiler. Problems encountered with the libraries can be checked against the *pcc* compiler.

AT&T's *pcc* compiler is located in:

*/usr/bin/pcc* *(a script that calls the real pcc compiler, located in /lib/cc)*

*/lib/cc* *(AT&T's pcc compiler)*

To invoke AT&T's *pcc* compiler, enter:

**pcc**
or
**/lib/cc**

Green Hill's compiler is located in:

*/bin/cc*
and
*/usr/bin/gcc* *(these are copies of the same file )*

To invoke Green Hill's compiler, enter:

**cc**
or
**gcc**

The following are Concurrent Workbench C features, limitations and workarounds:

1. **asm directive support**
   The Green Hills compiler is missing some support for the *asm* directives.

2. **-pg not supported by UNIX software**
   The **-pg** switch is not supported by UNIX System V/386 software. It is a Berkeley UNIX feature.

3. **-C compiler switch**
   The -C compiler switch is not implemented.

4. **-B compiler switch**
   There is a **-B** compiler switch for both Fortran and C which should be used when compiling programs for use with the debugger, DECON. The **-B** switch creates the optimum debugger environment by setting the appropriate switches. By listing **-B** last in the sequence of compiler flags, it will have precedence over the previous flags, except for the **-O** and **-OLM** flags. These flags should be removed.

   The **-B** flag sets the following flags: **-g, -ga, -X9, -X18, -X39, -X168, -X211, -X219, -X230, -X307, -X356**.

   If a program works correctly with these flags, but incorrectly in more optimized versions, you should report a compiler bug to iSC.

5. **SX compilation**
   Compiling for the SX requires that the **-sx** flag be on both the compile and link commands. Modules compiled with the **-sx** compiler flag may not be linked with those compiled without the **-sx** flag and vice versa.

   When you call math routines that are in */lib/libsxm.a*, you must also link them using the **-lm** switch (that is, **-lm -sx**).

6. **Data alignment by the compiler and linker**
   Refer to item 4. of the "Concurrent Workbench™ Assembler and Linker" section.

7. **Compiler -p switch**
   The compiler **-p** profiling switch causes an incorrect value to be placed in *argc*. Do not use profiling on C programs that use the command line argument capabilities. The **-p** switch and the command line arguments are mutually exclusive.

8. **Assignments to unused variables**
   Occasionally, assignments to unused variables are optimized out of the code, even when optimization is not invoked.

9. **Illegal external variable names**
   You may not use the following names as external variables in your C programs: *align, any, C, D, E, e, f, FP, herror, host, hostbuf, HOSTDB, hostf, index, IP, KS, L, line, LogFile, LogMask, nbuf, net, P, proto, R, recv, S, s, send, shifts, syslog*, or *token*. Network library, */usr/lib/libsocket.a*, has defined these names to be external. Errors such as "Bus error - Core dump" may occur at runtime if you use them in an application.

## Concurrent Workbench™ Fortran

1. **Fortran INCLUDE statement**
   The Fortran INCLUDE statement is supported in the compiler. The absolute path of the include file must be used if the file is not located in the current directory.

   Example:

```
INCLUDE  '/usr/include/fcube.h'
```

2. ***gray, ginv, cubeinfo,* and *csendrecv* return integers**
   The iPSC/2 functions *gray, ginv, cubeinfo,* and *csendrecv* must be declared as integers, either with the INCLUDE file */usr/include/fcube.h* or with user-provided declarations.

3. **Output lines from *write* or *print* statement**
   Instead of using the first character in a *write* or *print* statement as a printer control character, the character is printed. Also, an extra space is added for each continuation line in these statements. Use the **-vms** compiler switch to make the *write* and *print* statements use the first character as a printer control character and to eliminate the extra space on continuation lines.

4. **Error messages for the following are not provided:**
   Extra characters on a DO statement
   DATA statements used with a variable declared in a COMMON block but not in BLOCK DATA
   Trigonometric function whose arguments > 2**63
   Same Formal argument name used in a subroutine statement more than once
   Logical variable used as array index
   Integer variable used as logical
   Mix of character and any other type within a common block

5. **-I2 compiler flag**
   The **-I2** compiler flag (integers default to 2-bytes) returns an incorrect error message for large integer arrays, and the compilation aborts.

6. **SX compilation**
   Compiling for the SX requires that the **-sx** flag be on both the compile and link commands. Modules compiled with the **-sx** compiler flag may not be linked with those compiled without the **-sx** flag and vice versa.

7. **Invalid invocation line switches**
   Invalid switches presented to the compiler do not result in error messages, but are ignored.

8. **-pg not supported by UNIX software**
   The **-pg** switch is not supported by UNIX System V/386. It is a Berkeley UNIX feature.

9. **VAX/VMS extensions**
   VAX/VMS Fortran extensions are available during both compilation and linking, using the **-vms** switch. These extensions include %loc, %val, WHILE, ENCODE/DECODE, and VAX/VMS carriage control. For detailed information, refer to the *VAX-11 Fortran User's Guide and Language Reference Manual*, available from Digital Equipment Corporation.

10. **-B compiler switch**
    There is a **-B** compiler switch for both Fortran and C which should be used when compiling programs for use with the debugger, DECON. The **-B** switch creates the optimum debugger environment by setting the appropriate switches. By listing **-B** last in the sequence of compiler flags, it will have precedence over the previous flags, except for the **-O** and **-OLM** flags. These flags should be removed.

The **-B** flag sets the following flags: **-g, -ga, -X9, -X18, -X39, -X168, -X211, -X219, -X230, -X307, -X356**.

If a program works correctly with these flags, but incorrectly in more optimized versions, you should report a compiler bug to iSC.

11. **Data alignment by the compiler and linker**
Refer to item 4. of the "Concurrent Workbench™ Assembler and Linker" section.

12. **Do not link Fortran node programs using -lnode switch**
Fortran node programs should not be linked to the node library using **-lnode**, because it links the libraries in the wrong order. This may result in "Memory Fault" errors at runtime. Use "**-node**" instead.

13. **Running out of node board memory**
If you run out of node board memory while running a Fortran application, you may receive the message, "Fortran runtime error on external file "" (106): Buffer too large". This can happen when making the first write to a file, as the program tries to allocate a buffer for the file.

14. **I = MOD( J, 0 ) causes error message**
The expression I = MOD( J, 0 ) causes an " Illegal instruction - core dumped" message.

15. **-OL switches occasionally cause an error message on SX**
The **-OL** ( but not **-O** ) optimization switches occasionally cause an "Illegal address" error while executing on the SX.

16. **Compiler -p switch**
The compiler **-p** profiling switch causes an incorrect value to be returned by function IARGC.

17. **Formatted print statements on multiple nodes**
Using Fortran formatted print statements that contain carriage return control characters ("/") on multiple nodes may cause prints from multiple nodes to merge together. That is, a message line from one node may have portions of a message from another node mixed in.

18. **Fortran scratch files**
Fortran scratch files do not have unique names. They are created in the current directory of the executing program. This means that unless the node program changes directories, the scratch file is created on the SRM's disk, and all node programs share the same scratch file. A scratch file is deleted when the program terminates.

19. **Missing "FILE = " specifier**
In Fortran, if you are missing the *FILE =* specifier in the *open* call, unique file names cannot be created. The file is created in the current directory of the executing program. This means that unless the node program changes directories, the file is created on the SRM's disk.

20. **Node version of *rewind***
The node version of the Fortran *rewind* routine simply resets the file pointer to the beginning of the file. It does not shrink the file, when appropriate, as does the standard *rewind* routine.

## Concurrent Workbench™ Remote Host

1. **Cube ownership**
Cubes that you own are not automatically released when you logout from the remote workstation. You must use *relcube* to release the cube.

2. **Underscore byte swapping calls**
Underscore versions of byte-swapping calls can be used by including *cube.h* in your C source program.

3. **Maximum of seven cube partitions per remote host**
The remote host software supports a maximum of seven cube partitions per remote host. The *iPSC®/2 Programmer's Reference Manual* says that ten cubes are allowed.

4. **First *getcube* may fail**
The first *getcube* after a *bootcube* may fail with a "(host) getcube: No SRM that matched your request was found" message. Subsequent *getcubes* succeed.

5. ***setenv TTY `tty`***
The TTY environment variable is used by several cube commands and must be set before you use the cube. Because each newly created window on a Sun workstation inherits its parent's shell variables, you should set TTY in *.cshrc* and not in *.login*. However, for the remote copy command (*rcp*) to work properly, you must redirect the standard error from the *setenv TTY `tty`* to null as follows:

```
setenv TTY `tty` >& /dev/null
```

6. **Remote host include files**
To use remote host include files in remote host programs, you must compile with the remote compile command *rcc* **-cpp** . . . and do one of the following:

   A. Use absolute paths for the include files:
   In application: #include "*/usr/include/suntool.h*"

   B. Use relative paths in the *#include* statement and use the **-I** switch on the compile command line:
   In application: #include "*suntool.h*"
   command line: *rcc -cpp -I/usr/include*

   C. Have a *rhostinc.h* file included in the application that contains the standard includes:
   In application: *#include "rhostinc.h"*
   In rhostinc.h: *#include <suntool.h>*

   D. The iPSC/2 include files *cube.h*, *fcube.h*, and *errno.h* are installed in the user-specified *INCDIR* directory. To modify the actual location of *INCDIR*, edit the

top-level makefile found in the *rhost* source code and run *make* on the *rhost* software.

7. **_cubeinfo_'s SRM field does not contain complete name**
   On a remote host, the *cubeinfo* command's *srmname* field contains the SRM name specified by the *getcube* **-h** switch, which may not be the complete name.

8. **Heavy message passing**
   Heavy message passing between a remote host and the nodes for a sustained period of greater than five hours may cause the remote *commser* process to die. To recover, execute a *bootcube* on the remote host as root and release the cube on the SRM.

9. **Calling _getcube_ and _newserver_**
   If you call *getcube* with a *keep* parameter = 0 in a host program then call *newserver*, you receive a "(host) file server: There is no attached cube" error message when the host program terminates. Ignore this message.

10. **Maximum of 12 outstanding _irecv_'s may be posted**
    A maximum of 12 outstanding *irecv*'s per cube may be posted on a workstation by a host program. If you attempt to post more than 12, an error message displays and the process aborts. The maximum for the SRM is 11.

11. **iPSC error numbers**
    The iPSC error numbers used to set *errno* start at 150 in this release. They are now located in */usr/include/sys/errno.h*, to follow the standard UNIX model more closely. These numbers started at 110 in Release 2.4.

12. **New byte swapping procedures added**
    New procedures *HTOCC, CTOHC, createstruc*, and *relstruc* have been added to properly align elements of C structures to match the node byte alignment. Refer to the *iPSC®/2 Programmer's Reference Manual* for more information.

13. **FORCETYPE range messages**
    Message types in the FORCETYPE range (types that are greater than 40000000 hexadecimal), sent from a node to a remote host, must be received using a *csendrecv* or *isendrecv* call. Force messages sent from a node to a *crecv* or *irecv* on a remote host will be lost. Force messages can be sent from host program to host program and received using *crecv* or *irecv*. Force messages can also be sent from node program to node program and received using *crecv* or *irecv*.

14. **Nodes cannot _load_ or _exec_ files on remote host**
    Node processes cannot *load* or *exec* files on the remote host. The *load* and *exec* calls executed on the node try to find the files on the SRM, not the remote host workstation.
    **Workaround:** Any files that need to be *loaded* or *exec'ed* from the nodes should be resident in the SRM file system. Either *rcp* the files from the remote host to the SRM or use NFS to mount the remote host file system on the SRM.

## DECON Concurrent Debugger

1. **-B compiler switch**
   There is a **-B** compiler switch for both Fortran and C which should be used when compiling programs for use with the debugger, DECON. The **-B** switch creates the optimum debugger environment by setting the appropriate switches. By listing **-B** last in the sequence of compiler flags, it will have precedence over the previous flags, except for the **-O** and **-OLM** flags. These flags should be removed. Compiling object files for use with DECON may be done as follows:

   ```
   f77 -c -B host.f
   ```
   or
   ```
   cc -c -B host.c
   ```

2. **Loading node programs**
   DECON must be in control when a node program is loaded so that it can generate symbol information. Therefore, you must comment out any *load* calls in your host program.

3. **Accessing Fortran variables declared in COMMON**
   To access (i.e., *display* or *assign*) a variable declared in COMMON, you must indicate the name of the common block in which it is found using the following notation:

   ```
   [common_name].common_member
   ```

   where *common_name* identifies the name of the common block. Omit *common_name* if it is a blank (unnamed) common block. For example, given

   ```
   COMMON /abc/ a(10)
   COMMON // xyz (5)
   ```

   you would use *abc.a(1)* to identify the first element in array *a* in named common *abc*, whereas, *.xyz(3)* would identify the third element in array *xyz* in blank common.

4. **Debugging a program on the SRM only**
   If you are debugging a program only on the SRM, you should specify 0 nodes when doing a *getcube* before starting up DECON (e.g., *getcube* **-t0**). This will keep you from tying up nodes that you are not using.

5. **No remote DECON**
   DECON only runs on the SRM. When using a remote workstation, *rlogin* to the SRM to use DECON.

6. **Unsupported features**
   Data breakpoints/tracepoints on host processes are not supported.
   Label breakpoints/tracepoints on host and node processes are not supported.

7. **Register variables in C**
   Displaying and assigning register variables may produce inaccurate results because the compiler does not guarantee to keep these variables in registers at all times. In general, if you can successfully display a register variable, you may then assign a value to it.

8. ***assign* command limitations**
   An indirect *assign* (that is putting the value on the same line as the command) to a subrange or entire Fortran character array does not work correctly. All elements of the array are concatenated and treated as a single unit.
   **Workaround:** Use *assign* on each individual element of the array.

9. ***break* command limitations**
   When using the **break when <condition>** form of the *break* command, *<expr>* must be a simple integer value. Arithmetic expressions are unsupported.

10. ***display* command limitations**
    When displaying an entire Fortran character array, all of the elements are printed out in a single concatenated string.

11. ***step* command limitations**
    Occasionally, the compiler maps multiple source statements of a C or Fortran program into a single code address, especially when these statements are short (e.g., **cnt = 0**). When this is the case, DECON will execute more than one statement when it is asked to do a single step.

    *step* cannot be used to terminate a node program. If you attempt to step off the end of a node program, you will get the message "error: stopped at an unknown location xx."
    **Workaround:** Use the *run* command to complete execution of a node program.

12. ***type* command limitations**
    The type of a Fortran *character*n* variable will just be displayed as *character*. The type of a *logical*1* Fortran variable is displayed as *character*, and the type of a *logical*4* is displayed as *integer*4*.

13. **Documentation errors in the *DECON User's Guide***
    A. Page 3-8, remove "=" from the *assign* syntax.

       The current line reads:

       **assign [<debug-context>] variable [=<expr>]**

       the line should read:

       **assign [<debug-context>] variable [<expr>]**

    B. Page 1-2, Item 2 is misleading. It should read:

       The **-X18** compiler option turns off a portion of the compiler optimization phase.

14. **Interrupt while host process is doing a system call**
The system restarts a *csend*, *crecv*, *cprobe*, or *msgwait* system call if you resume a stopped host process that was previously interrupted while doing a system call.

15. **Fortran array starting index**
COFF only records the number of elements in an array. DECON assumes that the first element has index 1 for Fortran arrays and 0 for C arrays. If you have a starting index other than 1 in your Fortran program, you need to adjust the index to 1 so that *assign* and *display* can identify the correct element.

16. **Input to a user program may be intercepted**
The input you are trying to send to your program might be intercepted by DECON if it is waiting for input at the same time. If this is the case, DECON will complain that an unknown command has been entered. Simply retype the input to your program.

17. ***dimension* command returns number of nodes, not dimension of cube**
DECON's *dimension* command returns the number of nodes rather than the dimension of the allocated cube.

18. **Maximum pathname**
The maximum length of the source pathname is 32 characters.

19. ***hload* documentation is incomplete**
The pid specified with the *hload* command must be the same as the *setpid* value used within the host program.

20. ***list* has -f switch**
The -f switch on the *list* command displays the name of the source file being listed.

21. ***file* command has been modified**
The command *file <fname>* sets the current source file to *<fname>*, which is then the file used by the *list* command. Either a *step* or a *run* command changes the current source file back to the one being executed.

22. **Interrupting and resuming host application**
Interrupting a host application and continuing its execution several times during a debugging session may hamper message passing. To prevent this, kill the host process occasionally. This clears all messages associated with the process from the system. If messages are not getting through at all, execute *bootcube* as root to clean things up.

23. **Erroneous "Process terminated" message**
After a node loads and terminates normally, if you load another program with the same pid, you receive the message "Process terminated". Ignore this message because it refers to the first process that was loaded.
**Workaround:** To prevent this message from appearing, execute a *killcube* between each load.

24. **Automatic execution of command file now available**
When you invoke DECON, it first searches the current directory, then the home directory for the file name *.deconcf*. When it finds *.deconcf* in either directory, DECON executes the file. This file should contain valid DECON commands that

you want executed on entering DECON. The *.deconcf* file is useful for setting up aliases for DECON commands.

## iPSC®/2 Simulator

1. **Process creation**
   The simulator creates one UNIX/XENIX process for every simulated node or host process. Therefore, the size of a simulation is limited by the maximum number of processes allowed by the operating system being used. XENIX software allows 14 processes to be created and UNIX 4.2 BSD, UNIX 5.2 ATT, and UNIX 5.3 ATT allow 23 processes to be created. Limit the cube's dimension and/or the number of processes per node to conform to these limitations.

2. **New CFS calls and some node calls not supported**
   The new I/O calls (such as *cread, cwrite, iowrite, iodone,* and *iomode*) are not supported in the simulator. The NX *handler, hsendrecv, csendrecv, isendrecv, hsend* calls are not supported either.

3. **FORCE_TYPE messages not supported**
   FORCE_TYPE messages are not supported. Types greater than 40000000 hex are treated the same way as any other message.

4. **Clock calls and timing**
   Note that all timing uses a common time base. Because the simulated processes are executed in sequential timeslices by UNIX software, the values from the clock are different from those obtained on the iPSC/2 system.

5. **Remote host and cube sharing**
   Remote host and cube sharing functions are not supported by the simulator.

6. **System Resource Manager (SRM) programs**
   System Resource Manager programs are started within the simulator. Thus, the command line cannot be used to supply arguments to the SRM programs or redirect their standard input or output.

7. **File descriptors**
   File descriptors 9, 10, 11, and 12 are reserved for the simulator.

8. **Signals**
   Signal number 28 in the UNIX 4.2 BSD environment and signal number 16 on XENIX, UNIX 5.2 ATT, and UNIX 5.3 ATT are reserved for the simulator.

9. **Interchanging calls**
   The simulator accepts all host and node calls in both host and node programs, whereas the iPSC/2 System does not. Therefore, it is possible to write programs for the simulator which will not run on the iPSC/2 System. Refer to the *iPSC®/2 Programmer's Reference Manual* for more information on the calls supported on the host and nodes.

10. **UNIX 5.2 ATT version software**
To make and install UNIX 5.2 ATT version software, enter:

```
make  att52
make  instlatt
```

Other versions are documented under "Installing the Simulator" in Chapter 2 of the *iPSC®/2 Simulator Manual.*

## Cube Diagnostic Program

1. **Invoking the Cube Diagnostic Program**
To invoke the Cube Diagnostic Program (*cdp*) on the SRM, log in as root and perform the following steps:

```
cd  /usr/ipsc/diag
bootcube -D1-2
./cdp
```

2. **Aborting a test**
You can use the <Del> key to abort *cdp* tests. *cdp* ignores this key during diagnostic channel communications. Therefore, it may be necessary to hold the <Del> key down until *cdp* recognizes it, and the "Abort in progress" message appears on your terminal. If it becomes necessary to kill *cdp*, press <Ctrl, \> .

3. **Invoking the Optional Board Cube Diagnostic Program**
To invoke the Optional Board Cube Diagnostic Program (*cdpo*) on the SRM, log in as root and perform the following steps:

```
cd  /usr/ipsc/diag
bootcube -1 -o
getcube -tdiag
./cdpo
```

4. **I/O Subsystem Diagnostics**
The *cdpo* I/O Subsystem Diagnostics require a device configuration file to verify that the correct devices are installed and functional. This file, called *devconf*, resides in the */usr/ipsc/conf* directory. If the *devconf* file does not exist, cannot be read, or contains no useful information, the I/O diagnostics assume that no devices are installed. The *devconf* file can be created by hand, with your favorite editor, or automatically, by executing the *mkdevconf* program. To execute the *mkdevconf* program, login as root and perform the following steps:

```
bootcube
getcube
cd  /usr/ipsc/conf
../diag/mkdevconf
relcube
```

Examine the *devconf* file for accuracy and, if necessary, edit the file to match the actual system hardware.

5. **Timeout Table Overflow**
When you run *cdp* for extended periods of time, the UNIX software may produce a timeout table overflow warning message and hang the SRM. Should this happen, you must reboot the SRM.

## UNIX Software

1. **Exit value converted by sh**
*/bin/sh* converts a C program exit (-1) value to 255.

2. **Cartridge Tape**
The cartridge tape occasionally hangs. To recover, the SRM must be turned off and turned back on.
**Workaround:** Avoid typing the kill character while the cartridge tape is just beginning to read or write the tape. Wait for 10 seconds after the transfer to or from the tape has started.

3. **Removing the root password**
The procedure for removing the root password described in the *iPSC®/2 System Administrator's Guide* is incorrect. Refer to the "Procedure for Removing Root's Password" in the "User Information" chapter of these release notes for the correct procedure.

4. ***pcc* compiler *-p* switch**
The *pcc* compiler **-p** profiling switch causes an incorrect value to be placed in *argc*.

5. **Cannot use *cpio* with *noclobber* set**
You cannot have *noclobber* set in your environment when you wish to use *cpio* to write to a disk device.

6. **Forcing a script to execute in *csh***
To force a script to execute in *csh*, the first line of the script must be "*#!*". In UNIX 3.0 software, you could simply use a "*#*" as the first character in the file.

7. ***unsetenv* not supported**
The *unsetenv* command is not supported in UNIX software 3.2. *unsetenv* is not included in the *csh*.

## TCP/IP Software

1. **System names used by *uname* and *hostname***
In this release, system names that are to be used by *uname* and *hostname* must be less than or equal to eight characters, and consist of letters and numbers only.

2. **/etc/hosts file change**
The */etc/hosts* file has changed with this release. The word "localhost" must now appear on the first line as follows:

```
127.0.0.1 localhost
```

The word "loopback" is no longer used. Nothing is added to the address lines to indicate which system is the localhost. The localhost is determined by matching the *uname* with the host names listed in */etc/hosts*. The TCP installation process

creates a proper new */etc/hosts* file based on system names and addresses given during the installation. Before you replace this file with your old */etc/hosts* file, compare them and modify your old file accordingly.

3. ***rsh* and *rshx* replaced by *rcmd***
   Remote shell commands *rsh* and *rshx* have been replaced by the command *rcmd*. Its syntax is the same as the old commands.

4. ***TERM* variable in *.login***
   If the tty in a *.login* file is a pseudo-tty, do not explicitly set the environment variable *TERM*. *TERM* is set by *rlogin*.

   If you login from a system that does not have autologin (e.g. Excelan system), you can set *TERM* by hand, after you are logged in. The system assigns a bogus value to *TERM*. This value does not change, even after you do a *setenv*. The value given in the *setenv* is used, not the bogus value.

# SOFTWARE INSTALLATION 2

## INTRODUCTION

### IMPORTANT NOTE

**Before you attempt to install any of your system software, read all of the instructions in the following sections. If you encounter trouble during your installation, call Customer Support at 1-800-421-CUBE (in Oregon 629-7777).**

Use the following procedures to reinstall all software on the hard disk. You only need to do this if the software becomes damaged and is unusable. We recommend you call Intel Scientific Computers first, if you believe your UNIX system needs to be rebuilt.

## INSTALLING UNIX SYSTEM V/386 RELEASE 3.2 VERSION 2.1

Before you install UNIX software on your iPSC/2 System, back up all your user files on tape. **Installing the new UNIX software reformats your hard disk.** If you have modified any system files, save the modifications. After you have installed the UNIX software, add the modifications you saved to your new system files. **Do not replace your new system files with the modified system files you saved.**

**NOTE**

Before you install your UNIX software, verify that the proper Ethernet Controller board is installed in your system.

Use the following procedure to install UNIX V/386 Release 3.2 Version 2.1 on your system.

1. If your system is off, turn it on.

   If your system is running, shut it down by entering:

   ```
   shutdown -g0 -y
   ```

2. Insert the UNIX boot floppy diskette into the diskette drive.

3. Reboot the system by pressing **<Ctrl, Alt, Del>**.

4. Your system monitor displays:

   ```
   Please wait while existing file systems are checked for
   consistency...
   ```

5. After an interval, your monitor displays:

   ```
   Strike ENTER to install the UNIX System on your hard disk.
   ```

   Press **<Enter>**.

6. Your system monitor displays:

   ```
   WARNING: A new installation of the UNIX System will destroy
   all files currently on the system. Do you wish to continue
   (y or n)?
   ```

   If all your files are backed up, enter: **y**

7. Your system monitor displays:

   ```
   Total hard disk size is xxxx cylinders

                                        Cylinders
   Partition  Status  Type          Start  End    Length   %
       1      Active  UNIX Sys        0    xxx     xxx    100

   SELECT ONE OF THE FOLLOWING:

      1. Create a partition
      2. Change Active (Boot from) partition
      3. Delete a partition
      4. Exit (Update disk configuration and exit)
      5. Cancel (Exit without updating disk configuration)
   Enter Selection:
   ```

   Because the partitioning has already been done, enter: **5**

8. Your monitor displays:

```
A surface analysis will now be done.
This will destroy all data on the hard disk.
Strike ENTER to continue or DEL to abort.
```

To begin the surface analysis, press **<Enter>**.

9. When the surface analysis is done, your monitor displays:

```
xx bad tracks were found during the verify pass.

The UNIX System partition has xxx cylinders assigned to it.
x cylinders will be used for alternate sectors and tracks.
This leaves xxx cylinders (xxxxxxxx bytes) available.

The following seems like a reasonable partitioning of your
UNIX System disk space:

A root filesystem of xxx cylinders (xxxxxxx bytes), a user
(/usr) filesystem of xxx cylinders (xxxxxxx bytes), an
extra user filesystem (/usr2) of xxx cylinders (xxxxxxx
bytes), and a swap/paging area of xxx cylinders (xxxxxxx
bytes).

Is this allocation acceptable to you (y/n)?
```

To keep the partitioning that was originally shipped to you, enter: **n**

10. Your monitor displays:

```
Do you wish to have separate root and usr filesystems
(y/n)?
```

Enter: **y**

11. Your monitor displays:

```
Do you want an additional /usr2 filesystem (y/n)?
```

Enter: **n**

12. Your monitor displays:

```
You will now be given the opportunity to specify the size,
in cylinders, of each filesystem. (One megabyte of disk
space is approximately 8 cylinders.)

How many cylinders would you like for swap/paging (1-xxx)?
```

For a 140MB Maxtor disk drive, enter: **109**

For a 380MB Maxtor disk drive, enter: **78**

13. Your monitor displays:

```
How many cylinders would you like for root (1-xxx)?
```

For a 140MB Maxtor disk drive, enter: **201**

For a 380MB Maxtor disk drive, enter: **143**

14. Your monitor displays:

```
You have specified the following disk allocation: A root
filesystem of xxx cylinders (xxxxxx bytes), a user (/usr)
filesystem of xxx cylinders ( xxxx bytes), and a
swap/paging area of xxx cylinders ( xxxxxx bytes).

Is this allocation acceptable to you (y/n)?
```

If the cylinder quantities for root and swap/paging match what you intended to input, enter: **y**

15. Your monitor displays:

```
Reboot the system now.
```

Wait until the diskette access light turns off, then remove the boot diskette from the floppy diskette drive.

16. Press **<Ctrl, Alt, Del>**. The system reboots itself.

17. Your monitor displays:

```
Installation from Cartridge Tape is available using
Interrupt #5 and Address Range 300 through 301. Are you
installing from tape (y/n)?
```

Enter: **y**

18. Your monitor displays:

```
Please make sure your Cartridge Tape hardware is configured
correctly. Insert System Installation Tape in drive and
press <RETURN>
```

19. Insert the UNIX SYSTEM V/386, R3.2 V2.1 tape in tape drive.

20. Press **<Enter>**.

21. When installation of the base software is complete, the system asks you for a root password.

    Press **<Enter>** or enter your own password.

    Pressing **<Enter>** inputs <Enter>, which is one of the shortest, easiest passwords.

22. The system asks you for an "install" password.

    Press <Enter> or enter your own install password.

23. Your monitor displays:

```
Do you want to install the Cartridge Tape Driver?
Press ENTER for YES or ESC for NO

Strike ENTER when ready or ESC to stop.
```

    Press <Enter>.

24. Your monitor displays:

```
Installing Cartridge Tape Utilities - Version 2.0 ...

Available interrupt(s) for the Cartridge Tape Utilities -
Version 2.0:
   IRQ 2
   IRQ 5

   Type the interrupt number and strike the ENTER key or
   type Q to cancel installation.
```

    Choose IRQ 5 (interrupt 5) by entering: **5**

25. Your monitor displays:

```
You are installing the Cartridge Tape Utilities - Version
2.0 using hardware interrupt number 5 and addresses 300
through 301. Be sure that these values are not in use by
another add-on board.

Strike ENTER when ready
or ESC to stop
```

    Press <Enter>.

26. Your monitor displays:

```
Reboot the system now
```

    Wait until the tape has finished rewinding. Leave the tape in the drive.

27. Press <Ctrl, Alt, Del>.

28. Login as "root".

29. Enter:

```
installpkg
```

30. Your monitor displays:

```
Are you installing from tape (y/n)?
```

Enter: **y**

31. Your monitor displays:

```
Insert Installation Tape in drive and press <RETURN>
```

If you removed the UNIX tape, insert the tape in the tape drive.

32. Press **<Enter>**.

33. Your monitor displays:

```
Searching Tape for list of packages. This may take awhile.
Please wait.
```

After an interval, your monitor displays all the available packages, as follows:

```
Available Packages:

   1. Editing Package Version 2.0
   2. Extended Terminal Interface Package Version 2.0
   3. C Software Development Set 4.1.6
   4. Network Support Utilities Package (1.2) Version 2.0
   5. 2 Kilobyte File System Utility Package Version 2.0
   6. Kernel Debugger(s) - Version 2.0
   7. PC586 Ethernet Driver - Version 1.0
   8. System Administration Software
   9. Remote File Sharing Package (1.2) Version 2.0
   10.XENIX File System Package Version 1.0

Do you want to install all of the above packages? <y/[n]>:
```

Enter: **n**

34. Enter **y** next to the following packages to install them:

```
Editing Package Version 2.0
Extended Terminal Interface Package Version 2.0
C Software Development Set 4.1.6
Network Support Utilities Package (1.2) Version 2.0
2 Kilobyte File System Utility Package Version 2.0
Kernel Debugger(s) - Version 2.0
PC586 Ethernet Driver - Version 1.0
System Administration Software
```

35. When the kernel debugger is being installed, your monitor displays:

```
Which kernel debugger(s) do you want to install?

   1) DEBUGGER (polish calculator style)
   2) GDEBUGGER (traditional)
   3) both DEBUGGER and GDEBUGGER

Choose 1, 2, or 3

Enter: 1
```

36. At the end of System Administration package installation, your monitor displays:

```
Do you want to give passwords to administration login?
(y/n) [n]
```

Press **<Enter>**.

37. Your monitor displays:

```
To complete the install/remove process a shutdown is now
being initiated automatically.

Make sure your floppy drive is empty. If you are installing
or removing controller boards, you may power down the
system after the shutdown has completed.

Strike ENTER when ready or ESC to stop
```

Press **<Enter>**.

38. Reboot the system by pressing **<Ctrl, Alt, Del>**.

39. Login in as "root".

40. Insert the Remote Terminal Package diskette into the floppy diskette drive.

41. Enter:

```
installpkg
```

42. Your monitor displays:

```
Are you installing from tape (y/n)?

Enter: n
```

43. Your monitor displays:

```
If the program installation requires more than one floppy
disk, be sure to insert the disks in the proper order,
starting with disk number 1.

After the first floppy disk, instructions will be provided
for inserting the remaining floppy disks.

Strike ENTER when ready
or ESC to stop
```

Press <Enter>.

44. Your monitor displays:

```
Selective installation of the Remote Terminal Package
Version 2.0 database.
   0  Terminate installation
   1  Install terminfo file(s)
   2  Locate a specific terminal within terminfo file(s)
   3  Compile a SINGLE terminal entry
Enter option:
```

Enter: **1**

45. Your monitor displays:

```
Enter a file name, 'all', 'done', or 'files':
```

Enter: **all**

46. Your monitor again displays:

```
Enter a file name, 'all', 'done', or 'files':
```

Enter: **done**

47. Your monitor again displays:

```
Selective installation of the Remote Terminal Package
Version 2.0 database.
   0  Terminate installation
   1  Install terminfo file(s)
   2  Locate a specific terminal within terminfo file(s)
   3  Compile a SINGLE terminal entry
Enter option:
```

Enter: **0**

48. Remove the Remote Terminal Package diskette from the floppy diskette drive.

49. Complete the UNIX installation by running *setup* and answering the resulting questions, as appropriate for your system. Refer to the *AT&T UNIX System*

*V/386 Release 3.2 System Administrator's Guide* for instructions on how to use *setup*.

# INSTALLING TCP/IP AND ETHERNET DRIVERS

Use the following procedures to install the TCP/IP and Ethernet Driver software.

## Preliminary Preparations

1. Read through all of the following steps and make that sure you can answer all of the questions regarding your network configuration prior to beginning the installation.

2. Verify that the proper Ethernet Controller board is installed in your system.

3. Verify that UNIX System V/386 Release 3.2 Version 2.1 is installed. It must be installed prior to installing TCP/IP.

4. Verify that *setup* was run after UNIX was installed. This must be done prior to installing TCP/IP to enable the system to know its *uname*.

5. Make sure that you have the serial number and validation (activation) key pair. They can be found on a piece of paper that is shipped with the TCP/IP software.

6. If TCP/IP was installed previously, you must deinstall the TCP/IP and Ethernet Drivers Packages before installing the new software. Refer to the *AT&T UNIX System V/386 Release 3.2 System Administrator's Guide* for instructions on how to use *removepkg*.

## Installing Lachman TCP/IP

1. Login as root.

2. Enter:

   **installpkg**

   You will be installing from diskettes.

3. The system asks you to insert a TCP/IP diskette.

   Insert the TCP/IP disk labeled "Lachman TCP/IP System V/386 Release 3.2" into the diskette drive.

4. Press **<Enter>**.

5. The system asks you to insert the remaining TCP/IP diskettes.

   Insert TCP/IP diskettes 2 of 3 and 3 of 3 into the drive after the applicable prompts are displayed.

6. Answer the following questions:

**NOTE**

Information that you should enter in reply to questions that are not specific to your network configuration is printed in **Titan bold** type. If a default value is given, you can use it by pressing **<Enter>**.

```
Configure STREAMS parameters:

   Tuneable parameter "NQUEUE" is currently set to 160.
   Is it OK to change it to 384? (y/n) y

   Tuneable parameter "NSTREAM" is currently set to 40.
   Is it OK to change it to 192? (y/n) y

   Please enter your product serial number:
                (Refer to item 4 of the "Preliminary Preparations" section.)

   Please enter your product validation key:
                (Refer to item 4 of the "Preliminary Preparations" section.).

   Verify that you have entered the previous two values
   correctly.

   Do you wish to rebuild the kernel? [y/n]  n

Network Configuration:
  TCP/IP Configuration

   Enter the internet address of this host:
                                                (Example: 128.3.3.3)

   Enter the name of this network:
                                                (Example: eng)

   Is this network subnetted? (y/n) (default: n):

   Enter the broadcast address (default: ):  <Enter>.

   Enter the domain name:
                                                (Example: isc.intel.com)

   Do you want to configure a default route? (y/n)
   (default: n):  <Enter>.

   Do you wish to change any of these values? (y/n):

   Do you want to make entries for other hosts? (y/n):

   Enter the UNQUALIFIED name of the host:
                                                (Example: dragon)
```

```
Enter the name of the domain of this host (default: ):
                                        (Example: isc.intel.com)

Enter the internet address of this host:
                                        (Example: 128.3.3.4)

Is this an equivalent host? (y/n) (default: y):

Add this entry? (y/n)
```

You may answer the five previous questions as many times as you desire to add entries to the */etc/hosts* file.

7. Install the Ethernet Drivers package, as instructed in the next section.

## Installing Ethernet Drivers

**NOTE**

Your TCP/IP software must be installed before you attempt to install your Ethernet Drivers.

1. Login as root.
2. Enter:

   **installpkg**

   You will be installing from diskettes.
3. The system asks you to insert the Ethernet Drivers diskette.

   Insert the TCP/IP diskette labeled "Ethernet Drivers Release 3.2" into the diskette drive.
4. Press **<Enter>**.
5. Answer the following questions:

**NOTE**

Information that you should enter in reply to questions is printed in **Titan bold** type. If a default value is given, you can use it by pressing **<Enter>**.

```
Which ethernet board will you be using?  pc586
```

```
Do you wish to configure TCP/IP to use the pc586? [y/n] y
```

6. After the kernel has been built, the system asks you to reboot your system.

   Press **<Enter>**.

7. Reboot the system by pressing **<Ctrl, Alt, Del>**.

8. Install the iPSC/2 system software package, as instructed in the next section.

# INSTALLING THE NETWORK FILE SYSTEM OPTION

If you have the optional NFS software, use the following procedures to install it on your iPSC/2 System.

## Preliminary Preparations

1. Verify that the TCP/IP and Ethernet Drivers are installed. They must be installed prior to installing NFS.

2. Make sure that you have the serial number and validation (activation) key pair. They can be found on a piece of paper that is shipped with the NFS software.

3. If NFS was installed previously, you must deinstall it before installing the new software. Refer to the *AT&T UNIX System V/386 Release 3.2 System Administrator's Guide* for instructions on how to use *removepkg*.

## Installing NFS

1. Login as root.

2. Enter:

   **installpkg**

   You will be installing from diskettes.

3. The system asks you to insert an NFS diskette.

   Insert the NFS diskette labeled "System NFS V/386 Release 3.2" into the diskette drive.

4. Press **<Enter>**.

5. The system asks you to insert the remaining NFS diskettes.

   Insert NFS diskettes 2 of 4, 3 of 4, and 4 of 4 into the drive after the applicable prompts are displayed.

6. Answer the following questions when asked:

```
The /etc/exports file should contain a list of local file
systems that may be accessed by remote NFS hosts.  Would
you like one to be built containing all your local file
systems?

(y/n):

Please enter your product serial number:
Please enter your product validation key:

Serial number is xxxxx
Validation key is xxxxx

Is this correct? [y/n]
```

7. After the kernel has been built, the system asks you to reboot your system.

   Press **<Enter>**.

8. Reboot the system by pressing **<Ctrl, Alt, Del>**.

## INSTALLING iPSC®/2 RELEASE 3.0

The iPSC/2 system software must be installed before you can use your iPSC/2 System. Before you install your new iPSC/2 system software, you must deinstall your old iPSC/2 system software. Refer to the *AT&T UNIX System V/386 Release 3.2 System Administrator's Guide* for instructions.

**NOTE**

Your TCP/IP software must be installed before you attempt to install your iPSC/2 System Software.

1. Login as "root".
2. Enter:

   ```
   installpkg
   ```

3. Your monitor displays:

   ```
   Are you installing from tape (y/n)?
   ```

   Enter: **y**

4. Your monitor displays:

```
Insert Installation Tape in drive and press <RETURN>
```

   Insert the iPSC/2 System Software tape in the tape drive.

5. Press **<Enter>**.

6. Your monitor displays:

```
Searching Tape for list of packages. This may take awhile.
Please wait.
```

   After an interval, your monitor displays all the available packages, as follows:

```
Available Packages:

   1. iPSC Remote Host Software Package Release 3.0
   2. iPSC Simulator Software Package Release 2.1
   3. iPSC System Software Package Release 3.0

Do you want to install all of the above packages? <y/[n]>:
```

   What you install depends on what you use. If you do not have a use for Remote Host Software or for the Simulator, you can save disk space by not copying it to your system. You can install this software later.

   Enter y next to the packages you wish to install.

7. Your monitor displays:

```
Strike ENTER when ready or
ESC to stop
```

   Press **<Enter>**.

8. When your monitor prompts you to do so, reboot the system by pressing **<Ctrl, Alt, Del>**.

## BUILDING REMOTE HOST

The remote access feature of the iPSC/2 System extends the message-passing environment across the network to supported workstations. Certain Concurrent Workbench tools are also visible from the workstation.

Release 3.0 supports Sun/3, Sun/4, and Sun/386i workstations, running Sun OS 3.5 or Sun OS 4.0. To build remote host, proceed as follows:

1. Install the Remote Host Software Package as described "Installing iPSC®/2 Release 3.0" section previously.

2. Create a directory on the workstation to contain the remote host source code. This directory can be located wherever you choose.

3. Copy the remote host source code from the directory */usr/ipsc/rhost* on the SRM into the directory you created in step 2.

4. Edit the makefile in the top level remote host source directory on your workstation so that the installation directories are correct for your system.

   **IPSCDIR**--Contains daemons and named sockets. The default is */usr/ipsc* and the subdirectories *lib* and *log* will be created.

   **BINDIR**--Contains cube binaries. On the SRM these files are in */usr/bin*. Any directory can be used for the binaries, as long as the cube users have it in their search path. The default directory on the remote host is *IPSCDIR/bin*.

   **LIBDIR**--Contains libhost.a. The default location for the library is in */usr/lib*. Again any directory can be used as long as the user application makefiles reflect the correct directory.

   **INCDIR**--Contains iPSC/2 include files *cube.h* and *fcube.h*. The default is *IPSCDIR/include* .

   **NOTE**

   All pathnames must be absolute pathnames (i.e., start with "/").

5. Kill any existing *commser* and *fserver* daemons on the remote host.

6. To build binaries on your workstation:

   A. Become the super user.

   B. Change to the top level remote host source directory, using **cd**.

   C. Enter: **make**

7. To install the binaries in the prescribed directories:

   A. Login to the console as *root*

   B. Do a **cd** to the directory containing the remote host sources.

   C. Enter:

      ```
      make install
      ```

8. Edit the *srms* file (*IPSCDIR/lib/srms*) to include the names of all SRMs that the remote host may access.

9. Edit your *.cshrc* file and insert:

```
setenv TTY `tty` >& /dev/null
```

# CONCURRENT FILE SYSTEM™ 3

## INTRODUCTION

The iPSC/2 Concurrent File System™ can be a very useful addition to your concurrent environment. Here are some suggestions on using your Concurrent File System and keeping it healthy.

## CREATING A CONCURRENT FILE SYSTEM™

1. Login as root and partially boot the cube by typing:

   ```
   bootcube -D14
   ```

2. Use the CFS creation utility by typing:

   ```
   mkcfs
   ```

   This command asks some questions about the file system name and asks you to confirm that you want to destroy any existing files.

3. When *mkcfs* is done, finish the bootcube process by typing:

   ```
   bootcube -D15-15
   ```

   This starts the CFS processes. CFS is now ready for use.

## DEALING WITH DCHK AND FCHK FAILURES

When CFS is reinitialized at bootcube, directory and file consistency checkers are run to be sure that all is well in the file system before starting it up. Because there is no file fixing utility in CFS like the UNIX *fsck*, this must be done by hand. The easiest way to fix things if *fchk* fails is to remove the offending file or directory and try again. This can be painful if it is an important file. Be sure to back up anything important.

## BACKING UP YOUR FILES

**Be sure to keep current backups of any important files.** Use the CFS *backup* facility to back up all of the CFS, or specific disks. To back up specific files, use *tar* to put them on tape or *cp* to copy the files into your host file system.

When you use *backup* and *restore*, CFS must be fully started. Because this allows others to access CFS while you are trying to back it up, you should do a *getcube* to get all the nodes. This, however, does not prevent a user who allocates a cube with zero nodes from accessing the disks.

When you archive specific files to tape, the *star* command is usually faster than *tar*. It speeds tape access by using *stream* to gather the *tar* output into large buffers before writing to the tape. When you do a restore, *star* reads large buffers before unarchiving, again to speed up tape access.

## LOADING EXECUTABLES

Node executables can be stored in and loaded from the Concurrent File System. Not only is loading from CFS faster, but it also allows you to save space on your host file system. To load node executables, use the *load /cfs/bin/nodeprog* command or the *load("/cfs/bin/nodeprog", ...)* call, from either the node or the host.

## REMOTE HOST

To use the node shell (*nsh*) from a remote host workstation, you must copy all of the commands to CFS:

1. Login to the SRM.
2. Perform the following sequence to copy the commands to CFS:

```
getcube
nsh
cd /usr/ipsc/bin
mkdir /cfs/bin
cp * /cfs/bin
exit
relcube
```

Once you have copied the commands, you must add the following to the *.cshrc.ipsc* file in your home directory on the remote host:

```
set path=/cfs/bin
set shell=/cfs/bin/csh
```

# CFS ROUTINE SYNCHRONIZATION

Some of the library routines used to access the CFS occasionally perform some communication and/or synchronization. The following table lists the routines and shows when (or if) they do such synchronization.

| Routine | Conditions Causing the Routine to Synchronize |
|---|---|
| *close* | Mode 1, 2, or 3 |
| *setiomode* | Always |
| *lseek* | Mode 2 |
| *iseof* | Mode 2 |
| *cread* | Mode 2 |
| *cwrite* | Mode 2 |
| *iread* | Mode 2 |
| *iwrite* | Mode 2 |

Mode 3 is a synchronized mode. Routines *lseek*, *iseof*, *cread*, *cwrite*, *iread*, and *iwrite* do not perform any synchronization in Mode 3. The high performance provided by Mode 3 is attributable to this fact. Therefore, the following information from page 3-21 of the *iPSC/2 User's Guide* is not strictly true:

> "All the nodes in your cube must open the file and perform the very same operation (a read or write) on the file. For example, if a node makes a second read request, *it will wait until all the other nodes have completed their first reads.*"

The text in *italics* is not true for Mode 3.

Mode 3 must be used in a synchronized manner, even though reads and writes do not interact between nodes. All nodes must make the same calls to guarantee correct operation. Failure to do so will probably cause a run time error.

# CFS MEMORY REQUIREMENTS

All nodes involved in CFS must have at least 4 megabytes of memory. This includes Node 0 of the compute cube. For sites that currently have 1-megabyte node boards and want to add CFS to their system, a 4-megabyte replacement board for Node 0 will be included in the CFS upgrade.

# MISCELLANEOUS

To boot your cube without starting CFS, use the command *bootcube -o*.

The current CFS file size limit is $2^{31} - 1$ bytes, or 2,147,483,647 bytes, which is just under 2 gigabytes.

# REMOTE HOST 4

## RUNNING REMOTE HOST SOFTWARE

To use the cube from a remote host workstation, you must start a remote *commser* on the workstation:

1. Login as *root*.

2. To start a *commser*, execute *bootcube* or *rebootcube* on the remote host.

Once the remote *commser* is running, you can get cubes and run cube applications.

## COMPILING REMOTE HOST APPLICATIONS

To compile existing iPSC/2 SRM applications, the makefiles must be modified as follows:

1. If **LIBDIR** is defined to be either */lib* or */usr/lib*, replace the **-host** switch with **-lhost**. Otherwise, the entire path of the host library must be specified.

   For example: If **LIBDIR** is defined to be */usr/myusername/lib*, you must link with */usr/myusername/lib/libhost.a*.

2. Change the node compilation rule for *cc* and *f77* to *rcc* and *rf77*, respectively.

3. Change *ld*, *ar*, and *as* to *rld*, *rar*, and *ras*, respectively. Refer to the *iPSC®/2 Programmer's Reference Manual* for details.

**NOTE**

The Sun workstation and the iPSC/2 System have different internal representations of integer and floating point numbers. See the descriptions of the *createstruc* and *CTOH* utilities in the *iPSC®/2 Programmer's Reference Manual* for information on converting messages to and from cube byte order.

If you have a Concurrent I/O System, refer to the "Suggestions for Using CFS" section for more information.

# USER INFORMATION 5

## BOOTING STANDALONE UNIX

To boot UNIX from the boot floppy diskette:

1. Insert the floppy diskette in the floppy diskette drive.
2. Reset the CPU.
3. Your monitor displays:

   ```
   Strike ENTER to install the UNIX System on your hard disk
   ```

   Press <Ctrl,\>.
4. Your monitor displays a single user shell prompt: #
5. To shutdown from the single user mode, enter:

   ```
   sync; sync; sync; uadmin 2 0.
   ```

## NODE MEMORY USAGE

The following table shows the amount of free memory in each node when no application processes are loaded. It also shows the size of the operating system and short message buffers. Short message buffers are used to hold system control messages and application messages up to 100 bytes long.

| Node type | Free memory | | NX/2 code | | NX/2 data | Number of message buffers* | | Message buffer size | | Total node memory |
|---|---|---|---|---|---|---|---|---|---|---|
| 1M | 651,264 | + | 113,408 | + | 181,504 | 512 | + | 102,400 | = | 1,048,576 |
| 4M | 3,764,224 | + | 113,408 | + | 189,696 | 634 | + | 126,976 | = | 4,194,304 |
| 8M | 7,847,936 | + | 113,408 | + | 197,888 | 1,146 | + | 229,376 | = | 8,388,608 |

* Not part of node memory total

To calculate the size of an application process, use the UNIX *size* command to obtain code, data, and bss sizes. Round the code size up to a multiple of 4,096, then add 4,096 for each 4,194,304 or less of the result. Add the data and bss sizes, round the result up to a multiple of 4,096, and again add 4,096 for each 4,194,304 or less of that result. Add an additional 12,288 for the stack and other overhead to the other subtotals to obtain the total memory required. For example:

Size Node:
101272 + 46912 + 521280 = 669464

| Calculation: | Size (rounded to 4,096) | Overhead | |
|---|---|---|---|
| Code (101,272): | 102,400 | + 4,096 + | |
| Data + bss (568,192): | 569,344 | + 4,096 + | |
| Stack and other: | | 12,288 = | 692,224 |

This program would require a 4M node.

## SUGGESTIONS FOR OPTIMIZING MESSAGE PASSING PERFORMANCE

There are several techniques that can be used in an application to reduce message-passing overhead. Some of these techniques are good programming practice for the iPSC/2 System and should be used in all applications. Other techniques are useful only when it is necessary to get the very best message passing performance -- for example, when optimizing a critical section of an application.

The most important programming practice for the iPSC/2 System is to ensure that send and receive buffers are properly aligned and sized whenever possible. Although the message-passing system calls will work with any size or alignment of buffers, the hardware only works with well-aligned buffers. As a result, the software must copy messages which are in misaligned buffers, decreasing performance.

Send buffers can be any size but should be aligned on a 4-byte boundary if the message is longer than 100 bytes. Messages smaller than 100 bytes may also benefit slightly from 4-byte buffer alignment. Receive buffers should always be aligned on a 4-byte boundary and should be a multiple of 4 bytes. In addition, receive buffers should be no more than 4096 bytes larger than the message that will be received in them. Note that the size of a buffer refers to the length parameter in a send or receive call. The buffer may be declared to be larger than the length if desired.

While these rules may seem strict, they are easy to follow because the compilers tend to align data on appropriate boundaries. In Fortran, avoid using CHARACTER or LOGICAL*1 buffers if INTEGER, REAL, or DOUBLE PRECISION will do. If you have performance-sensitive CHARACTER or LOGICAL*1 buffers, equivalence them to an INTEGER and try to make them a multiple of 4 bytes. In C, declare buffers as *int* or *long* rather than *char* or *short*, or if it is more convenient to declare the buffer *char* or *short*, make sure the size is a multiple of 4 bytes, and that other *char* or *short* arrays are also multiples of 4 bytes. Buffers allocated with *malloc* or its derivatives will be correctly aligned.

Another useful technique is to arrange the application so that the receive call for a critical message is posted before the message comes in. In some cases, this will

require the use of *irecv* to post a receive before proceeding with a lengthy computation. The computation can then overlap with message reception.

In cases where it is difficult to overlap computation with message passing, use the simpler *csend* and *crecv* calls instead of the more complex *isend* and *irecv*. *csend* and *crecv* are not only easier to use, but they also have slightly less overhead because their data structures in the system are preallocated and they require fewer system calls to operate.

The techniques described above have the most impact on message-passing performance and are generally good practice. Below are additional tricks that can be used to gain slight additional speed, but at a cost that may not justify their use. Some of these techniques apply to C only.

The regular system calls (such as *csend*) check for errors by calling the underscore version (such as *__csend*) and then inspecting the return value. You can save a little time by calling the underscore version directly (it takes the same parameters) and ignoring any errors. Of course, this technique should only be used on fully debugged programs.

Avoid repeated use of calls like *mynode* and *mypid* in send and receive calls. Instead, declare variables for node and pid and set those variables to *mynode* and *mypid* at the beginning of the application. Use the variables in send and receive calls.

You may obtain a small performance increase for large messages by making the buffers static and avoiding a call to *malloc*.

There is a tiny overhead associated with running the green LED on the node board. This overhead can be eliminated by calling *led* once at the beginning of the program to transfer control of the LED from the system to the application.

# PROCEDURE FOR REMOVING ROOT'S PASSWORD

Perform the following steps if you forget root's password:

1. Make sure no one is logged into the system.
2. If it is possible to log in, log in as any user and enter:

   ```
   sync; sync
   ```

3. Logout.
4. Insert the diskette labeled "UNIX Boot Floppy" into the floppy diskette drive.
5. Turn the machine off, wait 10 seconds, and turn it back on.
6. The following message appears:

   ```
   Please wait while existing file systems are checked for . .
   ```

   Press < Ctrl, \> to get the command prompt.

7. Clean the file system by entering:

```
fsck /dev/dsk/0s1
```

If the system reports that modifications were made, run **fsck** again by entering:

```
fsck /dev/dsk/0s1
```

Repeat until no changes are reported.

8. Enter:

```
mount /dev/dsk/0s1 /mnt
```

9. Save a copy of the existing password and shadow files. This is a precaution against an editing mistake. Issue the following commands:

```
cp /mnt/etc/passwd /mnt/etc/passwd.sav
cp /mnt/etc/shadow /mnt/etc/shadow.sav
```

10. Remove the password from root as follows:

```
ed /mnt/etc/shadow          (Edit the shadow file. The number of bytes
                                 in the file appears on the screen.)
1p          (Prints the first line of the file, which should be the root entry.)
s/root:*:/root::                    (Removes the root password.)
w                          (Writes the file and shows the number of lines.)
q                                          (Quits the editor.)
ed /mnt/etc/passwd          (Edit the passwd file. The number of bytes
                                 in the file appears on the screen.)
1p          (Prints the first line of the file, which should be the root entry.)
s/x//                               (Removes the root password.)
w                          (Writes the file and shows the number of lines.)
q                                          (Quits the editor.)
```

11. Enter:

```
sync; sync
```

12. Enter:

```
umount /mnt
```

13. Enter:

```
sync; sync; uadmin 2 0
```

To shut down the system.

14. When the diskette light turns off, remove the boot diskette from the drive.

15. Reboot the UNIX operating system by pressing **<Ctrl, Alt, Del>**.

# COMPILING AND USING iPSC®/1 PROGRAMS ON THE iPSC®/2

Source programs created for an iPSC/1 System are not directly compatible with the current iPSC/2 software. We recommend that you convert your iPSC/1 code to iPSC/2 code rather than using the compatibility libraries for optimum performance. However, you may compile, link, and run iPSC/1 programs on the iPSC/2 System if the following procedures are used:

1. Use the following procedure to compile and link your iPSC/1 applications with iPSC/2 software:

   To compile and link your C host program, enter:

   ```
   cc -o host host.c /usr/ipsc/lib/chost.a -host
   ```

   To compile and link your C node program, enter:

   ```
   cc -o node node.c /usr/ipsc/lib/Llibcnode.a -node
   ```

   To compile and link your Fortran host program, enter:

   ```
   f77 -o host host.f /usr/ipsc/lib/fhost.a -host
   ```

   To compile and link your Fortran node program, enter:

   ```
   f77 -o node node.f /usr/ipsc/lib/Llibfnode.a -node
   ```

2. Because the Fortran compiler has changed from the iPSC/1 (Ryan McFarland) to the iPSC/2 (Green Hills) version, the definition files for your Fortran programs have changed from *rmfhost.def* and *rmfnode.def* to *fhost.def* and *fnode.def*, respectively.

3. To use the *ccommutl* and *fcommutl* libraries, be sure to link them into your program.

   Example:

   ```
   f77 -o host host.f /usr/ipsc/lib/fcommutl.a
   /usr/ipsc/lib/fhost.a -host
   ```

4. Host programs must specify a process ID to the system before any other calls to the system are made. Therefore, when using the iPSC/1 library, you should call *copen* with a process ID before making any other calls to the library. Subsequent *copen* calls should use the same process ID. (Other process IDs will be ignored.)

5. In the iPSC/2 environment, the size of an integer variable is 4 bytes. This may impact your message length, because integers in C are 2 bytes in the iPSC/1 environment.

6. The *cubelog* command is no longer supported. To send output from your node and host programs to the same file, use the following:

```
getcube > logfile
host | syslog
```

For more information on redirecting output, refer to the *iPSC®/2 Programmer's Reference Manual.*

7. Some iPSC/1 commands are no longer supported, and many have changed. (Refer to the "Command Summary" table that follows). Refer to the *iPSC®/2 Programmer's Reference Manual* for a description of the iPSC/2 commands. Refer to the *iPSC®/2 System Administrator's Guide* for a description of *bootcube*.

**Command Summary**

| iPSC/1 | iPSC/2 | Description |
|---|---|---|
| *cubeinit* | *bootcube* | Reset a cube. |
| *cubelog* | *syslog* | Send output to a logfile. |
| *getcube* | *getcube* | Get a cube. |
| *load* | *load* | Load a process into a cube. |
| *load -c* | *bootcube* | Download a new NX/2. |
| *loadkill* | *killcube* | Kill a cube process. |
| *loadstart* | *startcube* | Start a cube process. |
| *loadwait* | *waitcube* | Wait for a cube process to finish. |