October 1990
Order Number: 311532-006

iPSC®/2 and iPSC®/860
USER'S GUIDE

intel® Corporation

REV.	REVISION HISTORY	DATE
-001	Original Issue	12/87
-002	Revision	03/88
-003	Revision	03/89
-004	Revision	10/89
-005	Preliminary	12/89
-006	Revision	06/90
—	Revised by Change Notice 312003-001	10/90

RESTRICTED RIGHTS

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

CHAPTER 2
USING iPSC® SYSTEM COMMANDS

CHAPTER 3
USING iPSC® SYSTEM CALLS

CHAPTER 4
DESIGNING A CONCURRENT APPLICATION

CHAPTER 5
DEBUGGING

CHAPTER 6
PROGRAMMING TECHNIQUES

CHAPTER 7
THE CONCURRENT FILE SYSTEM

CHAPTER 8
TCP/IP ON THE NODES

CHAPTER 9
THE X WINDOW SYSTEM ON THE iPSC®/860

APPENDIX A
iPSC® SYSTEM COMMANDS, SYSTEM CALLS, and ROUTINES

APPENDIX B
iPSC® SYSTEM FEATURES

APPENDIX C
iPSC® SYSTEM SPECIFICATIONS

APPENDIX D
DECON EXAMPLE SOURCE FILES

APPENDIX E
TCP/IP SYSTEM CALLS

LIST OF ILLUSTRATIONS

LIST OF TABLES

COMPILING AND LINKING A C APPLICATION

This section explains how to compile and link a C application program to run on the host and the nodes.

First, be sure to include the file *cube.h* in both host and node programs. This file resides in */usr/include* on the SRM. Include it as `#include <cube.h>`.

Compiling a Host Program

When compiling a host program on the SRM, include the **-host** option on the **cc** invocation line. For example, consider the pi example. It has three source files: *host.c*, *prompt.c*, and *node.c*. The files *host.c* and *prompt.c* are compiled and linked to make the executable host program called *host* as follows:

```
% cc -o host host.c prompt.c -host
```

Compiling a Node Program

When compiling a node program on the SRM, include the **-node** option on the **cc** invocation line. Once again consider the pi example. The file *node.c* is compiled and linked to make the executable node program as follows:

```
% cc -o node node.c -node
```

If the node program is intended for an RX node, be sure to include the **cc** compiler's **-i860** option.

```
% cc -o node node.c -i860 -node
```

Or, you can use the **pgcc** compiler:

```
% pgcc -o node node.c -node
```

If the node program is intended for a CX node and makes use of an SX processor, include the **-sx** option. Do not include the **-sx** option if you do not have an SX processor. That's because if you don't have the SX option, you do have an Intel 80387 numeric coprocessor. The SX processor and the Intel 80387 numeric coprocessor use different registers to return their results, and an executable program should contain compiled modules for only one of the two. For example, if your nodes had SX processors and you were doing floating-point calculations, the **cc** invocation line would look as follows:

```
% cc -o node node.c -sx -node
```

If the node program uses a VX processor, also include the **-vx** and either the **-vec** or **-vecdb** option. The **-vx** and **-vecdb** switches are inappropriate for host programs.

The -vx option brings in the interface library for the vector board. The -vec option brings in the library of vector routines that execute on the VX processor, the SX processor, or the 80387 numeric coprocessor.

The **-vecdb** switch is like **-vec**, except that it brings in a debug version of the vector library routines. This debug version performs some additional testing such as checking for proper data types and alignment.

You can combine **-sx** and **-vx** switches. For example, the following Fortran compilation command makes an executable for a node that has VX and SX processors:

```
% cc -o node_tst node_tst.c -sx -vx -vec -node
```

If you are compiling node program *node.c* for use on an RX node with TCP/IP libraries, issue one of the following command lines on the SRM:

```
% cc -o node node.c -lsocknode -node -i860

% pgcc -o node node.c -lsocknode -node
```

Refer to the Chapter 8 for more information on the version of TCP/IP that runs on RX nodes.

Table 2-1 summarizes the **cc** compile options for node programs. Table 2-2 summarizes the **pgcc** compile options for node programs.

Table 2-1. The cc Options for Node Programs

cc Options	Description
-node	386-based node that has an 80387 numeric coprocessor (CX node).
-sx -node	386-based node that has an SX processor (SX node).
-vec -node	386-based node that has an 80387 numeric coprocessor and uses the 80387 version of the vector routines (CX node).
-sx -vec -node	386-based node that has SX processor and uses the vector routines that run on the SX processor (SX node).
-vx -vec -node	386-based node that has an 80387 numeric coprocessor and a VX processor (VX node).
-sx -vx -vec -node	386-based node that has both SX and VX processors (SXVX node)
-vx -vecdb -node	386-based node that has an 80387 numeric coprocessor and a VX processor and uses debug vector routines (VX node).
-sx -vx -vecdb -node	386-based node that has both SX and VX processors and uses debug vector routines (SXVX node).
-i860 -node	i860-based node (RX node).
-i860 -lvec -node	i860-based node that uses the i860 version of the vector routines (RX node). There are no name clashes between the i860 and the 386 vector libraries (VecLib).
-lsocknode -node	Links your program to all the socket libraries used by node TCP/IP.

Table 2-2. The pgcc Options for Node Programs

pgcc Options	Description
-node	i860-based node (RX node).
-lvec -node	i860-based node that uses the i860 version of the vector routines (RX node). There are no name clashes between the i860 and the 386 vector libraries (VecLib).
-lsocknode -node	Links your program to all the socket libraries used by node TCP/IP.

When they are required, the **-i860**, **-sx**, and **-vx** switches must appear in both the compile and link steps. The **-vec**, **- lvec**, **-vecdb**, **-node**,**-host**, and **-lsocknode** switches are needed only on the link step. The **-i860**, **-sx**, **-vx**, and **-node** flags result in the definitions of preprocessor symbols _ _ i860, _ _SX, _ _VX, and _ _NODE, respectively. These can be used for conditional compilation in programs that run in multiple environments. The symbol _ _i386 is defined by default when **-i860** is not specified (cc only).

For example, consider the following makefile. It is set up to enable you to compile the pi example used in this manual for CX, SX, or RX nodes, simply by invoking **make** with the appropriate argument.

```
#
# This file is used to compile and link the host.c, prompt.c,
# and node.c files for the pi numerical integration example.
#

help:
        @echo
        @echo "You must specify the type of node you wish to build a node"
        @echo "executable for, choose one of the following:"
        @echo
        @echo "      make cx    (for 386 nodes with 387 coprocessors)"
        @echo "      make sx    (for 386 nodes with SX coprocessors)"
        @echo "      make rx    (for i860 nodes)"
        @echo

cx:     host node          #Use default compile and link flags

sx:
        make host
        make "CFLAGS=-O -sx" "LDFLAGS=-sx" node

rx:
        make host
        make "CFLAGS=-O -i860" "LDFLAGS=-i860" node

host:   host.o prompt.o
        cc -o host host.o prompt.o -host

node:   node.o fx.o
        cc -o node node.o fx.o $(LDFLAGS) -node

clean:
        rm host node host.o node.o prompt.o
```

Assume that you want to compile the pi example for use on SX nodes. You must ensure that the **-sx** appears both in the predefined macro CFLAGS and on the cc line. **make**'s implicit rules bring in CFLAGS on the compile step. The **makefile** macro LDFLAGS explicitly places the **-sx** on the **cc** line for the link step.

Consider, also, the following makefile, which does the same thing using the **pgcc** compiler.

```
#
# This file is used to compile and link the host.c, prompt.c,
# and node.c files for the pi numerical integration example.
#
help:
        @echo
        @echo "You must specify the type of node you wish to build a node"
        @echo "executable for, choose one of the following:"
        @echo
        @echo "      make cx    (for 386 nodes with 387 coprocessors)"
        @echo "      make sx    (for 386 nodes with SX coprocessors)"
        @echo "      make rx    (for i860 nodes)"
        @echo

cx:
        make "COMPILER=cc" host
        make "COMPILER=cc" node

sx:
        make "COMPILER=cc" host
        make "COMPILER=cc" "CFLAGS=-O -sx" node

rx:
        make "COMPILER=cc" host
        make "COMPILER=pgcc" "CFLAGS=-O" node

host:   host.o prompt.o
        cc -o host host.o prompt.o -host

node:   node.o fx.o
        $(COMPILER) $(CFLAGS) -o node node.o fx.o -node

clean:
        rm host node host.o node.o prompt.o

.c.o:
        $(COMPILER) $(CFLAGS) -c $<
```

Compiling on the Remote Host

When running on a remote host, use the remote host's C compiler to compile the host code. Be sure to use the **-lhost** switch and not the **-host** switch.

Use **rcc** to compile the node program. The remote C compiler **rcc** copies the node source files to the SRM, compiles and links this file on the SRM, and sends the resulting executable file back to the remote host. If you have a network of iPSC systems, use the **-h** switch to choose a particular SRM. Otherwise **rcc** chooses the first available SRM listed in the *srms* file. For example, to choose the SRM called *jaxom*, issue the compile command:

```
% rcc -h jaxom -o node node.c -node
```

Including the **-cpp** option with **rcc** causes the C preprocessor to run on the remote host before the node source is copied to the SRM. This affects how include files are resolved.

- Without **-cpp** all include files are resolved on the SRM.
- With **-cpp**, files delimited with quotation marks are resolved on the remote host. For example,

  ```
  #include "rhost.inc"
  ```

  looks in the current directory for the file *rhost.inc* while

  ```
  #include "/usr/ipsc/include/rhost.inc"
  ```

  looks in the directory */usr/ipsc/include* on the remote host.
- The **-I** switch defines a prefix for include files. For example, the compilation line,

  ```
  % rcc -cpp -I/usr/ipsc/include -o node node.c -node
  ```

  sets the include prefix to */usr/ipsc/include*. Then,

  ```
  #include "rhost.inc"
  ```

  in the source file resolves as */usr/ipsc/include/rhost.inc* on the remote host. If an include file is not found on the remote host, the prefix is used to search for the file on the SRM.
- Even with **-cpp**, all *top-level* include files delimited with < > are resolved on the SRM. However, if an include file delimited with < > is *in* a file resolved on the remote host (that is, it is not a top-level include file), then **rcc** looks for the file in */usr/include* on the remote host.

For example, consider the following situation. The node source code contains the line `#include "rhost.inc"` and the file *rhost.inc* contains the line `#include <suntool.h>`. Because *rhost.inc* is in quotation marks, **rcc -cpp** resolves it on the remote host, and because the line `#include <suntool.h>` is in *rhost.inc*, *suntool.h* is not a top-level include file. The result is that **rcc** looks for *suntool.h* in */usr/include* on the remote host. If the line `#include <suntool.h>` were in the node source file, *suntool.h* would be a top-level include file and its delimiting <> would cause **rcc** to look for it on the SRM.

COMPILING AND LINKING A FORTRAN APPLICATION

This section explains how to compile and link a Fortran application program to run on the host and the nodes.

First be sure to include the file *fcube.h* in both host and node programs. This file resides in */usr/include* on the SRM. Include it as `include '/usr/include/fcube.h'`.

Delimit include files with single quotes (`''`). The compiler **f77** uses the quoted string as a pathname for the include file. For example:

```
include 'myfile.inc'
```

This example tells **f77** to look in the current directory for *myfile.inc*.

If the file to be compiled is called *node.F* (with an uppercase *F* extension, rather than a lowercase *f*), **f77** will run the C preprocessor on the file. This enables you to use lines like the following in a Fortran program:

```
#include <fcube.h>

#define MAX 87
```

Compiling a Host Program

When compiling a host program on the SRM, include the **-host** option on the **f77** invocation line. For example, consider the pi example. It has four source files: *host.f*, *prompt.f*, *fx.f*, and *node.f*. The files *host.f* and *prompt.f* are compiled and linked to make the executable host program called *host* as follows:

```
% f77 -o host host.f prompt.f -host
```

By including **-vec** you can use the vector routines that run on the SRM's 80387 numeric coprocessor.

Compiling a Node Program

When compiling a node program on the SRM, include the **-node** option on the **f77** invocation line. Once again consider the pi example. The files *node.f* and *fx.f* are compiled and linked to make an executable node program for a CX node as follows:

```
% f77 -o node node.f fx.f -node
```

If the node program is intended for an RX node, be sure to include the **f77** compiler's **-i860** option.

```
% f77 -o node node.f fx.f -i860 -node
```

Or, you can use the **pgf77** compiler:

```
% pgf77 -o node node.f fx.f -node
```

If the node program is intended for a CX node and makes use of an SX processor, include the **-sx** option. Do not include the **-sx** option if you do not have an SX processor. That's because if you don't have the SX option, you do have an Intel 80387 numeric coprocessor. The SX processor and the Intel 80387 numeric coprocessor use different registers to return their results, and an executable program should contain compiled modules for only one of the two.

For example, if your nodes had SX processors and you were doing floating-point calculations, the f77 invocation line would look as follows:

```
% f77 -o node node.f -sx -node
```

If the node program uses a VX processor, also include the **-vx** and either the **-vec** or **-vecdb** option. The **-vx** and **-vecdb** switches are inappropriate for host programs.

The **-vx** option brings in the interface library for the vector board. The **-vec** option brings in the library of vector routines that execute on the VX processor, the SX processor, or the 80387 numeric coprocessor.

The **-vecdb** switch is like **-vec**, except that it brings in a debug version of the vector library routines. This debug version performs some additional testing such as checking for proper data types and alignment.

You can combine **-sx** and **-vx** switches. For example, the following Fortran compilation command makes an executable for a node that has VX and SX processors:

```
% f77 -o node_tst node_tst.f -sx -vx -vec -node
```

Table 2-3 summarizes the **f77** compile options for node programs.Table 2-4 summarizes the **pgf77** compile options for node programs.When they are required, the **-i860**, **-sx**, and **-vx** switches must appear in both the compile and link steps. The **-vec**, **- lvec**, **-vecdb**, **-node**,**-host**, and **-lsocknode** switches are needed only on the link step. The **-i860**, **-sx**, **-vx**, and **-node** flags result in the definitions of preprocessor symbols _ _ i860, _ _SX, _ _VX, and _ _NODE, RESPECTIVELY. These can be used for conditional compilation in programs that run in multiple environments. The symbol _ _i386 is defined by default when **-i860** is not specified (**f77** only).

Table 2-3. The f77 Options for Node Programs

f77 Options	Description
-node	386-based node that has an 80387 numeric coprocessor (CX node).
-sx -node	386-based node that has an SX processor (SX node).
-vec -node	386-based node that has an 80387 numeric coprocessor and uses the 80387 version of the vector routines (CX node).
-sx -vec -node	386-based node that has SX processor and uses the vector routines that run on the SX processor (SX node).
-vx -vec -node	386-based node that has an 80387 numeric coprocessor and a VX processor (VX node).
-sx -vx -vec -node	386-based node that has both SX and VX processors (SXVX node)
-vx -vecdb -node	386-based node that has an 80387 numeric coprocessor and a VX processor and uses debug vector routines (VX node).
-sx -vx -vecdb -node	386-based node that has both SX and VX processors and uses debug vector routines (SXVX node).
-i860 -node	i860-based node (RX node).
-i860 -lvec -node	i860-based node that uses the i860 version of the vector routines (RX node).

Table 2-4. The pgf77 Options for Node Programs

pgf77 Options	Description
-node	i860-based node (RX node).
-lvec -node	i860-based node that uses the i860 version of the vector routines (RX node).

```
# This file is used to compile and link the host.f, prompt.f,
# fx.f, and node.f files for the pi numerical integration example.
#

help:
        @echo
        @echo "You must specify the type of node you wish to build a node"
        @echo "executable for, choose one of the following:"
        @echo
        @echo "      make cx    (for 386 nodes with 387 coprocessors)"
        @echo "      make sx    (for 386 nodes with SX coprocessors)"
        @echo "      make rx    (for i860 nodes)"
        @echo

cx:     host node       #Use default compile and link flags

sx:
        make host
        make "F77FLAGS=-sx" "FLDFLAGS=-sx" node

rx:
        make host
        make "F77FLAGS=-i860" "FLDFLAGS=-i860" node

host:   host.o prompt.o
        f77 -o host host.o prompt.o -host

node:   node.o fx.o
        f77 -o node node.o fx.o $(FLDFLAGS) -node

clean:
        rm host node host.o node.o prompt.o fx.o
```

Assume that you want to compile the pi example for use on SX nodes. You must ensure that the **-sx** appears both in the predefined macro F77FLAGS and on the cc line. **make**'s implicit rules bring in F77FLAGS on the compile step. The **makefile** macro FLDFLAGS explicitly places the **-sx** on the cc line for the link step.

Consider, also, the following makefile, which does the same thing using the **pgf77** compiler.

```
#
# This file is used to compile and link the host.f, prompt.f,
# and node.f files for the pi numerical integration example.
#
help:
        @echo
        @echo "You must specify the type of node you wish to build a node"
        @echo "executable for, choose one of the following:"
        @echo
        @echo "      make cx    (for 386 nodes with 387 coprocessors)"
        @echo "      make sx    (for 386 nodes with SX coprocessors)"
        @echo "      make rx    (for i860 nodes)"
        @echo

cx:
        make "COMPILER=f77" host
        make "COMPILER=f77" node

sx:
        make "COMPILER=f77" host
        make "COMPILER=f77" "FFLAGS=-O -sx" node

rx:
        make "COMPILER=f77" host
        make "COMPILER=pgf77" "FFLAGS=-O" node

host:   host.o prompt.o
        f77 -o host host.o prompt.o -host

node:   node.o fx.o
        $(COMPILER) $(FFLAGS) -o node node.o fx.o -node

clean:
        rm host node host.o node.o prompt.o

.f.o:
        $(COMPILER) $(FFLAGS) -c $<
```

The example presented later in this chapter has the node program accept the name of its iphost as a command line argument, provided when you loaded it with the **load** command. For example, assume that you load the program *node* as

```
% load node iphostname
```

Then, the node program can attach to *iphostname* by executing the call

```
setiphost(argv[1]);
```

NODE TCP/IP SPECIFICS

This section describes the particular version of TCP/IP that is supported on the nodes.

Addressing Scheme and Socket Type

The TCP/IP that is available to node programs follows the AF_INET addressing scheme. This addressing scheme uses Internet addresses to identify the machine. Port numbers allow more than one socket on a machine. Also, the node TCP/IP uses only the SOCK_STREAM type of sockets.

The consequence is that when you create a socket with the **socket()** call, you must specify AF_INET and SOCK_STREAM. Also, be sure to use the Internet socket structure, `struct sockaddr_in`. This structure is defined in */usr/include/CMC/netinet/in.h*.

Finally, by default sockets created for use with node TCP/IP are blocking sockets. When you issue an **accept()** call, your program blocks until someone makes a connection. You can make sockets nonblocking with **fcntl()**.

Include Files

Include files reside by default on the SRM under */usr/include/ipsc/CMC*. The directory structure under *CMC* is as you would expect for TCP/IP. For example, if your makefile specifies the **-I** option as *-I/usr/include/ipsc*, then to include *socket.h*, your program should contain the line,

```
#include <CMC/sys/socket.h>
```

There is one exception, however. When you use the byte order calls (such as **htons()**), you must also include */usr/include/ipsc/CMC/ntoh.h*. Under the UNIX operating system, these calls require only the header files, */usr/include/ipsc/sys/types.h* and */usr/include/ipsc/sys/in.h*. Under the NX operating system, they also require */usr/include/ipsc/CMC/ntoh.h*.

Server/Client Relationship

TCP/IP convention is to have the server do the **accept()**. The server doesn't have previous knowledge of its caller. It gets that information from the **accept()**. A client on the other hand must know from whom to request service. It must put that information in the **connect()** call.

A node can be a server or a client. However, in an iPSC system, the nodes are usually considered to be a compute server, and the host is the client. When you use TCP/IP to send data between a node and the host, the node is usually the acceptor and the host is the connector.

TCP/IP convention is to have the server **fork()** a child to handle a client request after a successful **accept()**. If you do this in an RX node, the NX operating system loads the child on another node in the cube. For this to be successful, you must have at least one node in your allocated cube that does not have a process running on it. You also might consider writing your application so that it does not **fork()** processes to handle requests.

If a name server is configured for the SRM, the iPSC system will use it. The calls **sethostent()** and **endhostent()** are available to open and close the */etc/hosts* file, but **gethostent()** is not provided. You have to use either **gethostbyname()** or **gethostbyaddr()** to access the file.

Also, when you use **gethostbyname()**, note that the argument is the name of the iphost, not the name of the SRM.

COMPILING A NODE PROGRAM WITH TCP/IP

The TCP/IP calls are in *libsocknode.a*. Include the switch **-lsocknode** on the linking step of your compile. For example, to compile the program *node.c* for use on an RX node with the TCP/IP libraries, issue one of the following commands on the SRM:

```
% cc -o node node.c -lsocknode -i860 -node

% pgcc -o node node.c -lsocknode -node
```

If you are developing on the remote host, use **rcc** instead of **cc**. If you use a makefile, the pertinent lines are as follows:

- For the **cc** compiler:

```
CFLAGS= -i860
node: node.o
      cc -o node node.o -lsocknode -i860 -node
host: host.c
      cc -o host host.c -host
```

- For the **pgcc** compiler:

```
node: node.o
      pgcc -o node node.o -lsocknode -node
host: host.c
      cc -o host host.c -host
```

Note that you may want to put **-node** as part of CFLAGS. The built-in rule for generating a C object from a C source uses CFLAGS. Putting **-node** into CFLAGS would include **-node** on the compilation step. This results in the definition of the preprocessor symbol __NODE, which your application may find useful.

INDEX

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W