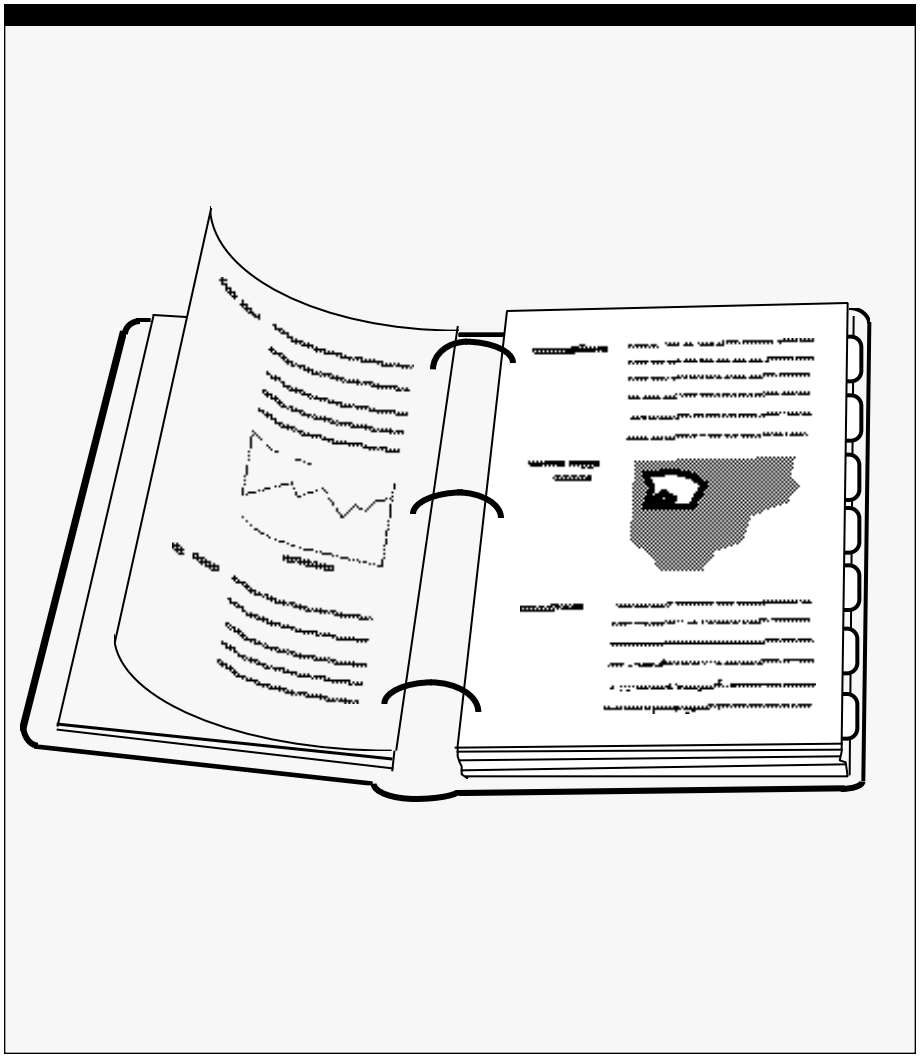
VMS/HP 1000

Interface

User's

Guide

SQUARE D COMPANY
CRISP AUTOMATION SYSTEMS



VMS/HP 1000 Interface User's Guide

CRISP® is a registered trademark of Square D Company

I/ONYX® is a registered trademark of Square D Company

Dedicated to Growth
Committed to Quality

VMS/HP 1000 Interface User's Guide

5160 Paul G. Blazer Memorial Parkway
Dublin, Ohio 43017
USA

(614) 764-4200

Dedicated to Growth
Committed to Quality

VMS/HP 1000

Interface

User's Guide

SQUARE D COMPANY

CRISP AUTOMATION SYSTEMS

VMS/HP 1000 Interface User's Guide
Document Number: 500 051-001, Rev. 2

Document History

Revision	Date	Pages affected/Description of change
1	2/8/91	Initial Release. ECN # 3843
2	7/8/91	Pages 3, 4, and 5. Update installation information. ECN # 3934

Software Version VMS/HP 1000 Interface Rev. 1.0

Table of Contents

Introduction

General....1

Operation

General....3
Requirements, HP 1000....3
Requirements, VAX....3
Installation, General....3
Installation, VAX....4
Installation, HP 1000....4
Privileges....5

HP 1000 and VMS Differences

Differences Between HP 1000 and VMS Implementations....7
Well Known Addresses....8
    Available Addresses....9
Exchanging Fortran Programs....9

Compiling and Linking

Compiling - Fortran....11
Compiling - C....11
Linking....11
Porting From the HP 9000....11
Operation With the HP 9000....12

Errors

Errors by Number....13
Errors by Symbol Name....15
Errors by VMS Status Value....17

Function Descriptions

ADDOPT....19
HP_VAX_INT2....21
HP_VAX_INT4....23
HP_VAX_REAL....25
INITOPT....27
IPCCONNECT....29
IPCCONTROL....33
IPCCREATE....37
IPCDEST....39
IPCRECV....43
IPCRECVCN....47
IPCSELECT....51
IPCSEND....55
IPCSHUTDOWN....59
IPC_DBG_DUMP....61
IPC_DBG_OUTPUT....63
IPC_TEST_VAX....65
READOPT....67
VAX_HP_INT2....69
VAX_HP_INT4....71
VAX_HP_REAL....73

General

The VMS/HP 1000 Interface provides a method for a program running on a VAX to exchange data with a program running on an HP 1000. The interface implements the facility NetIPC, which is available on the HP 1000.

+ **Note:**

This document is intended to be used in conjunction with the NetIPC manuals for the HP 1000.

It is important for the system manager to be familiar with the VMS/ULTRIX Connection System Manager's Guide.

To learn about how the VMS NetIPC implementation works, refer to the VMS/ULTRIX Connection Programming Manual.

This manual is broken down into the following sections.

Section	Description
Operation *(page 3)*	This section defines the basic operational details required for VMS/HP 1000 communication, including the steps necessary to communicate with HP 1000s and the privileges necessary for the VMS/HP 1000 interface to execute properly.
HP 1000 and VMS Differences *(page 7)*	This section defines the differences between HP 1000 and VMS implementation of NetIPC.
Compiling and Linking *(page 11)*	This section defines the requirements when compiling and linking programs for use with the VMS/HP 1000 interface.
Errors *(page 15)*	This section lists errors by NetIPC number, by symbol name, and by VMS status value.
Function Descriptions *(page 19)*	This section defines the VMS/HP 1000 interface functions. Each function contains a description, format, and operation section.

Notes:

General

NetIPC enables two processes to exchange data. The processes can run on two separate computers or can be running on the same machine. One of the processes, known as the 'server', creates a 'socket'. The other process, known as the 'client', connects to the socket of the server. This connection creates a bi-directional communications path. The processes then use the path until it is closed by one of the processes. The meaning of the data exchanged is completely application-dependent.

Requirements, HP 1000

- HP 1000 A-Series computer
- A minimum of 1.5 Mbytes of main memory
- RTE-A PCO 5.1 or later software
- VC+ System Enhancements Package software, HP Product Number 92078A/R/E
- NS-ARPA/1000 software, HP Product Number 91790A/R
- LAN/1000 Link hardware, HP Product Number 12076A

Requirements, VAX

- VMS 5.3 or later operating system
- VMS/ULTRIX Connection, Version 1.3 or later, DEC part number QL-VHRA9-J
- Network interface appropriate to VAX CPU selected

Installation, General

An Internet Protocol (IP) network number should be requested from the DDN Network Information Center (NIC). The NIC can be reached by way of the following.

DDN Network Information Center
SRI International
Room EJ210
333 Ravenswood Avenue
Menlo Park, CA 94025

(800) 235-3155

`HOSTMASTER@NIC.DDN.MIL` (via electronic mail)

Not having a unique network number may cause network failure if the HP and VAX systems are connected to a wide-area network.

Each system connected to the network must have a unique address.

Installation, VAX

The source files for the VMS implementation of NetIPC are installed in directory `DISK$USER:[CRISP.NETIPC]`.

When new systems are added to the network, they must be added to the UCX database of each VMS system. This is accomplished by using the **SET HOST** command in the UCX program. For more information, refer to the VMS/ULTRIX Connection System Manager's Guide.

Installation, HP 1000

File `/system/nsout01` of each HP 1000 should have a entry in it as follows.

```
*++ DCN ++*
* Enter information on this node for each Directly Connected Network.
* Format:
*
* <local IP addr>, <[subnet mask],> RTR, <segsize: [1200..(8000)]>
* <local IP addr>, <[subnet mask],> 802/LAN/ETHERNET,
* <segsize:[1200..(1514)]>, <link LU>, (E)/NE, <[station addr]>
*
* Where E = Enable, NE = do Not Enable.  Type /E to end.
*
*DCN: _
192.86.4.127,ETHERNET,,60,E
*DCN: _
```

Where the `192.86.4.127` is replaced by the address of the particular HP 1000. If `802` or `LAN` is specified instead of `ETHERNET`, communication with the VAX systems is impossible.

When new systems are added to the network, the names of each system should be added to file `/system/hosts` of each HP 1000. The contents of the file are as follows.

```
BEGIN SPA61 192.86.4.126 IP END
BEGIN SPA60 192.86.4.125 IP END
BEGIN XYZ 192.86.4.4 IP END
```

Where `SPA61` is the name of the system and `192.86.4.126` is the address of the system.

When the HP 1000 is re-booted, the following commands must be executed, either manually or automatically, as a part of the NS-ARPA/1000 initialization.

```
CI> WD /ARPA1000/RUN
CI> NETINIT /SYSTEM/NETOUT01
CI> NRINIT /SYSTEM/HOSTS
```

Privileges

All processes using NetIPC must have a privilege of NETMBX. Without that privilege, the return code from NetIPC calls will be NSR_NETWORK_DOWN (a value of 4).

A server creating a socket with a well-known address must have one of the following privileges: OPER, SYSPRV, or BYPASS. Privilege OPER is recommended. Privilege BYPASS is **not** recommended.

Notes:

Differences Between HP 1000 and VMS Implementations

The VMS version of NetIPC replicates the behavior of the HP 1000 implementation of NetIPC. However, the following differences are unavoidable.

- The connection-by-name service of NetIPC cannot be used. Well-known addresses are required. The following functions are not implemented:

  - ipclookup
  - ipcname
  - ipcnamerase.

- The capability of sharing sockets by NetIPC with a subprocess is not supported. The following functions are not implemented:

  - ipcget
  - ipcgive.

- Because the HP 1000 uses 16-bit addresses and the VAX uses 32-bit addresses, the `adrof` function is not provided. Fortran users can use the `%loc()` function instead.

- Each routine returns a VMS status value. These values can be ignored by the caller. For Fortran programs, the routines can be called as either a FUNCTION or as a SUBROUTINE. The returned status values are cross-indexed to NetIPC return codes.

- ADDOPT checks the data for the option being added to ensure that the value is valid for that option. HP 1000 NetIPC checks the value when the option array is actually used. This presents no problem, since an incorrect data value causes the user's program to fail in either case.

- Routine IPC_DBG_DUMP has been added. The VMS user of NetIPC calls IPC_DBG_DUMP to list all the internal status information from NetIPC. Some overall information is listed and then the information for each descriptor in use is listed. Use of this routine is optional.

- Routine IPC_DBG_OUTPUT has been added. The VMS user of NetIPC can call IPC_DBG_OUTPUT to enable special debugging output. When enabled, the VMS LIB$SIGNAL mechanism will be used to indicate important changes in NetIPC status. Use of this routine is optional. Debugging output is disabled by default.

(Continued on next page.)

Differences Between HP 1000 and VMS Implementations (cont)

- Routines have been added to convert between VAX and HP 1000 numerical formats.

HP_VAX_INT2	VAX_HP_INT2
HP_VAX_INT4	VAX_HP_INT4
HP_VAX_REAL	VAX_HP_REAL

- Checksumming is always enabled on connections with the VMS version of NetIPC. Therefore, flag bit 21 is ignored in routines IPCCONNECT and IPCRECVCN.

- IPCDEST in the VMS implementation determines if the host name is in the UCX database. If not, no communications are possible.

- IPCRECV and IPCSEND allow vectored read and write. The maximum number of vectors is 8. The structure of the vector is different, since addresses on the VAX are 32 bits, not 16 bits.

- IPCSELECT has the following differences and limitations.

  - If an infinite timeout period is specified, the status of the sockets is checked every 1/2 second, rather than continuously.
  - It is not possible to determine which sockets can support writing without blocking. The output bits in 'writemap' are always clear.
  - It is not possible to determine which sockets have queued connection requests. The output bits in 'exceptionmap' are never set by this condition.
  - The routine marks as exceptional sockets whose connection has failed if the socket is to be checked for reading. However, no check is made of other sockets to determine their connection status.

- Routine IPC_TEST_VAX has been added. This function returns a value of 1 when called on a VMS machine.

Well-Known Addresses

Because the by-name connection service of the HP 1000 is not possible in the VMS implementation, the processes to be communicating must use 'well-known' port numbers. The server process creates a socket with a well-known address by using the NSO_PROTOCOL_ADDRESS option (value 128) of IPCCREATE. The client process locates the socket of the server by specifying that number as the *protoaddr* argument to IPCDEST.

Well-Known Addresses (cont)

Available Addresses

Well-known addresses range from 1 to 1023. Addresses 1 through 255 are reserved and should not be used. The following addresses are already allocated for system services: 20, 21, 23, 25, 111, and 512 through 515.

Exchanging Fortran Programs

If programs are written completely in Fortran 77, they can be run on either the VAX or HP 1000. Important differences are as follows.

- The HP 1000 compiler expects a file type of `.FTN`. The VAX expects a file extension of `.FOR`.
- The HP 1000 does not have the functions that are added in the VMS implementation (e.g., IPC_TEST_VAX). File `DISK$USER:[CRISP.NETIPC]HP_STUBS.FTN` is provided as a 'library' of HP 1000 functions. These functions are inactive, but will prevent unresolved externals at link time. This file must be compiled on the HP 1000 and linked with the user's program.

Notes:

Compiling - Fortran

The VMS/HP 1000 interface has a header file, IPC_DEF_USER_FOR, that can be included in the user's source code. This file defines the symbols for return codes, VMS status values, and other items.

Before compiling the program, define the following logical name so the compiler can locate the appropriate library.

```
$ DEFINE/JOB FORT$LIBRARY CRISP$LIB:CRISPUSERLIB.TLB
```

Compiling - C

The VMS NetIPC implementation has a header file, IPC_DEF_USER_C, that should be included in the user's source code. This file defines the symbols for return codes, VMS status values, and other items.

Before compiling the program, define the following logical name so the compiler can locate the appropriate library.

```
$ DEFINE/JOB C$LIBRARY CRISP$LIB:CRISPUSERLIB.TLB
```

Linking

After the source files are compiled, the executable image can be created by performing the following.

```
$ LINK program.obj, CRISP$LIB:IPCLIB/LIBRARY,-
                    CRISP$LIB:CRISPIMAGELIB/LIBRARY
```

If function IPC_DBG_DUMP is called, the following must be added to the previous command: ", SYS$LIBRARY:VAXCRTL.EXE/SHARE".

Porting From the HP 9000

Programs written for the HP 9000 can be easily ported to the VMS implementation of NetIPC. Before including IPC_DEF_USER_C, define symbol HP_UX_COMPATIBILITY. This enables the program to use the argument passing mechanisms as used on the HP 9000.

References to include file <sys/ns-ipc.h> should be replaced by IPC_DEF_USER_C. Use of "#ifdef VMS" conditional compilation is recommended.

Many options and flags available on the HP 9000 are not available on the HP 1000. These cause errors when the program is run. However, if the HP 9000 program is restricted to the features on the HP 1000, it will work properly.

The only important incompatibility between the HP 1000 and the HP 9000 is in the use of bit masks for the IPCSELECT call. The following example illustrates the different usage (vc_dx is the descriptor for a connection socket).

(Continued on next page.)

Porting From the HP 9000 (cont)

HP 1000

```
long read;
long write;
long except;

read = 0;
write = 0;
except = 0;

read |= 1 << (32-vc_dx);
except |= 1 << (32-vc_dx);

ipcselect (&sdbound, &read, &write, &except, -1, &result);
```

HP 9000

```
long read[2];
long write[2];
long except[2];

read[0] = 0;
read[1] = 0;
write[0] = 0;
write[1] = 0;
except[0] = 0;
except[1] = 0;

read[vc_dx/32] |= ((unsigned int) 0x80000000 >> (vc_dx % 32));
except[vc_dx/32] |= ((unsigned int) 0x80000000 >> (vc_dx % 32));

ipcselect (&sdbound, &read[0], &write[0], &except[0], -1, &result);
```

Operation With the HP 9000

Before an HP 9000 process can connect to a VAX process, the VAX system must be added to the proxy database of the HP 9000. The following must be performed by the 'superuser' after reboot.

```
# proxy on
# proxy add hostname HPDSN xx.xx.xx.xx ieee
```

Where `xx.xx.xx.xx` is the Internet address of the host, identified by `hostname`.

Value	Symbolic Name	VMS Status
0	NSR_NO_ERROR	NETIPC_CONNECTED SS$_NORMAL
4	NSR_NETWORK_DOWN	SS$_DEVINACT SS$_DEVNOTMOUNT SS$_IVDEVNAM SS$_NONETMBX SS$_NOPRIV
5	NSR_SOCK_KIND	NETIPC_SOCKIND
6	NSR_PROTOCOL	NETIPC_PROTOCOL
7	NSR_FLAGS	NETIPC_FLAGS
8	NSR_OPT_OPTION	NETIPC_UNKOPT
10	-	(not applicable)
11	NSR_NO_MEMORY	SS$_INSFMEM
14	NSR_ADDR_OPT	SS$_NOPRIV (ipccreate)
15	NSR_NO_FILE_AVAIL	NETIPC_NOPORT
16	-	(not applicable)
19	NSR_MAX_MSG_SIZE_OPT	NETIPC_SIZEBAD
20	NSR_OFFSET_OPT	NETIPC_BADOFF
21	NSR_DUP_OPTION	NETIPC_DUPOPT
24	NSR_MAX_CONNECTQ	NETIPC_CONNLIM
28	NSR_NLEN	NETIPC_NODENAME
29	NSR_DESC	NETIPC_BADDX
30	-	(not applicable)
31	-	(not applicable)
34	-	(not applicable)
35	-	(not applicable)
36	-	(not applicable)
37	-	(not applicable)
38	-	(not applicable)
39	-	(not applicable)
40	NSR_NO_NODE	SS$_ENDOFFILE
44	-	(not applicable)
46	-	(not applicable)
50	NSR_DLEN	NETIPC_DATALEN NETIPC_SIZE
51	NSR_DEST	NETIPC_NOTDEST
52	-	(not applicable)
53	-	(not applicable)
54	NSR_NOT_CALL_SOCKET	NETIPC_NOTCALL
55	-	(not applicable)
56	NSR_WOULD_BLOCK	NETIPC_WUDBLOCK
59	NSR_SOCKET_TIMEOUT	NETIPC_TIMEOUT
62	NSR_CNCT_PENDING	NETIPC_IPCRECV
64	NSR_REMOTE_ABORT	SS$_CONNECFAIL SS$_REJECT
65	-	(not applicable)
66	NSR_NOT_CONNECTION	NETIPC_NOTCONNECT NETIPC_SELECT
68	-	(not applicable)
70	-	(not applicable)

Value	Symbolic Name	VMS Status
74	NSR_REQUEST	NETIPC_REQUEST
76	NSR_TIMEOUT_VALUE	NETIPC_TIMEOUT
98	NSR_VECT_DST_INDEX	NETIPC_BADVECT
99	NSR_VECT_COUNT	NETIPC_VECTS
106	NSR_DUP_ADDRESS	SS$_DUPLNAM
107	NSR_NETWORK_SHUTTING	SS$_SHUT
109	NSR_REMOTE_RELEASED	SS$_LINKDISCON
111	NSR_INTERNAL	**Any condition not otherwise listed**
116	-	(not applicable)
122	-	(not applicable)
123	-	(not applicable)
124	NSR_OPT_ENTRY_NUM	NETIPC_BOUNDS
125	NSR_OPT_DATA_LEN	NETIPC_OPTDATALEN
126	NSR_OPT_TOTAL	NETIPC_TOOMANY
127	NSR_OPT_CANTREAD	NETIPC_NOREAD
128	NSR_READ_THRESH	NETIPC_RTHRESH NETIPC_RTHRESHBIG
129	NSR_WRITE_THRESH	NETIPC_WTHRESH
130	NSR_WRITE_THRESH_BIG	NETIPC_WTHRESHBIG
131	NSR_RESOURCE_ERROR	SS$_EXQUOTA SS$_NOIOCHAN
132	-	(not applicable)
133	-	(not applicable)
134	-	(not applicable)
135	-	(not applicable)
136	NSR_BAD_SD_BOUND	NETIPC_BADSD
1001	-	(not applicable)

Value	Symbolic Name	VMS Status
14	NSR_ADDR_OPT	SS$_NOPRIV (ipccreate)
136	NSR_BAD_SD_BOUND	NETIPC_BADSD
62	NSR_CNCT_PENDING	NETIPC_IPCRECV
29	NSR_DESC	NETIPC_BADDX
51	NSR_DEST	NETIPC_NOTDEST
50	NSR_DLEN	NETIPC_DATALEN NETIPC_SIZE
106	NSR_DUP_ADDRESS	SS$_DUPLNAM
21	NSR_DUP_OPTION	NETIPC_DUPOPT
7	NSR_FLAGS	NETIPC_FLAGS
111	NSR_INTERNAL	**Any condition not otherwise listed**
24	NSR_MAX_CONNECTQ	NETIPC_CONNLIM
19	NSR_MAX_MSG_SIZE_OPT	NETIPC_SIZEBAD
4	NSR_NETWORK_DOWN	SS$_DEVINACT SS$_DEVNOTMOUNT SS$_NOPRIV SS$_NONETMBX SS$_IVDEVNAM
107	NSR_NETWORK_SHUTTING	SS$_SHUT
28	NSR_NLEN	NETIPC_NODENAME
54	NSR_NOT_CALL_SOCKET	NETIPC_NOTCALL
66	NSR_NOT_CONNECTION	NETIPC_NOTCONNECT NETIPC_SELECT
0	NSR_NO_ERROR	NETIPC_CONNECTED SS$_NORMAL
15	NSR_NO_FILE_AVAIL	NETIPC_NOPORT
11	NSR_NO_MEMORY	SS$_INSFMEM
40	NSR_NO_NODE	SS$_ENDOFFILE
20	NSR_OFFSET_OPT	NETIPC_BADOFF
127	NSR_OPT_CANTREAD	NETIPC_NOREAD
125	NSR_OPT_DATA_LEN	NETIPC_OPTDATALEN
124	NSR_OPT_ENTRY_NUM	NETIPC_BOUNDS
8	NSR_OPT_OPTION	NETIPC_UNKOPT
126	NSR_OPT_TOTAL	NETIPC_TOOMANY
6	NSR_PROTOCOL	NETIPC_PROTOCOL
128	NSR_READ_THRESH	NETIPC_RTHRESH NETIPC_RTHRESHBIG
64	NSR_REMOTE_ABORT	SS$_CONNECFAIL SS$_REJECT
109	NSR_REMOTE_RELEASED	SS$_LINKDISCON
74	NSR_REQUEST	NETIPC_REQUEST
131	NSR_RESOURCE_ERROR	SS$_EXQUOTA SS$_NOIOCHAN
59	NSR_SOCKET_TIMEOUT	NETIPC_TIMEOUT
5	NSR_SOCK_KIND	NETIPC_SOCKIND
76	NSR_TIMEOUT_VALUE	NETIPC_TIMEOUT
99	NSR_VECT_COUNT	NETIPC_VECTS
98	NSR_VECT_DST_INDEX	NETIPC_BADVECT
56	NSR_WOULD_BLOCK	NETIPC_WUDBLOCK
129	NSR_WRITE_THRESH	NETIPC_WTHRESH
130	NSR_WRITE_THRESH_BIG	NETIPC_WTHRESHBIG

Notes:

Value	Symbolic Name	VMS Status
111	NSR_INTERNAL	**Any condition not otherwise listed**
29	NSR_DESC	NETIPC_BADDX
20	NSR_OFFSET_OPT	NETIPC_BADOFF
136	NSR_BAD_SD_BOUND	NETIPC_BADSD
98	NSR_VECT_DST_INDEX	NETIPC_BADVECT
124	NSR_OPT_ENTRY_NUM	NETIPC_BOUNDS
0	NSR_NO_ERROR	NETIPC_CONNECTED
24	NSR_MAX_CONNECTQ	NETIPC_CONNLIM
50	NSR_DLEN	NETIPC_DATALEN
21	NSR_DUP_OPTION	NETIPC_DUPOPT
7	NSR_FLAGS	NETIPC_FLAGS
62	NSR_CNCT_PENDING	NETIPC_IPCRECV
28	NSR_NLEN	NETIPC_NODENAME
15	NSR_NO_FILE_AVAIL	NETIPC_NOPORT
127	NSR_OPT_CANTREAD	NETIPC_NOREAD
54	NSR_NOT_CALL_SOCKET	NETIPC_NOTCALL
66	NSR_NOT_CONNECTION	NETIPC_NOTCONNECT
51	NSR_DEST	NETIPC_NOTDEST
125	NSR_OPT_DATA_LEN	NETIPC_OPTDATALEN
6	NSR_PROTOCOL	NETIPC_PROTOCOL
74	NSR_REQUEST	NETIPC_REQUEST
128	NSR_READ_THRESH	NETIPC_RTHRESH
128	NSR_READ_THRESH	NETIPC_RTHRESHBIG
66	NSR_NOT_CONNECTION	NETIPC_SELECT
50	NSR_DLEN	NETIPC_SIZE
19	NSR_MAX_MSG_SIZE_OPT	NETIPC_SIZEBAD
5	NSR_SOCK_KIND	NETIPC_SOCKIND
59	NSR_SOCKET_TIMEOUT	NETIPC_TIMEOUT
76	NSR_TIMEOUT_VALUE	NETIPC_TIMEOUT
126	NSR_OPT_TOTAL	NETIPC_TOOMANY
8	NSR_OPT_OPTION	NETIPC_UNKOPT
99	NSR_VECT_COUNT	NETIPC_VECTS
129	NSR_WRITE_THRESH	NETIPC_WTHRESH
130	NSR_WRITE_THRESH_BIG	NETIPC_WTHRESHBIG
56	NSR_WOULD_BLOCK	NETIPC_WUDBLOCK
64	NSR_REMOTE_ABORT	SS$_CONNECFAIL
4	NSR_NETWORK_DOWN	SS$_DEVINACT
4	NSR_NETWORK_DOWN	SS$_DEVNOTMOUNT
106	NSR_DUP_ADDRESS	SS$_DUPLNAM
40	NSR_NO_NODE	SS$_ENDOFFILE
131	NSR_RESOURCE_ERROR	SS$_EXQUOTA
11	NSR_NO_MEMORY	SS$_INSFMEM
4	NSR_NETWORK_DOWN	SS$_IVDEVNAM
109	NSR_REMOTE_RELEASED	SS$_LINKDISCON
131	NSR_RESOURCE_ERROR	SS$_NOIOCHAN
4	NSR_NETWORK_DOWN	SS$_NONETMBX
4	NSR_NETWORK_DOWN	SS$_NOPRIV
14	NSR_ADDR_OPT	SS$_NOPRIV (ipccreate)
0	NSR_NO_ERROR	SS$_NORMAL
64	NSR_REMOTE_ABORT	SS$_REJECT
107	NSR_NETWORK_SHUTTING	SS$_SHUT

Notes:

Description

Adds an argument and related data to the *opt* array.

Format

ADDOPT (*opt, argnum, optioncode, datalength, data, error*)

Arguments

opt A pointer to the option array to be modified.

Usage: Option array.
Type: Array of words.
Access: Modify.
Mechanism: By reference.

argnum The number of the argument to be added, beginning with 0.

Usage: Number of argument.
Type: Word.
Access: Read only.
Mechanism: By reference.

optioncode The number of the option to be added.

Usage: Option code value.
Type: Word.
Access: Read only.
Mechanism: By reference.

datalength The length, in bytes, of the data to be stored with the option.

Usage: Length of data.
Type: Word.
Access: Read only.
Mechanism: By reference.

data A pointer to the array of data.

Usage: Data.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

error The error code from this operation.

Usage: Error code.
Type: Word.
Access: Write only.
Mechanism: By reference.

Operation

This function adds an argument and related data to the *opt* array.

INITOPT must be called before calling ADDOPT. ADDOPT may then be called one or more times. The first call to ADDOPT should specify an *argnum* of 0, the second 1, etc.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Success:

SS$_NORMAL	Normal successful completion.

Error:

NETIPC_BADOFF	The value of the data offset was not valid
NETIPC_BOUNDS	The value of *argnum* was less than zero, or greater than or equal to the number passed to INITOPT
NETIPC_CONNLIM	The number of connections that can be queued to the socket is not valid
NETIPC_DUPOPT	The value of *optioncode* duplicates a value already in the *opt* array
NETIPC_OPTDATALEN	The value of *datalength* was not valid for the option specified
NETIPC_SIZEBAD	The value specified for the maximum send or receive size was not valid
NETIPC_UNKOPT	Value of *optioncode* not recognized.

HP_VAX_INT2

Description

Converts an INTEGER*2 number from HP 1000 to VAX format.

Format

HP_VAX_INT2 (*hp, vax*)

Arguments

hp The HP 1000 number to be converted.

Usage: HP 1000 number.
Type: Word.
Access: Read only.
Mechanism: By reference.

vax The VAX number after conversion.

Usage: VAX number.
Type: Word.
Access: Write only.
Mechanism: By reference.

Operation

Converts an HP 1000-format number, as received, into a VAX-format number.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL Normal successful completion.

Notes:

HP_VAX_INT4

Description

Converts an INTEGER*4 number from HP 1000 to VAX format.

Format

HP_VAX_INT4 (*hp, vax*)

Arguments

hp The HP 1000 number to be converted.

Usage: HP 1000 number.
Type: Longword.
Access: Read only.
Mechanism: By reference.

vax The VAX number after conversion.

Usage: VAX number.
Type: Longword.
Access: Write only.
Mechanism: By reference.

Operation

Converts an HP 1000-format number, as received, into a VAX-format number.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL Normal successful completion.

Notes:

Description

Converts an HP 1000 REAL*4 floating point number into a VAX floating point number.

Format

HP_VAX_REAL (*hp, vax*)

Arguments

hp The number to be converted.

Usage: HP 1000 number (REAL*4).
Type: Unsigned longword.
Access: Read only.
Mechanism: By reference.

vax The number after conversion.

Usage: VAX number (REAL*4).
Type: F_FLOAT.
Access: Write only.
Mechanism: By reference.

Operation

This function converts an HP 1000 floating point number into the VAX equivalent.

Note that some small loss of precision (less than 0.0005%) can be expected.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL Normal successful completion.

Notes:

Description

Initializes the *opt* array.

Format

INITOPT (*opt, optnumarg, error*)

Arguments

opt A pointer to the option array to be modified.

Usage: Option array.
Type: Array of words.
Access: Modify.
Mechanism: By reference.

optnumarg The number of arguments that will be added.

Usage: Number of arguments.
Type: Word.
Access: Read only.
Mechanism: By reference.

error The error code from this operation.

Usage: Error code.
Type: Word.
Access: Write only.
Mechanism: By reference.

Operation

The function initializes the *opt* array. ADDOPT is then called to build up the list of options for a later IPC call.

A value of 0 for *optnumarg* is valid and indicates that the *opt* array will be empty.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Returns (cont)

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL	Normal successful completion.

Error:

NETIPC_TOOMANY	The value of *optnumarg* was too large. This implementation of NetIPC allows only 5 options.

IPCCONNECT

Description

Connect to a socket of a remote process.

Format

IPCCONNECT (*calldesc, destdesc, flags, opt, vcdesc, result*)

Arguments

calldesc The call descriptor to use.

Usage: Call descriptor.
Type: Longword.
Access: Read only.
Mechanism: By reference.

destdesc The destination descriptor to use.

Usage: Destination descriptor.
Type: Longword.
Access: Read only.
Mechanism: By reference.

flags A bit mask of request flags.

Usage: Request flags.
Type: Longword.
Access: Read only.
Mechanism: By reference.

opt A pointer to the option array.

Usage: Option array.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

vcdesc The connection descriptor created through the connection.

Usage: Connection descriptor.
Type: Longword.
Access: Write only.
Mechanism: By reference.

result The error code from this operation.

Usage: Error code.
Type: Longword.
Access: Write only.
Mechanism: By reference.

Operation

This function requests a connection to a socket of another process. The connection process is begun by IPCCONNECT, but IPCRECV must be called to confirm that the connection is established.

Refer to IPCDEST to create destination descriptor *destdesc.*

The HP 1000 allows flag bit 21 (NSF_CHECKSUM) to enable checksumming. Since there is no way to disable checksumming using the VMS socket interface, checksumming is always enabled.

Options Supported	Description
3 NSO_MAX_SEND_SIZE	Sets the maximum number of bytes allowed in a single IPCSEND call on this connection. Allowable range: 1 to 8000 bytes. Default: 100 bytes.
4 NSO_MAX_RECV_SIZE	Sets the maximum number of bytes allowed in a single IPCRECV call on this connection. Allowable range: 1 to 8000 bytes. Default: 100 bytes.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Returns (cont)

Condition Values Returned

Returns failure statuses from the following system services:

- LIB$GET_EF
- SYS$ASSIGN
- SYS$QIOW (function IO$_SETMODE)
- SYS$QIO (function IO$_ACCESS).

Success:

SS$_NORMAL	Normal successful completion.

Error:

NETIPC_BADDX	Descriptor was not valid type or value
NETIPC_FLAGS	One or more flags bits are not supported
NETIPC_NOPORT	Unable to create descriptor
NETIPC_NOTCALL	Not a call descriptor
NETIPC_NOTDEST	Not a destination descriptor.

Notes:

Description

Change the operation of a socket.

Format

IPCCONTROL (*descriptor, request, wrtdata, wlen, readdata, rlen, flags, result*)

Arguments

descriptor The descriptor whose operation is to be modified.

Usage: Descriptor to be modified.
Type: Longword.
Access: Read only.
Mechanism: By reference.

request The change to be made.

Usage: Request code.
Type: Longword.
Access: Read only.
Mechanism: By reference.

wrtdata A value associated with the request.

Usage: Input data buffer.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

wlen The length of *wrtdata*. Required when *wrtdata* is required.

Usage: Length of buffer.
Type: Longword.
Access: Read only.
Mechanism: By reference.

readdata Not used in this implementation.

Usage: Unused.
Type: Array of words.
Access: Write only.
Mechanism: By reference.

rlen Not used in this implementation.

Usage: Unused.
Type: Longword.
Access: Modify.
Mechanism: By reference.

(Continued on next page.)

Arguments (cont)

flags A bit mask of request flags. No flags are defined for this function.

Usage: Request flags.
Type: Longword.
Access: Read only.
Mechanism: By reference.

result The error code from this operation.

Usage: Error code.
Type: Longword.
Access: Write only.
Mechanism: By reference.

Operation

Modifies the operation of a socket. Requests that can be made are as follows.

request	Operation
1 NSC_NBIO_ENABLE	Put socket into asynchronous mode. ‹ ›
2 NSC_NBIO_DISABLE	Put socket into synchronous mode. ‹ ›
3 NSC_TIMEOUT_RESET	Change synchronous timeout. ‹ › ´
1000 NSC_RECV_THRESH_RESET	Change read threshold. › ´
1001 NSC_SEND_THRESH_RESET	Change write threshold. › ´

‹ = Call sockets only.
› = Connection sockets only.
´ = Arguments *wrtdata* and *wlen* are required.

request	*wrtdata*
3 NSC_TIMEOUT_RESET	*wrtdata* = timeout in tenths of a second (e.g., 700 = 70 seconds)
1000 NSC_RECV_THRESH_RESET	Receive threshold, bytes.
1001 NSC_SEND_THRESH_RESET	Send threshold, bytes.

wlen must be a value of 2 when required. *wrtdata* must be a word.

(Continued on next page.)

Operation (cont)

Defaults

Timeout	60 seconds (value of 600)
Receive threshold	1 byte
Send threshold	1 byte

Returns

Usage:	Condition code.
Type:	Unsigned longword.
Access:	Write only.
Mechanism:	By value.

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL	Normal successful completion.

Error:

NETIPC_BADDX	Descriptor was not valid type or value
NETIPC_DATALEN	Length of data not valid
NETIPC_FLAGS	One or more flags bits are not supported
NETIPC_NOTCONNECT	Attempt to set threshold on other than a connection socket
NETIPC_REQUEST	Request not valid
NETIPC_RTHRESH	Receive threshold ≤ 0
NETIPC_RTHRESHBIG	Receive threshold > 255
NETIPC_TIMEOUT	Timeout value not valid
NETIPC_WTHRESH	Write threshold ≤ 0
NETIPC_WTHRESHBIG	Write threshold > 255.

Notes:

Description

Create a call socket.

Format

IPCCREATE (*socketkind, protocol, flags, opt, calldesc, result*)

Arguments

socketkind Must be 3 (NS_CALL), to indicate a call socket.

Usage: Kind of socket.
Type: Longword.
Access: Read only.
Mechanism: By reference.

protocol Must be 4 (NSP_TCP) or 0 (use default, which is NSP_TCP).

Usage: Protocol to use.
Type: Longword.
Access: Read only.
Mechanism: By reference.

flags A bit mask of request flags. No flags are defined for this function.

Usage: Request flags.
Type: Longword.
Access: Read only.
Mechanism: By reference.

opt A pointer to the option array.

Usage: Option array.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

calldesc The call descriptor that is created.

Usage: Call descriptor.
Type: Longword.
Access: Write only.
Mechanism: By reference.

result The error code from this operation.

Usage: Error code.
Type: Longword.
Access: Write only.
Mechanism: By reference.

Operation

Creates a call socket.

Allowable options in *opt* are as follows.

6 NSO_MAX_CONN_REQ_BACK	Maximum number of connection requests that may be queued to the socket. Default: 3.
128 NSO_PROTOCOL_ADDRESS	The TCP protocol address to be used by the call socket. If not specified, or if zero, the system allocates an address.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Returns failure statuses from the following system services:

- LIB$GET_EF
- SYS$ASSIGN
- SYS$QIOW (function IO$_SETMODE).

Success:

SS$_NORMAL	Normal successful completion.

Error:

NETIPC_FLAGS	One or more flags bits are not supported
NETIPC_NOPORT	Unable to create descriptor
NETIPC_PROTOCOL	The value of *protocol* was not valid
NETIPC_SOCKIND	The value of *socketkind* was not valid.

Description

Creates a destination (or path report) descriptor.

Format

IPCDEST (*socketkind, hostname, hostlen, protocol, protoaddr, protolen, flags, opt, destdesc, result*)

Arguments

socketkind Must be 3 (NS_CALL), to indicate a call socket.

Usage: Kind of socket.
Type: Longword.
Access: Read only.
Mechanism: By reference.

hostname The address of the . In Fortran, this should be the address of an array of INTEGER*2 equivalenced to the desired string.

Usage: Name of destination node.
Type: Character string.
Access: Read only.
Mechanism: By reference.

hostlen The length of string *hostname.*

Usage: Length of node name.
Type: Longword.
Access: Read only.
Mechanism: By reference.

protocol Must be 4 (NSP_TCP) or 0 (use default, which is NSP_TCP).

Usage: Protocol to use.
Type: Longword.
Access: Read only.
Mechanism: By reference.

protoaddr The first word in the array is the TCP protocol address (port number) to be used on the destination node.

Usage: Protocol address.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

(Continued on next page.)

Arguments (cont)

protolen The length of *protoaddr*. Must be 2.

Usage: Length of protocol address, in bytes.
Type: Longword.
Access: Read only.
Mechanism: By reference.

flags A bit mask of request flags. No flags are defined for this function.

Usage: Request flags.
Type: Longword.
Access: Read only.
Mechanism: By reference.

opt A pointer to the option array. Not used in this implementation.

Usage: Option array.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

destdesc The destination descriptor that is created.

Usage: Destination descriptor.
Type: Longword.
Access: Read only.
Mechanism: By reference.

result The error code from this operation.

Usage: Error code.
Type: Longword.
Access: Read only.
Mechanism: By reference.

Operation

This function creates a destination descriptor. The descriptor is then passed to IPCCONNECT to connect to the desired socket.

Unlike the HP 1000, this function determines if *hostname* is defined in the UCX database. If the name is not defined in the database, no communications is possible. Refer to the VMS/ULTRIX Connection System Manager's Guide for details.

Returns

Usage: Condition Code
Type: Unsigned Longword
Access: Write only
Mechanism: By value

Condition Values Returned

Returns failure statuses from the following system services:

- LIB$GET_EF
- SYS$ASSIGN
- SYS$QIOW (functions IO$_SETMODE and IO$_ACPCONTROL).

Success:

SS$_NORMAL	Normal successful completion.

Error:

SS$_ENDOFFILE	Host name not located in UCX database
NETIPC_DATALEN	The value of *protolen* is not valid, or the pointer to *protoaddr* is NULL
NETIPC_FLAGS	One or more flags bits are not supported
NETIPC_HOSTNAME	The value of *hostlen* is ≤ 0 or greater than NS_MAX_NODE_NAME
NETIPC_NOPORT	Unable to create descriptor
NETIPC_PROTOCOL	The value of *protocol* was not valid
NETIPC_SOCKIND	The value of *socketkind* was not valid.

Notes:

Description

Completes a connection started by IPCCONNECT. Receives data on an active connection.

Format

IPCRECV (*vcdesc, data, dlen, flags, opt, result*)

Arguments

vcdesc The connection descriptor from IPCCONNECT or IPCRECVCN.

Usage: Connection descriptor.
Type: Longword.
Access: Read only.
Mechanism: By reference.

data The address of the data buffer or data vector.

Usage: Data buffer or data vector.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

dlen The length of the data buffer, in bytes. For data vectors, *dlen* is the size, in bytes, of *data.* Upon return from the function, contains the number of bytes read.

Usage: Length of *data.*
Type: Longword.
Access: Modify.
Mechanism: By reference.

flags A bit mask of request flags.

Usage: Request flags.
Type: Longword.
Access: Read only.
Mechanism: By reference.

opt A pointer to the option array.

Usage: Option array.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

(Continued on next page.)

Arguments (cont)

result The error code from this operation.

Usage: Error code.
Type: Longword.
Access: Write only.
Mechanism: By reference.

Operation

This function has two purposes: to check the status of a connect request initiated by IPCCONNECT, and to receive data on a connection.

If the connection has not yet been established the following occurs:

- Asynchronous socket: Status NETIPC_WUDBLOCK is returned.
- Synchronous socket: This function waits up to the timeout time for the connection to be completed. If the connection does not complete, SS$_TIMEOUT is returned.

If the connection has been established and either this is the first call since calling IPCCONNECT or the value of *dlen* is zero, the function returns NETIPC_CONNECTED.

On further calls where *dlen* is not zero, the function reads data. In reading data, the following flags can be specified.

20 NSF_DATA_WAIT	The function waits until the data buffer(s) are completely filled if the socket is synchronous. If the socket is asynchronous, gets the available data and returns NETIPC_WUDBLOCK.
30 NSF_PREVIEW	'Peeks' into the data buffer. Must not be specified along with NSF_DATA_WAIT.
31 NSF_VECTORED	*data* and *dlen* are used as an array of data vectors. In this case, *dlen* is the size of the array of vectors (8 bytes per vector). The first 4 bytes of *data* is the address of the buffer, and the second 4 bytes is the size of the buffer. There can be a maximum of MAX_VECTORS (8) vectors.

When flag bit NSF_VECTORED is not set, the following option is valid:

8 NSO_DATA_OFFSET	Holds a 2-byte offset value. This value is added to pointer *data* before the operation is performed.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Returns failure statuses from the following system services:

- SYS$QIOW (functions IO$_ACCESS and IO$_READVBLK).

Success:

SS$_NORMAL	Normal successful completion
NETIPC_CONNECTED	Connection is established.

Informational:

NETIPC_WUDBLOCK	Socket is in asynchronous mode. Operation would have blocked.

Error:

NETIPC_BADDX	Connect descriptor is not valid
NETIPC_BADVECT	Bad length field in vector
NETIPC_FLAGS	One or more flags bits are not supported
NETIPC_NOTCONNECT	Not a connection socket
NETIPC_SIZE	Total size of buffer(s) greater than the maximum read size set by IPCCONNECT or IPCRECVCN
NETIPC_VECTS	More than MAX_VECTORS (8) were specified by *dlen.*

Fatal:

SS$_TIMEOUT	Timeout occurred while receiving data.

Notes:

Description

Receives a connection.

Format

IPCRECVCN (*calldesc, vcdesc, flags, opt, result*)

Arguments

calldesc The call descriptor to use.

Usage: Call descriptor.
Type: Longword.
Access: Read only.
Mechanism: By reference.

vcdesc The connection descriptor created through the connection.

Usage: Connection descriptor.
Type: Longword.
Access: Write only.
Mechanism: By reference.

flags A bit mask of request flags.

Usage: Request flags.
Type: Longword.
Access: Read only.
Mechanism: By reference.

opt A pointer to the option array.

Usage: Option array.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

result The error code from this operation.

Usage: Error code.
Type: Longword.
Access: Write only.
Mechanism: By reference.

Operation

Receives a connection on a call socket.

The HP 1000 allows flag 21 (NSF_CHECKSUM) to enable checksumming. Since there is no way to disable checksumming using the socket interface, checksumming is always enabled.

Options Supported	Description
3 NSO_MAX_SEND_SIZE	Sets the maximum number of bytes allowed in a single IPCSEND call on this connection. Allowable range: 1 to 8000 bytes. Default: 100 bytes.
4 NSO_MAX_RECV_SIZE	Sets the maximum number of bytes allowed in a single IPCRECV call on this connection. Allowable range: 1 to 8000 bytes. Default: 100 bytes.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Returns failure statuses from the following system services:

- LIB$GET_EF
- SYS$ASSIGN
- SYS$QIOW (function IO$_SETMODE)
- SYS$QIO (function IO$_ACCESS | IO$_ACCEPT).

Success:

SS$_NORMAL	Normal successful completion.

Informational:

NETIPC_WUDBLOCK	Socket is in an asynchronous mode. Operation would have blocked.

(Continued on next page.)

Condition Values Returned (cont)

Error:

NETIPC_BADDX	Descriptor was not valid type or value
NETIPC_FLAGS	One or more flags bits are not supported
NETIPC_NOPORT	Unable to create descriptor
NETIPC_NOTCALL	Not a call socket.

Fatal:

SS$_TIMEOUT	Timeout occurred while waiting for connection request.

Notes:

Description

Find statuses of sockets.

Format

IPCSELECT (*sdbound, readmap, writemap, exceptionmap, timeout, result*)

Arguments

sdbound

The maximum value of the descriptor. On input, *sdbound* can be set to one greater than the highest descriptor number of interest to reduce overhead. On output, *sdbound* is set to one greater than the highest descriptor number of all the descriptors that meet the selection criteria. If no criteria are met, *sdbound* is set to zero. On error, *sdbound* is set to -1.

Usage: Maximum descriptor number.
Type: Longword.
Access: Modify.
Mechanism: By reference.

readmap

Bit map for connection descriptors. A bit will be set upon output if it was set on input, and the number of unread bytes is at least as large as the corresponding read threshold.

Usage: Read bit map.
Type: Longword.
Access: Modify.
Mechanism: By reference.

writemap

Bit map for connection descriptors. A bit will be set upon output if it was set on input, and the socket can accommodate a write of at least the corresponding write threshold without blocking.

Usage: Write bit map.
Type: Longword.
Access: Modify.
Mechanism: By reference.

(Continued on next page.)

Arguments (cont)

exceptionmap Bit map for call and connection descriptors. A bit will be set upon output if it was set on input, and the corresponding socket is exceptional. A connection socket is exceptional if the connection has a problem associated with it. A call socket is exceptional if there is at least one connection queued. Unused descriptors and destination descriptors are always exceptional.

Usage: Exception bit map.
Type: Longword.
Access: Modify.
Mechanism: By reference.

timeout How long the call will wait until at least one socket meets the criteria.

Usage: Timeout value.
Type: Longword.
Access: Write only.
Mechanism: By reference.

result The error code from this operation.

Usage: Error code.
Type: Longword.
Access: Write only.
Mechanism: By reference.

Operation

Allows a process to monitor multiple sockets simultaneously.

If *timeout* > 0 and no sockets meet the selection criteria, the function will wait that time, check the sockets one more time, and return SS$_TIMEOUT.

If *timeout* = -1 and no sockets meet the selection criteria, the function will wait 1/2 second between checks, returning only if at least one socket meets the criteria.

Operation (cont)

Limitations

It is not possible to determine which sockets can immediately support writing. Therefore, the bits in *writemap* are always clear when the function returns.

It is not possible to determine the number of connections queued to a socket; therefore, sockets with pending connections are not marked as exceptional.

In general, it is not possible to determine the status of each connection. Therefore, most connection descriptors never appear to be exceptional.

Map Structure

Each bit map is arranged so the most significant bit corresponds to descriptor 1. The least significant bit corresponds to descriptor 32.

Returns

Usage:	Condition code.
Type:	Unsigned longword.
Access:	Write only.
Mechanism:	By value.

Condition Values Returned

Returns failure statuses from the following system services:

- SYS$SETIMR
- SYS$WAITFR.

Success:

SS$_NORMAL	Normal successful completion.

Error:

NETIPC_BADSD	The value of *sdbound* is ≤ 0, or > 32
NETIPC_SELECT	Attempt to read or write select on a socket that is not a connection socket
NETIPC_TIMEOUT	Value of *timeout* was not valid.

(Continued on next page.)

Condition Values Returned (cont)

Fatal:

SS$_TIMEOUT	Timeout while waiting for selection criteria to become true.

IPCSEND

Description

Send data on a connection.

Format

IPCSEND (*vcdesc, data, dlen, flags, opt, results*)

Arguments

vcdesc The connection descriptor from IPCCONNECT or IPCRECVCN.

Usage: Connection descriptor.
Type: Longword.
Access: Read only.
Mechanism: By reference.

data The address of the data buffer or data vector.

Usage: Data buffer or data vector.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

dlen The length of the data buffer, in bytes. For data vectors, the size, in bytes, of *data.*

Usage: Length of *data.*
Type: Longword.
Access: Read only.
Mechanism: By reference.

flags A bit mask of request flags.

Usage: Request flags.
Type: Longword.
Access: Read only.
Mechanism: By reference.

opt A pointer to the option array.

Usage: Option array.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

results The error code from this operation.

Usage: Error code.
Type: Longword.
Access: Write only.
Mechanism: By reference.

Operation

Sends data across a connection. The connection must be confirmed first using IPCRECV, if established by IPCCONNECT.

The following flags can be specified.

26 NSF_MORE_DATA	If clear, the data is sent immediately. If set, TCP may accumulate data from several calls to IPCSEND before the data is sent.
31 NSF_VECTORED	*data* and *dlen* are used as an array of data vectors. In this case, *dlen* is the size of the array of vectors (8 bytes per vector). The first 4 bytes of *data* is the address of the buffer, and the second 4 bytes is the size of the buffer. There can be a maximum of MAX_VECTORS (8) vectors.

When flag bit NSF_VECTORED is not set, the following option is valid.

8 NSO_DATA_OFFSET	Contains a 2-byte offset value. This value is added to pointer *data* before the operation is performed.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Returns failure statuses from the following system services:

- SYS$QIOW (functions IO$_ACCESS, IO$_SETMODE, and IO$_WRITEVBLK).

Success:

SS$_NORMAL	Normal successful completion.

Informational:

NETIPC_WUDBLOCK	Socket is in asynchronous mode. Operation would have blocked.

(Continued on next page.)

Condition Values Returned (cont)

Error:

NETIPC_BADDX	Connect descriptor is not valid
NETIPC_BADVECT	Bad length field in vector
NETIPC_FLAGS	One or more flags bits are not supported
NETIPC_IPCRECV	Function IPCRECV must be called first to verify that the connection has been established
NETIPC_NOTCONNECT	Not a connection socket
NETIPC_SIZE	Total size of buffer(s) greater than the maximum read size set by IPCCONNECT or IPCRECVCN
NETIPC_VECTS	More than MAX_VECTORS (8) were specified by *dlen.*

Fatal:

SS$_TIMEOUT	Timeout occured while sending data.

Notes:

IPCSHUTDOWN

Description

Releases a descriptor and any related resources.

Format

IPCSHUTDOWN (*descriptor, flags, opt, result*)

Arguments

descriptor The descriptor to be released.

Usage: Descriptor.
Type: Longword.
Access: Read only.
Mechanism: By reference.

flags A bit mask of request flags. No flags are defined for this function.

Usage: Request flags.
Type: Longword.
Access: Read only.
Mechanism: By reference.

opt A pointer to the option array. Not used by this implementation.

Usage: Option array.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

result The error code from this operation.

Usage: Error code.
Type: Longword.
Access: Write only.
Mechanism: By reference.

Operation

This function releases a descriptor and any associated resources.

For connection sockets, the connection is aborted immediately.

For all socket types, the socket is deleted.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Returns failure statuses from the following system services:

- SYS$DASSGN
- SYS$QIOW (functions IO$_DEACCESS and IO$_DEACCESS | IO$_SHUTDOWN).

Success:

SS$_NORMAL	Normal successful completion.

Error:

NETIPC_BADDX	Descriptor is not valid
NETIPC_FLAGS	One or more flags bits are not supported.

IPC_DBG_DUMP

Description

Dumps all internal information.

Format

IPC_DBG_DUMP ()

Operation

Dumps all information about all socket descriptors and all associated information to file `NETIPC_DUMP.DMP`.

Defining logical name NETIPC_DUMP overrides the default location and name of the file.

+ **Note:**

Uses *fprintf* to perform this function. Fortran programs have to include `VAXCRTL.EXE` when linking.

] **Caution:**]

ASTs are disabled during this time.

Returns

Usage:	Condition code.
Type:	Unsigned longword.
Access:	Write only.
Mechanism:	By value.

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL	Normal successful completion.

Notes:

IPC_DBG_OUTPUT

Description

Enables or disables debugging output.

Format

IPC_DBG_OUTPUT (*state*)

Arguments

state The new state for debugging output.

Usage: New state for debugging output.
Type: Longword.
Access: Read only.
Mechanism: By reference.

Operation

This function disables (*state* = 0) or enables (*state* > 0) debugging output. If *state* < 0 and the function is called for the first time, debugging output is enabled if the NetIPC library is compiled with DEBUG; otherwise, debugging output is disabled.

Production versions of NetIPC are not compiled with DEBUG.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL Normal successful completion.

Notes:

IPC_TEST_VAX

Description

'Tests' to determine if the system is a VAX.

Format

IPC_TEST_VAX ()

Operation

This function returns 1 if the system is a VAX. Otherwise, a 0 is returned.

Programs meant to be run on either a VAX or HP 1000 can use this function to detect the system type.

Returns

Usage:	Status value.
Type:	Longword.
Access:	Write only.
Mechanism:	By value.

Notes:

Description

Returns the option code and data related to a specified argument.

Format

READOPT (*opt, argnum, optioncode, datalength, data, error*)

Arguments

opt A pointer to the option array that is to be read.

Usage: Option array.
Type: Array of words.
Access: Modify.
Mechanism: By reference.

argnum The number of the argument to be read, beginning with 0.

Usage: Number of argument.
Type: Word.
Access: Read only.
Mechanism: By reference.

optioncode The number of the option corresponding to argument number *argnum*.

Usage: Option code value.
Type: Word.
Access: Write only.
Mechanism: By reference.

datalength Input: The length, in bytes, of the caller's data buffer. Output: The count of how many bytes were put into *data*.

Usage: Length of data buffer.
Type: Word.
Access: Modify.
Mechanism: By reference.

data A pointer to the array that will receive the data associated with argument number *argnum*.

Usage: Data.
Type: Array of words.
Access: Read only.
Mechanism: By reference.

(Continued on next page.)

Arguments (cont)

error The error code from this operation.

Usage: Error code.
Type: Word.
Access: Write only.
Mechanism: By reference.

Operation

This function retrieves the argument and related data from the *opt* array. The array was previously loaded by calling INITOPT and ADDOPT.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL	Normal successful completion.

Error:

NETIPC_BOUNDS	The value of *argnum* was less than zero, or greater than or equal to the number passed to *initopt*
NETIPC_NOREAD	Unable to read option at *argnum*
NETIPC_OPTDATALEN	The value of *datalength* was not valid for the option specified.

VAX_HP_INT2

Description

Converts an INTEGER*2 number from VAX to HP 1000 format.

Format

VAX_HP_INT2 (*vax, hp*)

Arguments

vax The VAX number to be converted.

Usage: VAX number.
Type: Word.
Access: Read only.
Mechanism: By reference.

hp The HP 1000 number after conversion.

Usage: HP 1000 number.
Type: Word.
Access: Write only.
Mechanism: By reference.

Operation

Converts a VAX-format number into a HP 1000-format number, suitable for sending to an HP 1000.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL Normal successful completion.

Notes:

VAX_HP_INT4

Description

Converts an INTEGER*4 number from VAX to HP 1000 format.

Format

VAX_HP_INT4 (*vax, hp*)

Arguments

vax The VAX number to be converted.

Usage: VAX number.
Type: Longword.
Access: Read only.
Mechanism: By reference.

hp The HP 1000 number after conversion.

Usage: HP 1000 number.
Type: Longword.
Access: Write only.
Mechanism: By reference.

Operation

Converts a VAX-format number into a HP 1000-format number, suitable for sending to an HP 1000.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL Normal successful completion.

Notes:

Description

Converts a VAX REAL*4 floating point number into an HP 1000 floating point number.

Format

VAX_HP_REAL (*vax*, *hp*)

Arguments

vax The number to be converted.

Usage: VAX number (REAL*4).
Type: F_FLOAT.
Access: Read only.
Mechanism: By reference.

hp The number after conversion.

Usage: HP 1000 number (REAL*4).
Type: Unsigned longword.
Access: Write only.
Mechanism: By reference.

Operation

This function converts a VAX floating point number into the HP 1000 equivalent.

Note that some small loss of precision (less than 0.004%) can be expected.

Returns

Usage: Condition code.
Type: Unsigned longword.
Access: Write only.
Mechanism: By value.

Condition Values Returned

Condition values returned are as follows.

Success:

SS$_NORMAL Normal successful completion.

Notes:

ADDOPT 7, 19, 20
address 3, 4
adrof 7
asynchronous 34

BYPASS 5

C$LIBRARY 11
Change 33
Checksum 8, 30, 48
client 3
Completes 43
conditional compilation 11
Connect 29
connection 43, 47, 59
Converts 21, 23, 25, 69, 71, 73
Creates 37, 39

debug 63
destination descriptor 30, 39
Dumps 61

Errors, by name 15
Errors, by number 13
Errors, by value 17
exceptional 52

FORT$LIBRARY 11
Fortran 9, 11

header 11
HOSTMASTER 3
HP 1000 4, 7, 65
HP 1000 floating point 25
HP 1000 floating point number 73
HP 1000-format number 21, 23, 69, 71
HP 9000 11, 12
HP_UX_COMPATIBILITY 11
HP_VAX_INT2 8, 21
HP_VAX_INT4 8, 23
HP_VAX_REAL 8, 25

INITOPT 27
IPC_DBG_DUMP 7, 11, 61
IPC_DBG_OUTPUT 7, 63
IPC_DEF_USER_C 11
IPC_DEF_USER_FOR 11
IPC_TEST_VAX 8, 65
IPCCONNECT 8, 29, 43
IPCCONTROL 33
IPCCREATE 8, 37
IPCDEST 8, 39
ipcget 7
ipcgive 7
ipclookup 7
ipcname 7
ipcnamerase 7
IPCRECV 8, 43
IPCRECVCN 8, 47
IPCSELECT 8, 11, 51
IPCSEND 8, 55
IPCSHUTDOWN 59

loc 7
logical name 11, 61

monitor 52

NETIPC_DUMP 61
NETMBX 5
network number 3
NIC 3
NSC_NBIO_DISABLE 34
NSC_NBIO_ENABLE 34
NSC_RECV_THRESH_RESET 34
NSC_SEND_THRESH_RESET 34
NSC_TIMEOUT_RESET 34
NSF_DATA_WAIT 44
NSF_MORE_DATA 56
NSF_PREVIEW 44
NSF_VECTORED 44, 56
NSO_DATA_OFFSET 44, 56
NSO_MAX_CONN_REQ_BACK 38
NSO_MAX_RECV_SIZE 30, 48
NSO_MAX_SEND_SIZE 30, 48
NSO_PROTOCOL_ADDRESS 8, 38
NSR_NETWORK_DOWN 5

OPER 5
options array 19, 27, 67

Porting 11
privileges 5
proxy 12

read threshold 34
READOPT 67
Receives 43, 47
Releases 59

Send 55
server 3
shutdown 59
socket 3, 29, 33, 37, 51, 52
status 51
synchronous 34
SYSPRV 5

timeout 34

UCX 4, 8, 40

VAX 65
VAX floating point number 73
VAX-format number 21, 23, 25, 69, 71
VAX_HP_INT2 8, 69
VAX_HP_INT4 8, 71
VAX_HP_REAL 8, 73
vector 8
VMS status value 7
VMS/ULTRIX Connection 3, 4

well-known address 5, 7, 8, 39
wide-area network 3
write threshold 34