intel

iRMX® Human Interface System Calls Reference Manual

Order Number: 462918-001

intel®

iRMX®
Human Interface
System Calls
Reference Manual

Order Number: 462918-001

Intel Corporation
3065 Bowers Avenue
Santa Clara, California 95051

In locations outside the United States, obtain additional copies of Intel documentation by contacting your local Intel sales office. For your convenience, international sales office addresses are located directly after the reader reply card in the back of the manual.

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update or to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

Intel software products are copyrighted by and shall remain the property of Intel Corporation. Use, duplication or disclosure is subject to restrictions stated in Intel's software license, or as defined in ASPR 7-104.9 (a) (9).

No part of this document may be copied or reproduced in any form or by any means without prior written consent of Intel Corporation.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

Above	iLBX	iPSC	Plug-A-Bubble
BITBUS	i_m	iRMX	PROMPT
COMMputer	iMDDX	iSBC	Promware
CREDIT	iMMX	iSBX	QUEST
Data Pipeline	Insite	iSDM	QueX
Genius	int_el	iSSB	Ripplemode
$\hat{i}$	Intel376	iSXM	RMX/80
i	Intel386	Library Manager	RUPI
I^2ICE	int_elBOS	MCS	Seamless
ICE	Intelevision	Megachassis	SLD
iCEL	$int_eligent$ Identifier	MICROMAINFRAME	UPI
iCS	$int_eligent$ Programming	MULTIBUS	VLSiCEL
iDBP	Intellec	MULTICHANNEL	376
iDIS	Intellink	MULTIMODULE	386
	iOSP	OpenNET	386SX
	iPDS	ONCE	
	iPSB		

XENIX, MS-DOS, Multiplan, and Microsoft are trademarks of Microsoft Corporation. UNIX is a trademark of Bell Laboratories. Ethernet is a trademark of Xerox Corporation. Centronics is a trademark of Centronics Data Computer Corporation. Chassis Trak is a trademark of General Devices Company, Inc. VAX and VMS are trademarks of Digital Equipment Corporation. Smartmodem 1200 and Hayes are trademarks of Hayes Microcomputer Products, Inc. IBM, PC/XT, and PC/AT are registered trademarks of International Business Machines. Soft-Scope is a registered trademark of Concurrent Sciences.

REV.	REVISION HISTORY	DATE
-001	Original Issue.	02/89

PREFACE

This manual documents the system calls of the Human Interface, a subsystem of the iRMX® I and iRMX II Operating Systems. The information provided in this manual is intended as a reference to the system calls and provides detailed descriptions of each call.

READER LEVEL

This manual is intended for programmers who are familiar with the concepts and terminology introduced in the *iRMX® I Nucleus User's Guide* and the *iRMX® II Nucleus User's Guide* and with the PL/M programming language.

CONVENTIONS

System call names appear as headings on the outside upper corner of each page. The first appearance of each system call name is printed in blue ink; subsequent appearances are in black.

Throughout this manual, system calls are shown using a generic shorthand (such as CGETCHAR instead of RQCGET$CHAR). This convention is used to allow easier alphabetic arrangement of the calls. The actual PL/M external-procedure names must be used in all calling sequences.

You can also invoke the system calls from assembly language, but you must obey the PL/M calling sequences when doing so. For more information on these calling sequences refer to the *iRMX® I Programming Techniques Reference Manual* or the *iRMX® II Programming Techniques Reference Manual.*

CONTENTS

Chapter 1. iRMX® Human Interface System Calls

1.1 Introduction 1
1.2 System Call Dictionary 3
C$BACKUP$CHAR 5
C$CREATE$COMMAND$CONNECTION 6
C$DELETE$COMMAND$CONNECTION 10
C$FORMAT$EXCEPTION 11
CGETCHAR 13
CGETCOMMAND$NAME 15
CGETINPUT$CONNECTION 17
CGETINPUT$PATHNAME 23
CGETOUTPUT$CONNECTION 29
CGETOUTPUT$PATHNAME 36
CGETPARAMETER 39
C$SEND$COMMAND 43
C$SEND$CO$RESPONSE 50
C$SEND$EO$RESPONSE 53
CSETCONTROL$C 56
CSETPARSE$BUFFER 58

Index

iRMX® HUMAN INTERFACE SYSTEM CALLS 1

1.1 INTRODUCTION

The Human Interface system calls described in this manual are presented in alphabetical sequence and are not organized by function. However, the calls are grouped according to function in the System Call Dictionary. For each call, the following information is provided:

- Brief functional description
- Calling sequence format
- Input parameter definitions, if applicable
- Output parameter definitions, if applicable
- Considerations and consequences of call usage
- Potential exception codes and their possible causes

This manual refers to PL/M data types such as BYTE, WORD, and SELECTOR, and iRMX data types such as STRING. These words, when used as data types, are always capitalized; their definitions are found in Appendix A of the *iRMX® Human Interface User's Guide*. This manual also refers to an iRMX data type called TOKEN. You declare a TOKEN to be literally a SELECTOR. The word "token" in lowercase refers to a value that the iRMX Operating System assigns to an object. The operating system returns this value to a TOKEN (the data type) when it creates the object.

NOTE

The POINTER value of NIL is used throughout this manual. For the iRMX I Operating System, you may also use a value of zero in place of NIL. However, Intel recommends that you use NIL in your iRMX I code to maintain upward compatibility with the iRMX II Operating System. For a description of the NIL built-in function, refer to the PL/M-86 or PL/M-286 user's guides.

If you are a new user of the Human Interface calls, you should review the parsing considerations in the *iRMX® Human Interface User's Guide* before writing your source code. You should also review the format of the released Human Interface commands. They are described in the *Operator's Guide To The iRMX® Human Interface.*

This manual assumes that you are familiar with terms and concepts of the iRMX Operating System. If you are not, you should read *Introduction To The iRMX® Operating System* and the chapters in the *iRMX® I Nucleus User's Guide* or the *iRMX® II Nucleus User's Guide* that refer to the terms "memory pool" and "catalog."

1.2 SYSTEM CALL DICTIONARY

I/O PROCESSING		
Call	**Description**	**Page**
CGETINPUT$CONNECTION	Return an EIOS connection for the specified input file.	17
CGETOUTPUT$CONNECTION	Return an EIOS connection for the specified output file.	29
COMMAND PARSING		
C$BACKUP$CHAR	Move the parsing buffer pointer back one byte.	5
CGETCHAR	Get a character from the command line.	13
CGETINPUT$PATHNAME	Parse the command line and return an input pathname.	23
CGETPARAMETER	Parse the command line for the next parameter and return it as a keyword name and a value.	39
CGETOUTPUT$PATHNAME	Parse the command line and return an output pathname.	36
CSETPARSE$BUFFER	Parse a buffer other than the current command line.	58
CGETCOMMAND$NAME	Return the command name by which the current command was invoked	15
MESSAGE PROCESSING		
C$FORMAT$EXCEPTION	Create a default message for an exception code and place it in a user buffer.	11
C$SEND$CO$RESPONSE	Send a message to the console output (CO) and read a response from the console input (CI).	50
C$SEND$EO$RESPONSE	Send a message to the operator's terminal and return a response from that terminal.	53

COMMAND PROCESSING		
Call	**Description**	**Page**
C$CREATE$COMMAND$-CONNECTION	Create a command connection and return a token.	6
C$DELETE$COMMAND$-CONNECTION	Delete a specific command connection.	10
C$SEND$COMMAND	Concatenate command lines into the data structure created by CREATE$COMMAND$CONNECTION and then invoke the command.	43
PROGRAM CONTROL		
CSETCONTROL$C	Change the default response for a CONTROL-C.	56

C$BACKUP$CHAR, a command parsing call, moves the parsing buffer pointer back one byte.

```
CALL RQ$C$BACKUP$CHAR(except$ptr);
```

Output Parameter

except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

When an operator invokes a command, the command's parameters are placed in a parsing buffer. The C$BACKUP$CHAR system call allows you to move the parsing buffer's pointer back one character for each occurrence of the call.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$LIMIT	0004H	The parsing buffer's pointer is at the start of the command.
E$CONTEXT	0005H	The job that issued the call is not an I/O job.

C$CREATE$COMMAND$CONNECTION

C$CREATE$COMMAND$CONNECTION, a command processing call, creates an iRMX object called a command connection that is required in order to invoke commands programmatically.

```
command$conn = RQ$C$CREATE$COMMAND$CONNECTION(default$ci, default$co,
                                              flags, except$ptr);
```

Input Parameters

default$ci	A TOKEN for a connection that is used as the :CI: (console input) for any commands you invoke using this command connection.
default$co	A TOKEN for a connection that is used as the :CO: (console output) for any commands you invoke using this command connection.
flags	A WORD used to indicate that the Human Interface should return an E$ERROR$OUTPUT exception code if the system call C$SEND$EO$RESPONSE is used by any task. If the user wants the exception code, then the parameter is set to one (1); otherwise, the parameter must equal zero (0).

Output Parameters

command$conn	A TOKEN which receives a token for the new command connection.
except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

You can use this call when you want to invoke a command programmatically instead of interactively. It provides a place to store command lines until the command invocation is complete.

The call creates an iRMX object called a command connection and returns a token for that command connection. The C$SEND$COMMAND system call can use this token to send command lines to the command connection, where they are stored until the command invocation is complete. The command connection also defines default :CI: and :CO: connections that are used by any commands invoked via this command connection.

Although a job can contain multiple command connections, the tasks in a job cannot create command connections simultaneously. Attempts to do this result in an E$CONTEXT exception code. Therefore, it is advisable for one task to create the command connections for all tasks in the job.

A possible application where the parameter "flags" might be set to one is when you want to write a custom CLI to perform batch jobs in the background. When any of the background batch jobs attempt to communicate with the terminal through C$SEND$EO$RESPONSE, the Human Interface issues an exception code. In this way, the Human Interface keeps all the jobs in the background. Note that the Human Interface CLI does not provide resident background or batch processing capability.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$ALREADY$ATTACHED	0038H	While creating a STREAM file, the Extended I/O System was unable to attach the :STREAM: device because another task had already invoked a Basic I/O system call to attach the :STREAM: device.
E$CONTEXT	0005H	At least one of the following is true: • Two command connections were being created simultaneously by two tasks in the same job. • The calling task's job was not created by the Human Interface.(Refer to the *iRMX® Extended I/O System User's Guide* for information.)
EDEVDETACHING	0039H	The :STREAM: device, the default$ci device, or the default$co device was in the process of being detached.
E$DEVFD	0022H	The Extended I/O System attempted the physical attachment of the :STREAM: device. This device had formerly been only logically attached. In the process, the Extended I/O System found that the device and the device driver specified in the logical attachment are incompatible. The operating system would not have returned this exception code if the :STREAM: device had been properly configured.

E$EXIST	0006H	The default$ci or default$co parameter is not a token for an existing object.
E$FNEXIST	0021H	The :STREAM: file does not exist or is marked for deletion.
E$IFDR	002FH	The Extended I/O System attempted to obtain information about the default$ci or default$co connection. However, the request for information resulted in an invalid file driver request.
E$INVALID$FNODE	003DH	The fnode associated with the specified file (:CI: or :CO:) is invalid.
EIOMEM	0042H	The Basic I/O System job does not currently have a block of memory large enough to allow the Human Interface to create a stream file.
E$LIMIT	0004H	At least one of the following is true: • The object directory of the calling task's job has already reached the maximum object directory size. • The calling task's job has exceeded its object limit. • The calling task's job (or that job's default user object) is already involved in 255 (decimal) I/O operations. • The calling task's job was not created by the Human Interface. (Refer to the *iRMX® Extended I/O System User's Guide* for information.)
ELOGNAME$NEXIST	0045H	The call was unable to find the logical name :STREAM: in the object directories of the local job, the global job, or the root job.
E$MEM	0002H	The memory available to the calling task's job is not sufficient to complete the call.
E$NOPREFIX	8022H	The calling task's job does not have a valid default prefix.

ENOTCONNECTION	8042H	The default$ci or default$co parameter is a token for an object that is not a connection to a file.
ENOTLOG$NAME	8040H	The logical name :STREAM: refers to an object that is not a file or device connection.
E$NOUSER	8021H	The calling task's job does not have a valid default user object.
E$PARAM	8004H	The system call forced the Extended I/O System to attempt the physical attachment of the :STREAM: device, which had formerly been only logically attached. In the process, the Extended I/O System found that the stream file driver is not properly configured into your system, so the physical attachment is not possible.
E$SUPPORT	0023H	The default$ci or default$co device connection was not created by this job.

C$DELETE$COMMAND$CONNECTION

C$DELETE$COMMAND$CONNECTION, a command processing call, deletes a command connection object and frees the memory used by the command connection's data structures.

```
CALL RQ$C$DELETE$COMMAND$CONNECTION(command$conn, except$ptr);
```

Input Parameter

command$conn	A TOKEN for a valid command connection.

Output Parameter

except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

This call deletes a command connection object previously defined in a C$CREATE$COMMAND$CONNECTION call and releases the memory used by the command connection's data structures.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$EXIST	0006H	The command$conn parameter is not a token for an existing object.
E$TYPE	8002H	The command$conn parameter is a token for an object that is not a command connection object.

C$FORMAT$EXCEPTION, a message processing call, creates a default message for a given exception code and writes that message into a user-provided string.

```
CALL RQ$C$FORMAT$EXCEPTION(buff$p, buff$max, exception$code,
                           reserved$byte, except$ptr);
```

Input Parameters

buff$max	A WORD that specifies the maximum number of bytes that may be contained in the string pointed to by buff$p.
exception$code	A WORD containing the exception code value for which a message is to be created.
reserved$byte	A BYTE reserved for future use. Its value must be one (1).

Output Parameters

buff$p	A POINTER to a STRING into which the Human Interface concatenates the formatted exception message.
except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

C$FORMAT$EXCEPTION causes the Human Interface to create a message for the exception code. The message consists of the exception code value and exception code mnemonic in the following format:

value : mnemonic

where the mnemonics are provided by the Human Interface from an internal table and are listed in the *Operator's Guide To The iRMX® Human Interface*.

The call concatenates the message to the end of the string pointed to by the buff$p pointer and updates the count byte to reflect the addition. If a string is not already present in the buffer, the first byte of the buffer must be a zero. The message added by C$FORMAT$EXCEPTION will not be longer than 30 characters (not including the length byte).

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$PARAM	8004H	An undefined exception code value was specified.
E$STRING	8084H	The message to be returned exceeds the length limit of 255 characters.
E$STRING$BUFFER	0081H	The buffer pointed to by the buff$p parameter is not large enough to contain the exception message.

CGETCHAR, a command parsing call, gets a character from the parsing buffer.

```
char = RQ$C$GET$CHAR(except$ptr);
```

Output Parameters

char	A BYTE in which the Human Interface places the next character from the parsing buffer. A null (00H) character is returned when the parsing buffer's pointer is at the end of the parsing buffer.
except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

When an operator invokes a command, the command's parameters are placed in a parsing buffer. The CGETCHAR system call gets a single character from that buffer and moves the parsing buffer pointer to the next character. Consecutive calls to CGETCHAR return consecutive characters from the parsing buffer.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$CONTEXT	0005H	The calling task's job was not created by the Human Interface. Refer to the *iRMX® Extended I/O System User's Guide* for information.
E$LIMIT	0004H	At least one of the following situations occurred: • The object directory of the calling task's job has already reached the maximum object directory size. • The calling task's job has exceeded its object limit. • The calling task's job was not created by the Human Interface. Refer to the *iRMX® Extended I/O System User's Guide* for information.

E$MEM	0002H	The memory available to the calling task's job is not sufficient to complete the call.

CGETCOMMAND$NAME, a command parsing call, obtains the pathname of the command that the operator used when invoking the command.

```
CALL RQ$C$GET$COMMAND$NAME (path$name$p, name$max, except$ptr);
```

Input Parameter

name$max	A WORD that specifies the maximum length in bytes of the string pointed to by the path$name$p parameter.

Output Parameters

path$name$p	A POINTER to a buffer that receives a STRING containing the name of the command.
except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

If a command needs to know the name under which it was invoked, the CGETCOMMAND$NAME returns this information. This information is available to each command and is stored in a buffer that is separate from the parsing buffer. Therefore, calling CGETCOMMAND$NAME does not obtain information from the parsing buffer, nor does it move the parsing pointer.

If the operator invokes the command without specifying a logical name, the Human Interface automatically searches a number of directories for the command. In such cases, the value returned by CGETCOMMAND$NAME also includes the directory name (such as :SYSTEM:, :PROG:, or :$:) as a prefix to the command name.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$LIMIT	0004H	The calling task's job was not created by the Human Interface.
E$PATHNAME$SYNTAX	003EH	The specified pathname contains invalid characters.

E$PATHNAME$SYNTAX	003EH	The specified pathname contains invalid characters.
E$STRING$BUFFER	0081H	The buffer pointed to by the path$name$p parameter is not large enough to contain the command name.
E$TIME	0001H	The calling task's job was not created by the Human Interface.

CGETINPUT$CONNECTION, an I/O processing call, returns an Extended I/O System connection to the specified input file.

```
connection = RQ$C$GET$INPUT$CONNECTION(path$name$p, except$ptr);
```

Input Parameter

path$name$p	A POINTER to a buffer that receives a STRING. (The path of the specified input file.)

Output Parameters

connection	A TOKEN in which the operating system returns the token for the connection to the specified pathname.
except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

CGETINPUT$CONNECTION obtains a connection to the specified file. This connection is open for reading and has the following attributes:

- Read only
- Accessible to all users
- Has two 1024-byte buffers (This is the default size.)

CGETINPUT$CONNECTION causes an error message to be displayed at the operator's terminal (:CO:) whenever the operating system encounters an exceptional condition. The exceptional condition that triggers the error message can either be one of those listed for CGETINPUT$CONNECTION or it can be one of those associated with the Extended I/O System calls S$ATTACH$FILE and S$OPEN. The following messages can occur:

- `<pathname>, file does not exist`

  The input file does not exist.

- `<pathname>, invalid file type`

  The input file was a data file and a directory was required, or vice versa.

- `<pathname>, invalid logical name`

  The input pathname contains a logical name that is longer than 12 characters, that contains unmatched colons, invalid characters, or zero characters.

- `<pathname>, logical name does not exist`

  The input pathname contains a logical name that does not exist.

- `<pathname>, READ access required`

  The user does not have read access to the input file.

- `<pathname>, <exception value>:<exception mnemonic>`

  An exceptional condition occurred when CGETINPUT$CONNECTION attempted to obtain the input connection. The <exception value> and <exception mnemonic> portions of the message indicate the exception code encountered. Refer to "Exception Codes" in this call description and to the descriptions of S$ATTACH$FILE and S$OPEN in the *iRMX® Extended I/O System Calls Reference Manual.*

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$ALREADY$ATTACHED	0038H	The device containing the file specified in the path$name$p parameter is already attached.
E$CONTEXT	0005H	At least one of the following is true: • The calling task's job was not created by the Human Interface. (Refer to the *iRMX® Extended I/O System User's Guide* for information.) • The calling task's job was not created by the Human Interface.
EDEVDETACHING	0039H	The device specified in the path$name$p parameter is in the process of being detached.
E$DEVFD	0022H	The call attempted the physical attachment of a device that had formerly been only logically attached. In the process, the call found that the device and the device driver specified in the logical attachment were incompatible.
E$EXIST	0006H	The specified device does not exist.

E$FACCESS	0026H	The specified connection does not have read access to the file.
E$FNEXIST	0021H	At least one of the following is true: • The target file does not exist or is marked for deletion. • While attaching the file pointed to by the path$name$p parameter, the call attempted the physical attachment of the device as a named device. It could not complete this process because the device specified when the logical attachment was made was not defined during configuration.
E$FTYPE	0027H	The path pointed to by the path$name$p parameter contained a file name that should have been the name of a directory, but is not. (Except for the last path component, each file in a pathname must be a named directory.)
E$ILLVOL	002DH	The call attempted the physical attachment of the specified device as a named device. This device had formerly been only logically attached. The call found that the volume did not contain named files. This prevented the call from completing physical attachment because the named file driver was requested during logical attachment.
E$INVALID$FNODE	003DH	The fnode for the specified file is invalid, so the file must be deleted.
EIOHARD	0052H	While attempting to access the file specified in the path$name$p parameter, the call detected a hard I/O error. Another call is useless.
EIOMEM	0042H	While attempting to create a connection, the call needed memory from the Basic I/O subsystem's memory pool. However, the Basic I/O System job does not currently have a block of memory large enough to allow this call to run to completion.

EIONOT$READY	0053H	At least one of the following is true: • While attempting to access the file specified in the path$name$p parameter, the call found that the device was off-line. Operator intervention is required. • Communication failed between the local system and the remote server. Operator intervention is required.
EIOSOFT	0051H	While attempting to access the file specified in the path$name$p parameter, the call detected a soft I/O error. It tried the operation again but was unsuccessful. Another try might be successful.
EIOUNCLASS	0050H	An unknown type of I/O error occurred while this call tried to access the file given in the path$name$p parameter.
E$LIMIT	0004H	At least one of the following is true: • The calling task's job or the job's default user object is already involved in 255 (decimal) I/O operations. • The calling task's job was not created by the Human Interface. • The object limit of the calling job has been reached. • Processing this call would deplete the remote server's resources. For a list of remote server resources, refer to the *iRMX® Networking Software User's Guide*.
ELOGNAME$NEXIST	0045H	The pathname for the specified device contains an explicit logical name. The call was unable to find this name in the object directories of the local job, the global job, or the root job.
ELOGNAME$SYNTAX	0040H	The pathname pointed to by the path$name$p parameter contains a logical name. This logical name contains an unmatched colon, is longer than 12 characters, has zero (0) characters, or contains invalid characters.

E$MEDIA	0044H	The specified device was off-line. If the device has removable media, the media may not be in place.
E$MEM	0002H	The memory available to the calling task's job is not sufficient to complete the call.
E$NOPREFIX	8022H	The calling task's job does not have a valid default prefix.
ENOTLOG$NAME	8040H	The logical name specified by the path$name$p parameter does not refer to a file or device connection.
E$NOUSER	8021H	The calling task's job does not have a valid default user.
E$PARAM	8004H	At least one of the following is true: • The system call forced the Extended I/O System to attempt the physical attachment of the device referenced by the path$name$p parameter. This device had formerly been only logically attached. In the process, the Extended I/O System found that the logical attachment referred to a file driver (named, physical, or stream) that is not configured into your system, so the physical attachment is not possible. • The connection to the specified file cannot be opened for reading.
E$PASSWORD$-MISMATCH	004BH	The password of the user object does not match the password of the corresponding user defined on the remote server.
E$PATHNAME$SYNTAX	003EH	The specified pathname contains invalid characters.
E$SHARE	0028H	The file sharing attribute currently does not allow new connections to the file to be opened for reading.
E$STREAM$SPECIAL	003CH	The call attempted to attach a stream file and in so doing issued an invalid stream file request.

EUDFIO	02D0H	An error occurred while accessing the remote server's User Definition File (UDF). The server's UDF must have world read permission.

CGETINPUT$PATHNAME, a command parsing call, gets a pathname from the list of input pathnames in the parsing buffer.

```
CALL RQ$C$GET$INPUT$PATHNAME(path$name$p, path$name$max,
                             except$ptr);
```

Input Parameter

path$name$max — A WORD that specifies the maximum length in bytes of the string pointed to by the path$name$p parameter. The maximum length that you can specify is 256 bytes (255 characters for the pathname and one byte for the count).

Output Parameters

path$name$p — A POINTER to a STRING which receives the next pathname in the pathname list. A zero-length string indicates that there are no more pathnames.

except$ptr — A POINTER to a WORD in which the Human Interface returns a condition code.

Description

The first call to CGETINPUT$PATHNAME retrieves the entire input pathname list and moves the parsing pointer to the next parameter. CGETINPUT$PATHNAME stores the list in an internal buffer and returns the first pathname in the string pointed to by the path$name$p parameter. Succeeding calls to CGETINPUT$PATHNAME return additional pathnames from the input pathname list but do not move the parsing pointer. CGETINPUT$PATHNAME denotes the end of the pathname list by returning a zero-length string.

CGETINPUT$PATHNAME accepts wild-card characters in the last component of a pathname. It treats a pathname that contains a wild-card as a list of pathnames. To obtain each pathname, it searches in the parent directory of the component containing the wild-card, comparing the "wild-carded" name with the names of all files in the directory. It returns the next pathname that matches.

The pathname returned by CGETINPUT$PATHNAME can be used for any purpose. However, it is most often used in a call to CGETINPUT$CONNECTION, to obtain a connection.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$ALREADY$ATTACHED	0038H	The device containing the file pointed to by the path$name$p parameter is already attached.
E$CONTEXT	0005H	At least one of the following is true: • The calling task's job was not created by the Human Interface. (Refer to the *iRMX® Extended I/O System User's Guide* for more information.) • The task called CGETOUTPUT$PATHNAME before calling CGETINPUT$PATHNAME.
EDEVDETACHING	0039H	The device pointed to by the path$name$p parameter is in the process of being detached.
E$DEVFD	0022H	The Extended I/O System attempted the physical attachment of a device that had formerly been only logically attached. In the process, the Extended I/O System found that the device and the device driver specified in the logical attachment were incompatible.
E$EXIST	0006H	At least one of the following is true: • The connection to the parent directory of the file pointed to by the path$name$p parameter is not a token for the existing job. • The calling task's job was not created by the Human Interface.
E$FACCESS	0026H	The connection used to open the directory does not have read access to the directory.
E$FLUSHING	002CH	The device containing the directory was in the process of being detached.

E$FNEXIST	0021H	At least one of the following is true: • The target file does not exist or is marked for deletion. • While attaching the parent directory of the file pointed to by the path$name$p parameter, the I/O System attempted the physical attachment of the device as a named device. It could not complete this process because the device specified when the logical attachment was made was not defined during configuration.
E$FTYPE	0027H	The path pointed to by the path$name$p parameter contained a file name that should have been the name of a directory, but is not. (Except for the last file, each file in a pathname must be a named directory.)
E$IFDR	002FH	The specified file is a stream or physical file.
E$ILLVOL	002DH	The call attempted the physical attachment of the specified device as a named device. This device had formerly been only logically attached. The call found that the volume did not contain named files. This prevented the call from completing physical attachment because the named file driver was requested during logical attachment.
E$INVALID$FNODE	003DH	The fnode for the specified file is invalid, so the file must be deleted.
EIOHARD	0052H	While attempting to access the parent directory of the file pointed to by the path$name$p parameter, the call detected a hard I/O error. This means that another call is probably useless.
EIOMEM	0042H	While attempting to create a connection, this call needed memory from the Basic I/O System's memory pool. However, the Basic I/O System job does not currently have a block of memory large enough to allow this call to run to completion.

EIONOT$READY	0053H	At least one of the following is true: • While attempting to access the file specified in the path$name$p parameter, the call found that the device was off-line. Operator intervention is required. • Communication failed between the local system and the remote server. Operator intervention is required.
EIOSOFT	0051H	While attempting to access the parent directory of the file pointed to by the path$name$p parameter, this call detected a soft I/O error. It tried the operation again, but was unsuccessful. Another try might be successful.
EIOUNCLASS	0050H	An unknown type of I/O error occurred while this call tried to access the parent directory of the file pointed to by the path$name$p parameter.
E$LIMIT	0004H	At least one of the following is true: • The calling task's job has already reached its object limit. • The calling task's job or the job's default user object is already involved in 255 (decimal) I/O operations. • The calling task's job was not created by the Human Interface. • Processing this call would deplete the remote server's resources. For a list of remote server resources, refer to the *iRMX® Networking Software User's Guide*.
E$LIST	0085H	The last value of the input pathname list is missing. For example, "ABLE,BAKER," has no value following the second comma.
ELOGNAME$NEXIST	0045H	The pathname for the specified device contains an explicit logical name. The call was unable to find this name in the object directory of the local job, the global job, or the root job.

ELOGNAME$SYNTAX	0040H	The pathname pointed to by the path$name$p parameter contains a logical name that has an unmatched colon, is longer than 12 characters, has zero (0) characters, or contains invalid characters.
E$MEDIA	0044H	The specified device was off-line. If the device has removable media, the media may not be in place.
E$MEM	0002H	The memory available to the calling task's job is not sufficient to complete the call.
E$NOPREFIX	8022H	The calling task's job does not have a valid default prefix.
ENOTLOG$NAME	8040H	The logical name specified by the path$name$p parameter does not refer to a file or device connection.
E$NOUSER	8021H	The calling task's job does not have a valid default user object.
E$PARAM	8004H	At least one of the following is true: • The Extended I/O System attempted the physical attachment of the device pointed to by the path$name$p parameter. This device had formerly been only logically attached. In the process, the Extended I/O System found that the logical attachment referred to a file driver (named, physical, or stream) that is not configured into your system, so the physical attachment is not possible. • The connection to the parent directory cannot be opened for reading.
E$PARSE$TABLES	8080H	The call detected an error in an internal table used by the Human Interface.
E$PASSWORD$-MISMATCH	004BH	The password of the user object does not match the password of the corresponding user defined on the remote server.
E$PATHNAME$SYNTAX	003EH	The specified pathname contains invalid characters.

E$SHARE	0028H	The connection to the parent directory cannot be opened for reading.
E$STREAM$SPECIAL	003CH	The Extended I/O System issued an invalid stream file request when an attempt to attach a stream file failed.
E$STRING	8084H	The pathname to be returned exceeds the length limit of 255 characters.
E$STRING$BUFFER	0081H	The buffer pointed to by the path$name$p parameter was not large enough for the pathname to be returned.
E$SUPPORT	0023H	This call attempted to read the parent directory of the pathname pointed to by the path$name$p parameter. However, the file driver corresponding to that directory does not support this operation.
E$WILDCARD	0086H	The pathname to be returned contains an invalid wild-card specification.
EUDFIO	02D0H	An error occurred while accessing the remote server's User Definition File (UDF). The server's UDF must have world read permission.

CGETOUTPUT$CONNECTION, an I/O processing call, parses the command line and returns an Extended I/O System connection referring to the requested output file.

```
connection = RQ$C$GET$OUTPUT$CONNECTION(path$name$p, preposition,
                                        except$ptr);
```

Input Parameters

path$name$p — A POINTER to a STRING containing the pathname of the file to be accessed.

preposition — A BYTE that defines which preposition to use to create the output file. Use one of the following values to specify the preposition mode:

Value	Meaning
0	Use same preposition as was returned by the last CGETOUTPUT$PATHNAME call
1	TO
2	OVER
3	AFTER
4-255	Undefined, results in an error

Output Parameters

connection — A TOKEN in which the Human Interface returns a token for the connection to the output file.

except$ptr — A POINTER to a WORD in which the Human Interface returns a condition code.

CGETOUTPUT$CONNECTION

Description

CGETOUTPUT$CONNECTION obtains a connection to the specified file.

This connection is open for writing and has the following attributes:

- Write only
- Accessible to all
- Has two 1024-byte buffers

If the call to CGETOUTPUT$CONNECTION specifies the TO preposition and the output file already exists, CGETOUTPUT$CONNECTION issues the following message to the terminal (:CO:):

```
<pathname>, already exists, OVERWRITE?
```

If the operator enters Y, y, R, or r, CGETOUTPUT$CONNECTION returns a connection to the existing file, allowing the command to write over the file. Any other response causes CGETOUTPUT$CONNECTION to return an E$FACCESS exception code.

CGETOUTPUT$CONNECTION causes an error message to be displayed at the operator's terminal (:CO:) whenever an exceptional condition occurs. The exceptional condition that causes the error message can be one of those listed below or one associated with an Extended I/O System call. The following messages can occur:

- `<pathname>, DELETE access required`

  The user does not have delete access to an existing file.

- `<pathname>, directory ADD entry access required`

  The user does not have add entry access to the parent directory.

- `<pathname>, file does not exist`

  The output file does not exist.

- `<pathname>, invalid file type`

  The output file was a data file and a directory was required, or vice versa.

- `<pathname>, invalid logical name`

  The output pathname contains a logical name longer than 12 characters, contains unmatched colons, contains invalid characters, or zero characters.

- `<pathname>, logical name does not exist`

  The output pathname contains a logical name that does not exist.

- <pathname>, <exception value>:<exception mnemonic>

  An exceptional condition occurred when CGETOUTPUT$CONNECTION attempted to obtain the output connection. The <exception value> and <exception mnemonic> portions of the message indicate the exception code encountered. Refer to "Exception Codes" in this call description and to the *iRMX® Extended I/O System User's Guide*.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$ALREADY$ATTACHED	0038H	The Extended I/O System was unable to attach the device containing the file because the Basic I/O System has already attached the device.
E$CONTEXT	0005H	The calling task's job was not created by the Human Interface.
EDEVDETACHING	0039H	The device referred to by the path$name$p parameter was in the process of being detached.
E$DEVFD	0022H	The call attempted the physical attachment of a device that had formerly been only logically attached. In the process, the call found that the device and the device driver specified in the logical attachment were incompatible.
E$EXIST	0006H	The connection parameter for the device containing that file is not a token for an existing object.
E$FACCESS	0026H	At least one of the following is true: • The default user for the calling task's job did not have update access to an existing file and/or add-entry access to the parent directory. • The TO or OVER preposition was specified and the default user for the calling task's job did not have the ability to truncate the file.

E$FNEXIST	0021H	At least one of the following is true: • The target file does not exist or is marked for deletion. • While attaching the file pointed to by the path$name$p parameter, the Extended I/O System attempted the physical attachment of the device as a named device. It could not complete this process because the device specified when the logical attachment was made was not defined during configuration.
E$FTYPE	0027H	The path pointed to by the path$name$p parameter contained a file name that should have been the name of a directory, but is not. (Except for the last component, each file in a pathname must be a named directory.)
E$IFDR	002FH	The call requested information about the specified file, but the request was an invalid file driver request.
E$ILLVOL	002DH	The call attempted the physical attachment of the specified device as a named device. This device had formerly been only logically attached. The call found that the volume did not contain named files. This prevented the call from completing physical attachment because the named file driver was requested during logical attachment.
E$INVALID$FNODE	003DH	The fnode for the specified file is invalid, so the file must be deleted.
EIOHARD	0052H	While attempting to access the file specified in the path$name$p parameter, the call detected a hard I/O error. A retry is probably useless.
EIOMEM	0042H	While attempting to create a connection, this call needed memory from the Basic I/O System's memory pool. However, the Basic I/O System job does not currently have a block of memory large enough to allow this call to run to completion.

EIONOT$READY	0053H	At least one of the following is true: • While attempting to access the file specified in the path$name$p parameter, the call found that the device was off-line. Operator intervention is required. • Communication failed between the local system and the remote server. Operator intervention is required.
EIOSOFT	0051H	While attempting to access the file specified in the path$name$p parameter, the call detected a soft I/O error. It tried the operation again but was unsuccessful. Another try might be successful.
EIOUNCLASS	0050H	An unknown type of I/O error occurred while this call tried to access the file given in the path$name$p parameter.
EIOWRPROT	0054H	While attempting to obtain an input connection to the file specified in the path$name$p parameter, this call found that the volume containing the file is write-protected.
E$LIMIT	0004H	At least one of the following is true: • The calling task's job or the job's default user object is already involved in 255 (decimal) I/O operations. • The calling task's job was not created by the Human Interface. • The calling task's job has reached its object limit. (Refer to the *iRMX® Extended I/O System User's Guide* for more information about I/O jobs.) • Processing this call would deplete the remote server's resources. For a list of remote server resources, refer to the *iRMX® Networking Software User's Guide*.
ELOGNAME$NEXIST	0045H	The specified pathname contains an explicit logical name. The call was unable to find this name in the object directory of the local job, the global job, or the root job.

ELOGNAME$SYNTAX	0040H	The pathname pointed to by the path$name$p parameter contains a logical name. However, the logical name contains unmatched colons, is longer than 12 characters, contains invalid characters, or contains zero characters.
E$MEDIA	0044H	The specified device was off-line. If the device has removable media, the media may not be in place.
E$MEM	0002H	The memory available to the calling task's job is not sufficient to complete the call.
E$NOPREFIX	8022H	The calling task's job does not have a valid default prefix.
ENOTLOG$NAME	8040H	The logical name specified by the path$name$p parameter does not refer to a file or device connection.
E$NOUSER	8021H	The calling task's job does not have a valid default user object.
E$PARAM	8004H	The system call forced the Extended I/O System to attempt the physical attachment of the device referenced by the path$name$p parameter. The device had formerly been only logically attached. In the process, the Extended I/O System found that the logical attachment referred to a file driver (named, physical, or stream) that is not configured into your system, so the physical attachment is not possible.
E$PASSWORD$-MISMATCH	004BH	The password of the user object does not match the password of the corresponding user defined on the remote server.
E$PATHNAME$SYNTAX	003EH	The specified pathname contains invalid characters.

E$PREPOSITION	0087H	One of the following is true: • The command line contained an invalid preposition value (a value greater than 3). • The command line contained a zero as the preposition value. This indicated that the same preposition was to be used as in the last call to CGETOUTPUT$CONNECTION. However, this is the first call to CGETOUTPUT$CONNECTION.
E$SHARE	0028H	The new connection cannot be opened for writing.
E$SPACE	0029H	One of the following is true: • The volume is full. • The volume already contains the maximum number of files.
E$STREAM$SPECIAL	003CH	The Extended I/O System issued an invalid stream file request when an attempt to attach a stream file failed.
EUDFIO	02D0H	An error occurred while accessing the remote server's User Definition File (UDF). The server's UDF must have world read permission.

CGETOUTPUT$PATHNAME

CGETOUTPUT$PATHNAME, a command parsing call, gets a pathname from the list of output pathnames in the parsing buffer.

```
preposition = RQ$C$GET$OUTPUT$PATHNAME(path$name$p, path$name$max,
                                       default$output$p, except$ptr);
```

Input Parameters

path$name$max — A WORD that specifies the maximum length in bytes of the string pointed to by the path$name$p parameter. The maximum length that you can specify is 256 bytes (255 characters for the pathname and one byte for the count).

default$output$p — A POINTER to a STRING containing the command's default standard output. If the first invocation of this system call does not encounter a TO/OVER/AFTER preposition, the text of this parameter will be used as though it had appeared in the command line. The text must specify TO, OVER, or AFTER for the output mode. Examples: TO :CO: or TO :LP:.

Output Parameters

preposition — A BYTE describing the preposition type that CGETOUTPUT$PATHNAME encountered. You can pass this value to CGETOUTPUT$CONNECTION when obtaining an output connection to the file. The value will be one of the following:

Value	Meaning
1	TO
2	OVER
3	AFTER

path$name$p — A POINTER to a buffer that receives a STRING. (The next pathname in the pathname list.)

except$ptr — A POINTER to a WORD in which the Human Interface returns a condition code.

Description

You should not call CGETOUTPUT$PATHNAME before first calling CGETINPUT$PATHNAME.

The first call to CGETOUTPUT$PATHNAME retrieves the preposition (TO/OVER/AFTER) and the entire output pathname list; it then moves the parsing pointer to the next parameter. If the parsing buffer does not contain a preposition and pathname list, CGETOUTPUT$PATHNAME uses the default pointed to by the default$output$p parameter (and does not move the parsing pointer). After retrieving the pathname list, CGETOUTPUT$PATHNAME stores it in an internal buffer, returns the first pathname in the string pointed to by the path$name$p parameter, and returns the preposition in the preposition parameter. Succeeding calls to CGETOUTPUT$PATHNAME return additional pathnames from the output pathname list (as well as the preposition), but they do not move the parsing pointer. CGETOUTPUT$PATHNAME denotes the end of the pathname list by returning a zero-length string in the STRING pointed to by path$name$p.

CGETOUTPUT$PATHNAME accepts characters with a wild-card as the last component of a pathname. It generates each output pathname based on this pathname and wild-card, the corresponding pathname and wild-card that was input to CGETINPUT$PATHNAME, and the most recent input pathname returned by CGETINPUT$PATHNAME.

The pathname returned by CGETOUTPUT$PATHNAME can be used for any purpose. However, it is most often used in a call to CGETOUTPUT$CONNECTION to obtain a connection to the file. In such a case, CGETOUTPUT$CONNECTION processes the TO/OVER/AFTER preposition. If the pathname is used as input to a system call other than CGETOUTPUT$CONNECTION, the interpretation of the TO/OVER/AFTER preposition is the user's responsibility.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$CONTEXT	0005H	The calling task's job was not created by the Human Interface.
E$DEFAULT$SO	8083H	The default output string pointed to by default$output$p contained an invalid preposition or pathname.

E$LIMIT	0004H	At least one of the following is true: • The calling task's job has already reached its object limit. • The calling task's job was not created by the Human Interface. • The calling task's job or the job's default user object is already involved in 255 (decimal) I/O operations.
E$MEM	0002H	The memory available to the calling task's job is not sufficient to complete the call.
E$PATHNAME$SYNTAX	003EH	The specified pathname contains invalid characters.
E$STRING	8084H	The pathname to be returned exceeds the length limit of 255 characters.
E$STRING$BUFFER	0081H	The buffer pointed to by the path$name$p parameter was not large enough for the pathname to be returned.
E$UNMATCHED$LISTS	008BH	The numbers of files in the input and output lists are not the same.
E$WILDCARD	0086H	The output pathname contains an invalid wild-card specification.

CGETPARAMETER, a command parsing call, gets a parameter from the parsing buffer.

```
more = RQ$C$GET$PARAMETER(name$p, name$max, value$p, value$max,
                          index$p, predict$list$p, except$ptr);
```

Input Parameters

name$max	A WORD that specifies the maximum length in bytes of the string pointed to by the name$p parameter. The maximum length is 256 bytes (255 characters for the name and one byte for the count).
value$max	A WORD that specifies the maximum length in bytes of the string pointed to by the value$p parameter. The maximum length is 65535 decimal bytes.
predict$list$p	A POINTER to a STRING$TABLE, as described in Appendix B of the *iRMX® Human Interface User's Guide*, that specifies the values that this system call accepts as prepositions. The predict$list$p POINTER should be NIL if you do not intend to retrieve parameters that use prepositions.

Output Parameters

more	A BYTE value that indicates whether or not the current call to CGETPARAMETER returned a parameter. A value of 00H indicates that there are no more parameters (and that no parameter was returned); a value of 0FFH indicates that a parameter was returned.
name$p	A POINTER to a buffer that receives the keyword portion of the parameter. If this parameter does not contain a keyword portion, the Human Interface returns a null (zero-length) string.
value$p	A POINTER to a buffer used to store a STRING$TABLE, as described in Appendix B of the *iRMX® Human Interface User's Guide*, that receives the value portion of the parameter. If the value portion contains a list of values separated by commas, the Human Interface returns the values to the string table one value per string.

index$p	A POINTER to a BYTE that receives the index to the list of prepositions pointed to by predict$list$p. This index identifies the name$p keyword as a preposition and identifies it out of the list of possible prepositions. If the predict$list$p list is empty, or if the keyword name is not contained in the predict$list$p list, the system call returns a value of zero for the index. That is, the index will be non-zero only if a keyword exists and it is one of the prepositions in the predict$list$p list.
except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

CGETPARAMETER retrieves one parameter from the parsing buffer and moves the parsing pointer to the next parameter. The parameter can be one of the following:

- keyword/value-list parameter using parentheses
- keyword/value-list parameter using an equal sign
- keyword/value-list parameter with the keyword as a preposition
- value-list without a keyword

A description of the types, format, and syntax of acceptable parameters is provided in the *iRMX® Human Interface User's Guide*.

CGETPARAMETER places the keyword portion of the parameter in the string pointed to by name$p; it places the keyword list in the string table pointed to by value$p.

Without input from you, CGETPARAMETER cannot determine whether groups of characters separated by spaces are separate parameters or a single parameter that uses a preposition. CGETPARAMETER uses the list of prepositions that you supply in the string table pointed to by predict$list$p to determine the prepositions that can appear. When CGETPARAMETER retrieves a parameter, it obtains, from the parsing buffer, the next group of characters that are separated by spaces. These characters are checked against those in the predict$list$p list. If the characters match one of the values in the list, CGETPARAMETER realizes that the characters represent a preposition and not an entire parameter; it then obtains the next group of characters separated by spaces as the value portion of the parameter.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$CONTEXT	0005H	The calling task's job was not an I/O job. (Refer to the *iRMX® Extended I/O System User's Guide* for information about I/O jobs.)
E$CONTINUED	0083H	The call found a continuation character in the parse buffer. Command lines should not contain continuation characters.
E$LIMIT	0004H	At least one of the following is true: • The calling task's job has already reached its object limit. • The calling task's job was not an I/O job. (Refer to the *iRMX® Extended I/O System User's Guide* for information about I/O jobs.)
E$LIST	0085H	At least one of the following is true: • The parameter contains an unmatched parenthesis. • A value in the value list is missing or an improper value was entered. Examples of both these conditions are:

Value	Comments
A,B,	No value following second comma.
A,B=C,D	The equal sign can not be used unless it is between quotes: 'B=C' is valid.
A,B(C,E),F	The parentheses can not be used in a value unless it is between quotes or set off by commas. A,B,(C,E),F is valid.

E$LITERAL	0080H	The call found a literal (quoted string) in the parsing buffer with no closing quote. This condition should not occur in the command line buffer.

E$MEM	0002H	The memory available to the calling task's job is not sufficient to complete the call.
E$PARAM	8004H	The predict$list$p parameter pointed to a string table, but the index$p parameter was set to zero (0).
E$PARSE$TABLES	8080H	The call found an error in an internal table used by the Human Interface.
E$SEPARATOR	0082H	The call found an invalid command separator in the parsing buffer. This condition should not occur in the command line buffer. The following is a list of invalid command separators: > <, < >, ||, |, [, and].
E$STRING	8084H	The string to be returned as the parameter name or one of the parameter values exceeds the length limit of 255 characters.
E$STRING$BUFFER	0081H	The string to be returned as the parameter name or one of the parameter values exceeds the buffer size provided in the call.

C$SEND$COMMAND, a command processing call, sends command lines to a command connection created by C$CREATE$COMMAND$CONNECTION and, when the command is complete, invokes the command.

```
CALL RQ$C$SEND$COMMAND(command$conn, line$p, command$except$ptr,
                       except$ptr);
```

Input Parameters

command$conn	A TOKEN for the command connection that receives the command line.
line$p	A POINTER to a buffer used to store a STRING containing a command line to execute.

Output Parameters

command$except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code indicating the status of the invoked command. This parameter is undefined if an exceptional condition code is returned in the WORD pointed to by except$ptr.
except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code indicating the status of the C$SEND$COMMAND system call.

Description

You can use this system call when you want to invoke a command programmatically instead of interactively. It stores a command line in the command connection created by the C$CREATE$COMMAND$CONNECTION call, concatenates the command line with any others already stored there, and (if the command invocation is complete) invokes the command. The command can be any standard Human Interface command (as described in the *Operator's Guide To The iRMX® Human Interface*) or a command that you create.

As described in greater detail in the *Operator's Guide To The iRMX® Human Interface*, a command invocation can contain several continuation marks. The continuation mark (&) indicates that the command line is continued on the next line. If the command line sent by C$SEND$COMMAND is continued on another line (that is, contains a continuation mark), the Human Interface returns an E$CONTINUED exception code and does not invoke the command. You can then call C$SEND$COMMAND any number of times to send the continuation lines.

C$SEND$COMMAND concatenates the original command line and all continuation lines into a single command line in the command connection. It removes all continuation marks and comments from this command line.

When the command invocation is complete (that is, the line sent by C$SEND$COMMAND does not contain a continuation mark), the Human Interface parses the command pathname from the command line. If no exception conditions halt the process at this point, the Human Interface requests the Application Loader to load and execute the command.

An Application Loader call creates an I/O job for the command, and validates the header, group definition and segment definition records of the command's object file. Refer to the *8086 Family Utilities User's Guide* or the *iAPX 286 Utilities User's Guide For iRMX® II Systems* for explanations of segments, groups and object file formats.

C$SEND$COMMAND returns two condition codes: one for the C$SEND$COMMAND call and one for the invoked command. The word pointed to by the except$ptr parameter returns the C$SEND$COMMAND conditions, as described under the "Exception Codes" heading in this command description. The WORD pointed to by the command$except$ptr returns the invoked command's condition codes; the values returned depend on the command invoked. The E$CONTROL$C exception code can be returned at either place.

NOTE

When a C$SEND$COMMAND call is made, the Human Interface sets the CONTROL-C semaphore to the default Human Interface CONTROL-C handler. If you previously set the CONTROL-C handler, it must be set again after making this call. For more information see the *iRMX® Human Interface User's Guide*.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$ALREADY$ATTACHED	0038H	The Extended I/O System was unable to attach the device containing the object file because the Basic I/O System has already attached the device.
EBADGROUP	0061H	The object file represented by the command's pathname contained an invalid group definition record.

EBADHEADER	0062H	The object file represented by the command's pathname does not begin with a header record for a loadable object module.
EBADSEGDEF	0063H	The object file represented by the command's pathname contains an invalid segment definition record.
E$CHECKSUM	0064H	At least one record of the object file represented by the command's pathname contains a checksum error. This situation could occur if the CHECKSUM amount calculated during the read operation did not match the CHECKSUM field of the record being read.
E$CONTEXT	0005H	The calling task's job was not created by the Human Interface.
E$CONTINUED	0083H	The operating system detected a continuation character while scanning the command line pointed to by the line$p parameter. This condition should occur if the command line is to continue on the next line.
EDEVDETACHING	0039H	The device containing the object file was in the process of being detached.
E$DEVFD	0022H	The Extended I/O System attempted the physical attachment of a device that had formerly been only logically attached. In the process, the Extended I/O System found that the device and the device driver specified in the logical attachment were incompatible.
E$EOF	0065H	The Application Loader encountered an unexpected end of file on the object file represented by the command's pathname.
E$EXIST	0006H	At least one of the following is true: • The call detached the device containing the object file before completing the loading operation. • The command$conn parameter is not a TOKEN for a command connection.

E$FACCESS	0026H	The default user for the calling task's job does not have read access to the object file.
E$FLUSHING	002CH	The device containing the object file was being detached.
E$FNEXIST	0021H	At least one of the following is true: • The file in the command's pathname is either marked for deletion or does not exist. • While attaching the file specified in the line$p parameter, the Extended I/O System attempted the physical attachment of the device as a named device. It could not complete this process because the device specified when the logical attachment was made was not defined during configuration.
E$FTYPE	0027H	The path pointed to by the path$name$p parameter contained a component name that should have been the name of a directory, but is not. (Except for the last file, each file in a pathname must be a named directory.)
E$ILLVOL	002DH	The call attempted the physical attachment of the specified device as a named device. This device had formerly been only logically attached. The call found that the volume did not contain named files. This prevented the call from completing physical attachment because the named file driver was requested during logical attachment.
E$INVALID$FNODE	003DH	The fnode for the specified file is invalid, so the file must be deleted.
EIOHARD	0052H	While attempting to access the object file, this call detected a hard I/O error.
EIOMEM	0042H	The Basic I/O System does not currently have enough memory to allow the Human Interface to create the connection necessary to allow this call to run to completion.

EIONOT$READY	0053H	While attempting to access the object file, this call found that the device was off-line. Operator intervention is required.
EIOSOFT	0051H	While attempting to access the object file, this call detected a soft I/O error. It tried again, but was not successful. Another try might be successful.
EIOUNCLASS	0050H	An unknown type of I/O error occurred while this call tried to access the object file.
E$LIMIT	0004H	At least one of the following is true: • The calling task's job has already reached its object limit. • The calling task's job , or the job's default user object, is already involved in 255 (decimal) I/O operations. • The new I/O job, or its default user, is already involved in 255 (decimal) I/O operations. • The calling task's job was not created by the Human Interface. (See to the *iRMX® Extended I/O System User's Guide* for information.)
E$LITERAL	0080H	The call found a literal (quoted string) with no closing quote while scanning the contents of the command line pointed to by the line$p parameter.
ELOGNAME$NEXIST	0045H	The command's pathname contains an explicit logical name, but the call was unable to find this name in the object directory of the local job, the global job, or the root job.
ELOGNAME$SYNTAX	0040H	The pathname pointed to by the path$name$p parameter contains a logical name. However, the logical name contains an unmatched colon, is longer than 12 characters, has zero (0) characters, or contains invalid characters.

E$MEDIA	0044H	The device containing the object file was off-line. If the device has removable media, the media may not be in place.
E$MEM	0002H	The memory available to the calling task's job, the new I/O job, or the Basic I/O System job is not sufficient to complete the call.
ENOLOADER$MEM	0067H	At least one of the following is true: • The memory pool of the newly created I/O job does not currently have a block of memory large enough to allow the Application Loader to run. • The memory pool of the Basic I/O System's job does not currently have a block of memory large enough to allow the Application Loader to run.
E$NOPREFIX	8022H	The calling task's job does not have a valid default prefix.
ENOSTART	006CH	The object file represented by the command pathname does not specify the entry point for the program being loaded.
ENOTCONNECTION	8042H	The default$ci or default$co parameter is a token for an object that is not a file connection.
ENOTLOG$NAME	8040H	The command pathname contains a logical name. The logical name of an object that is neither a device connection nor a file connection.
E$NOUSER	8021H	The calling task's job does not have a valid default user.
E$PARAM	8004H	The Extended I/O System attempted the physical attachment of a device containing the object file. This device had formerly been only logically attached. While attempting this, the Extended I/O System found that the logical attachment referred to a file driver (named, physical, or stream) that is not configured into your system. Hence, the physical attachment is not possible.

E$PARSE$TABLES	8080H	The call found an error in an internal table.
E$PATHNAME$SYNTAX	003EH	The command's pathname contains invalid characters.
ERECFORMAT	0069H	At least one record in the object file contains a record format error.
ERECLENGTH	006AH	The object file contains a record that is longer than the Loader's maximum record length. The Application Loader's maximum record length is a parameter specified during the configuration of the Loader. (Refer to the ICU reference manual for details.)
ERECTYPE	006BH	At least one of the following is true: • At least one record in the file being loaded is of a type that the Application Loader cannot process. • The Application Loader has encountered records in a sequence that it cannot process.
ESEGBOUNDS	0070H	The Application Loader created multiple segments in which to load information. One of the data records in the object file specified a load address outside of the created segments.
E$SEPARATOR	0082H	The call found an invalid separator while scanning the command line. The following is a list of the invalid command separators: > <, < >, | |, |, [, and].
E$STRING	8084H	The size of the command's pathname exceeds the length limit of 255 (decimal) characters.
E$STRING$BUFFER	0081H	The size of the command's pathname exceeds the size of the command name buffer specified during the configuration of the Human Interface.
E$TIME	0001H	The calling task's job was not created by the Human Interface.
E$TYPE	8002H	The command$conn parameter is a token for an object that is not a command connection.

C$SEND$CO$RESPONSE

C$SEND$CO$RESPONSE, a message processing call, sends a message to :CO: and reads a response from :CI:.

```
CALL RQ$C$SEND$CO$RESPONSE(response$p, response$max, message$p,
                           except$ptr);
```

Input Parameters

response$max	A WORD whose value specifies the maximum length in bytes of the string pointed to by the response$p parameter. The value in response$max must equal the length of the string plus one (string length + 1). If response$max is zero or one, no response from :CI: will be requested; control returns to the calling task immediately.
message$p	A POINTER to a STRING containing the message to be sent to :CO:. If NIL, no message is sent.

Output Parameters

response$p	A POINTER to a buffer that receives the operator's response from :CI:.
except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

When used with all its features, C$SEND$CO$RESPONSE sends the string pointed to by message$p to :CO: and waits for a response from :CI:. It places this response in the string pointed to by response$p. However, if message$p is NIL, C$SEND$CO$RESPONSE omits sending the message to :CO:; if either response$max or response$p is NIL, it does not wait for a response from :CI:. Therefore, the operations performed by C$SEND$CO$RESPONSE depend on the values of the message$p and response$max parameters, as follows:

message$p	response$max	Action
NIL	zero	Perform no I/O
NIL	non-zero	Send no message, wait for input
NOT NIL	non-zero	Send message, wait for input
NOT NIL	zero	Send message, don't wait

If C$SEND$CO$RESPONSE requests a response from :CI:, output from other tasks can be displayed at :CO: while the system waits for a response from :CI:.

The difference between the C$SEND$CO$RESPONSE and C$SENDEORESPONSE calls is that C$SEND$EO$RESPONSE always sends messages to and receives messages from the operator's terminal; input and output cannot be redirected to another device. In contrast, C$SEND$CO$RESPONSE sends messages to :CO: and receives messages from :CI:; therefore, programs such as SUBMIT can redirect this input and output.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$CONTEXT	0005H	The calling task's job was not created by the Human Interface.
E$CONN$OPEN	0035H	At least one of the following is true: • The connection to :CI: was not open for reading or the connection to :CO: was not open for writing. • The connection to :CI: or :CO: was not open. • The connection to :CI: or :CO: was opened with A$OPEN rather than S$OPEN.
E$EXIST	0006H	The token value for :CI: or :CO: is not a token for an existing object.
E$FLUSHING	002CH	The device containing the :CI: and :CO: files was being detached.
EIOHARD	0052H	While attempting to access the :CI: or :CO: file, the operating system detected a hard I/O error.
EIONOT$READY	0053H	While attempting to access the :CI: or :CO: file, this call found that the device was off-line. Operator intervention is required.
EIOSOFT	0051H	While attempting to access the :CI: or :CO: file, this call detected a soft I/O error. It tried again, but was unsuccessful. Another try might be successful.
EIOUNCLASS	0050H	An unknown type of I/O error occurred while this call tried to access the :CI: or :CO: file.

EIOWRPROT	0054H	While attempting to obtain a connection to the :CO: file, this call found that the volume containing the file is write-protected.
E$LIMIT	0004H	At least one of the following is true: • The calling task's job has already reached its object limit. • The calling task's job, or the job's default user object, is already involved in 255 (decimal) I/O operations. • The calling task's job was not created by the Human Interface.
E$MEM	0002H	The memory available to the calling task's job is not sufficient to complete the call.
ENOTCONNECTION	8042H	The call obtained a token for an object that should have been a connection to :CI: or :CO:, but was not a file connection.
E$PARAM	8004H	The call attempted to write beyond the end of a physical file.
E$SPACE	0029H	One of the following is true: • The output volume is full. • The call attempted to write beyond the end of a physical file.
E$STREAM$SPECIAL	003CH	When attempting to read or write to :CI: or :CO:, the Extended I/O System issued an invalid stream file request.
E$SUPPORT	0023H	The connection to :CI: or :CO: was not created by this job.
E$TIME	0001H	The calling task's job was not created by the Human Interface.

C$SEND$EO$RESPONSE, a message processing call, sends a message to and reads a response from the operator's terminal.

```
CALL RQ$C$SEND$EO$RESPONSE(response$p, response$max, message$p,
                           except$ptr);
```

Input Parameters

response$max	A WORD that specifies the maximum length in bytes of the string pointed to by the response$p parameter. The value in response$max must equal the length of the string plus one (stringlength + 1). If response$max is zero or one, no response from the operator's terminal will be requested; control returns to the calling task immediately.
message$p	A POINTER to a buffer containing the message to be sent to the operator's terminal. If NIL, no message is sent.

Output Parameters

response$p	A POINTER to a STRING that receives the operator's response from the terminal.
except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

When used with all its features, C$SEND$EO$RESPONSE sends the string pointed to by message$p to the operator's terminal and waits for a response from the operator. It places this response in the string pointed to by response$p. However, if message$p is NIL, C$SEND$EO$RESPONSE omits sending the message to the operator; if either response$max is zero or response$p is NIL, it does not wait for a response. Therefore, the operations performed by C$SEND$EO$RESPONSE depend on the values of the message$p and response$max parameters, as follows:

message$p	response$max	Action
NIL	zero	Perform no I/O
NIL	non-zero	Send no message, wait for input
NOT NIL	non-zero	Send message, wait for input
NOT NIL	zero	Send message, don't wait

If C$SEND$EO$RESPONSE requests a response from the terminal, no other output can be displayed at the terminal until C$SEND$EO$RESPONSE receives a line terminator from the operator. However, the operator can choose to ignore the displayed message by entering a line terminator only.

The main distinction between the C$SEND$CO$RESPONSE and C$SENDEORESPONSE calls is that C$SEND$EO$RESPONSE always sends messages to and receives messages from the operator's terminal; input and output cannot be redirected to another device. In contrast, C$SEND$CO$RESPONSE sends messages to :CO: and receives messages from :CI:; therefore, programs such as SUBMIT can redirect this input and output.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$CONN$OPEN	0035H	At least one of the following is true: • Either, the connection to the operator's terminal was not open for reading or it was not open for writing. • The connection to the operator's terminal was not open. • The connection to the operator's terminal was opened with A$OPEN rather than S$OPEN.
E$CONTEXT	0005H	The calling task's job was not created by the Human Interface.
E$ERROR$OUTPUT	8085H	The call to SENDEORESPONSE was attempted through an invalid method.
E$EXIST	0006H	The token values for the operator's terminal are not for existing objects.
E$FLUSHING	002CH	The operator's terminal was being detached.
EIONOT$READY	0053H	While attempting to access the terminal, this call found that the device was off-line. Operator intervention is required.

E$LIMIT	0004H	At least one of the following is true: • The calling task's job has already reached its object limit. • The calling task's job or the job's default user object is already involved in 255 (decimal) I/O operations. • The calling task's job was not created by the Human Interface.
E$MEM	0002H	The memory pool of the calling task's job does not currently have a block of memory large enough to allow this system call to run to completion.
ENOTCONNECTION	8042H	The call obtained a token for an object that should have been a connection to the operator's terminal, but was not a file connection.
E$PARAM	8004H	The call attempted to write beyond the end of a physical file.
E$STREAM$SPECIAL	003CH	When attempting to read or write to the operator's terminal, the Extended I/O System issued an invalid stream file request.
E$SUPPORT	0023H	The connection to the terminal was not created by this job.
E$TIME	0001H	The calling task's job was not created by the Human Interface.

CSETCONTROL$C, a program control call, changes a calling task's CONTROL-C exchange to the semaphore specified by the first parameter in the CSETCONTROL$C call.

```
CALL RQ$C$SET$CONTROL$C(control$c$semaphore, except$ptr);
```

Input Parameter

control$c$semaphore — A TOKEN for a user-created semaphore that will receive units when a CONTROL-C is typed on the console keyboard.

NOTE

When a C$SEND$COMMAND call is made, the Human Interface sets the CONTROL-C semaphore to the default Human Interface CONTROL-C handler. If you previously set the CONTROL-C handler, it must be set again after making this call. For more information see the *iRMX® Human Interface User's Guide*.

Output Parameter

except$ptr — A POINTER to a WORD in which the Human Interface returns a condition code.

Description

This call lets you change the default response to a CONTROL-C entry to a response that meets the needs of your task. (The Human Interface's default CONTROL-C action is to delete the acting job--for example, any Human Interface command.)

One unit will be sent to the semaphore each time a CONTROL-C is typed. Any units sent to the semaphore that exceed the maximum number specified during system configuration will be ignored.

A job running in background mode cannot set CONTROL-C.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$CONTEXT	0005H	The calling task's job was not an I/O job. (Refer to the *iRMX® Extended I/O System User's Guide* for information about I/O jobs.)
E$LIMIT	0004H	At least one of the following is true: • The calling task's job has already reached its limit. • The calling task's job was not created by the Human Interface. • The calling task's job or the job's default user object is already involved in 255 (decimal) I/O operations.
E$TYPE	8002H	The TOKEN given in the parameter control$c$semaphore is not a TOKEN for a semaphore.

CSETPARSE$BUFFER

CSETPARSE$BUFFER, a command parsing call, permits parsing the contents of a buffer other than the command line buffer whenever the parsing system calls are used.

```
offset = RQ$C$SET$PARSE$BUFFER(buff$p, buff$max, except$ptr);
```

Input Parameters

buff$p	A POINTER to a buffer containing a STRING containing the text to be parsed. If the buff$p is NIL, the buffer used for parsing reverts to the command line buffer and the buff$max parameter is ignored.
buff$max	A WORD that specifies the length in bytes of the STRING pointed to by the buff$p parameter.

Output Parameters

offset	A WORD in which the Human Interface places the byte offset from the start of the parsing buffer of the last byte parsed in the previous parsing buffer.
except$ptr	A POINTER to a WORD in which the Human Interface returns a condition code.

Description

CSETPARSE$BUFFER allows you to parse buffers other than the command line. You can change buffers at will; you can also revert to the command line parsing buffer by calling CSETPARSE$BUFFER with buff$p=NIL. However, only one parsing buffer per job can be active at any given time.

When called, CSETPARSE$BUFFER sets the parsing pointer to the beginning of the specified buffer. However, it also returns a value (in the offset parameter) that identifies the last byte parsed in the previous parsing buffer. This gives you the ability, when switching back to the previous buffer, of positioning the parsing pointer to its previous position with successive calls to CGETCHAR.

Note that CSETPARSE$BUFFER does not affect the buffer from which CGETINPUT$PATHNAME and C$GET$OUTPUT$PATHNAME retrieve pathnames. These system calls always obtain their pathnames from the command line.

Exception Codes

E$OK	0000H	No exceptional conditions were encountered.
E$CONTEXT	0005H	The calling task's job was not created by the Human Interface. (Refer to the *iRMX® Extended I/O System User's Guide* for information.)
E$LIMIT	0004H	At least one of the following is true: • The calling task's job has already reached its object limit. • The calling task's job was not created by the Human Interface.
E$MEM	0002H	The memory available to the calling task's job is not sufficient to complete the call.

INDEX

C

C$BACKUP$CHAR 5
C$CREATE$COMMAND$CONNECTION 6
C$DELETE$COMMAND$CONNECTION 10
C$FORMAT$EXCEPTION 11
  exception code format 11
CGETCHAR 13
CGETCOMMAND$NAME 15
CGETINPUT$CONNECTION 17
  errors returned to :CO: 17
CGETINPUT$PATHNAME 23
CGETOUTPUT$CONNECTION 29
  errors returned to :CO: 30
CGETOUTPUT$PATHNAME 36
CGETPARAMETER 39
C$SEND$CO$RESPONSE 50
C$SEND$COMMAND 43
C$SEND$EO$RESPONSE 53
CSETCONTROL$C 56
CSETPARSE$BUFFER 58
Command connection 6
  deleting 10
Command pathname 15
CONTROL-C
  default handler 56
  semaphore 56

D

Default message
  creating 11
Deleting a command connection 10

E

E$LIST
  improper value examples 41
EIOS connection 17, 29
Exception code
  default message 11
  format 11

I

Invalid command separators 42
Invoking a command 43
Invoking commands programmatically 6

M

Message
  reading from :CI: 50
  sending to :CO: 50
  reading from operator's terminal 53
  sending to operator's terminal 53

P

Parsing buffer
  changing 58
  getting a character 13
  getting a parameter 39
  input pathnames 23
  output pathname 36
  pointer 5
Preposition parameter values
  CGETOUTPUT$CONNECTION 29
  CGETOUTPUT$PATHNAME 36

S

System call dictionary 3

intel®

REQUEST FOR READER'S COMMENTS

Intel's Technical Publications Departments attempt to provide publications that meet the needs of a Intel product users. This form lets you participate directly in the publication process. Your comment will help us correct and improve our publications. Please take a few minutes to respond.

Please restrict your comments to the usability, accuracy, organization, and completeness of thi publication. If you have any comments on the product that this publication describes, please contac your Intel representative.

1. Please describe any errors you found in this publication (include page number).

2. Does this publication cover the information you expected or required? Please make suggestion for improvement.

3. Is this the right type of publication for your needs? Is it at the right level? What other types o publications are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this publication on a scale of 1 to 5 (5 being the best rating). _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____ PHONE ( )

CITY _____ STATE _____ ZIP CODE _____

(COUNTRY)

Please check here if you require a written reply. ☐

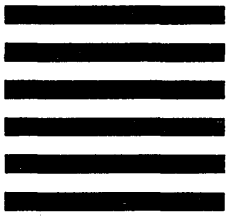
E'D LIKE YOUR COMMENTS . . .

is document is one of a series describing Intel products. Your comments on the back of this form will lp us produce better manuals. Each reply will be carefully reviewed by the responsible person. All mments and suggestions become the property of Intel Corporation.

'ou are in the United States, use the preprinted address provided on this form to return your mments. No postage is required. If you are not in the United States, return your comments to the Intel les office in your country. For your convenience, international sales office addresses are printed on ; last page of this document.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 79 HILLSBORO, OR



POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
OMSO Technical Publications, MS: HF3-72
5200 N.E. Elam Young Parkway
Hillsboro, OR 97124-9978

INTERNATIONAL SALES OFFICES

INTEL CORPORATION
3065 Bowers Avenue
Santa Clara, California 95051

BELGIUM
Intel Corporation SA
Rue des Cottages 65
B-1180 Brussels

DENMARK
Intel Denmark A/S
Glentevej 61-3rd Floor
dk-2400 Copenhagen

ENGLAND
Intel Corporation (U.K.) LTD.
Piper's Way
Swindon, Wiltshire SN3 1RJ

FINLAND
Intel Finland OY
Ruosilante 2
00390 Helsinki

FRANCE
Intel Paris
1 Rue Edison-BP 303
78054 St.-Quentin-en-Yvelines Cedex

ISRAEL
Intel Semiconductors LTD.
Atidim Industrial Park
Neve Sharet
P.O. Box 43202
Tel-Aviv 61430

ITALY
Intel Corporation S.P.A.
Milandfiori, Palazzo E/4
20090 Assago (Milano)

JAPAN
Intel Japan K.K.
Flower-Hill Shin-machi
1-23-9, Shinmachi
Setagaya-ku, Tokyo 15

NETHERLANDS
Intel Semiconductor (Netherland B.V.)
Alexanderpoort Building
Marten Meesweg 93
3068 Rotterdam

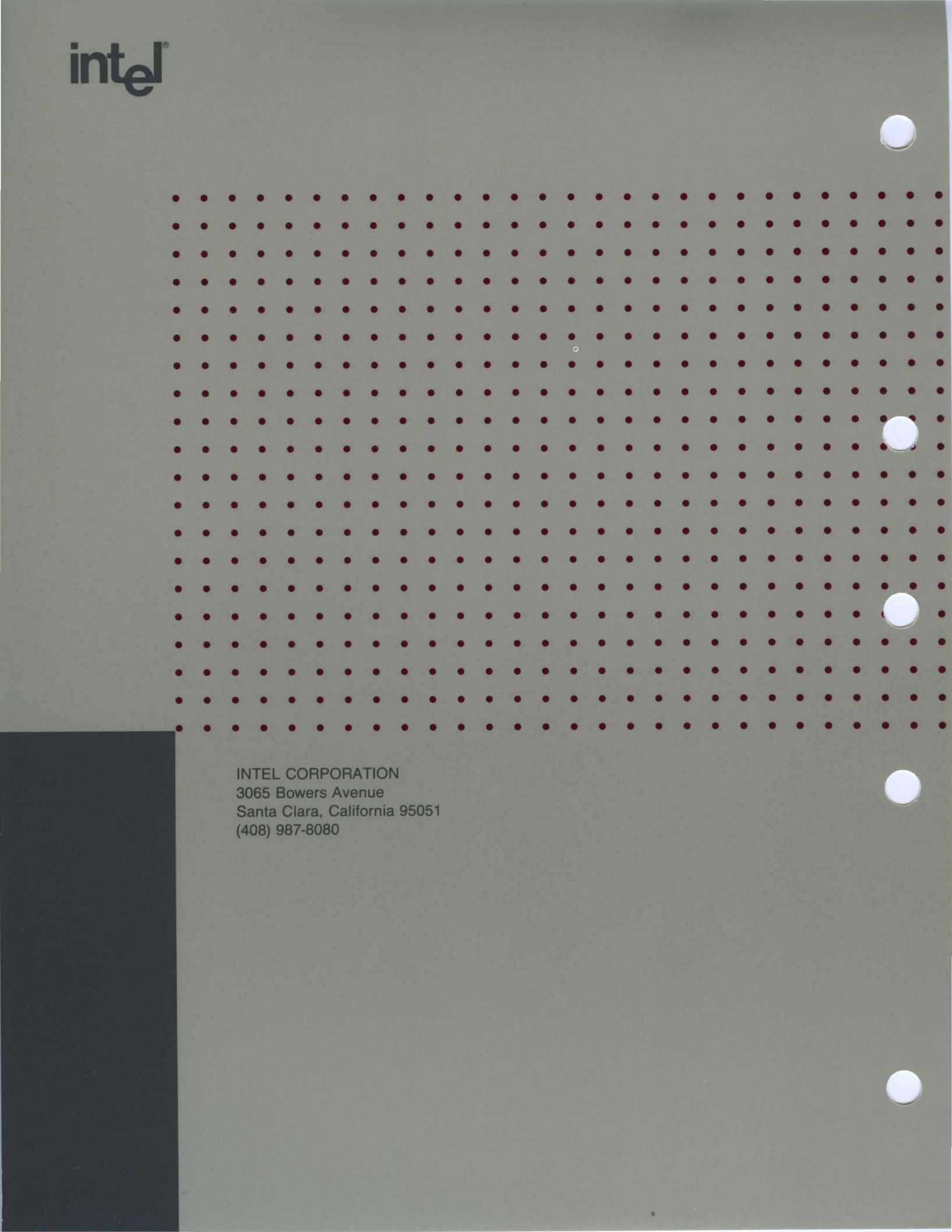
NORWAY
Intel Norway A/S
P.O. Box 92
Hvamveien 4
N-2013, Skjetten

SPAIN
Intel Iberia
Calle Zurbaran 28-IZQDA
28010 Madrid

SWEDEN
Intel Sweden A.B.
Dalvaegen 24
S-171 36 Solna

SWITZERLAND
Intel Semiconductor A.G.
Talackerstrasse 17
8125 Glattbrugg
CH-8065 Zurich

WEST GERMANY
Intel Semiconductor G.N.B.H.
Seidlestrasse 27
D-8000 Munchen



INTEL CORPORATION
3065 Bowers Avenue
Santa Clara, California 95051
(408) 987-8080