intel

iRMX® I Terminal Handler Reference Manual

Order Number: 462930-001

intel

iRMX® I
Terminal Handler
Reference Manual

Order Number: 462930-001

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 Terminal Handler, a subsystem of the iRMX® I Operating System that supports the use of a terminal as an I/O device for an application system. It describes the effects of certain special keyboard characters as well as how to use the Terminal Handler for task input from, and output to, a terminal.

READER LEVEL

This manual is intended for both application and system programmers who are familiar with the concepts and terminology introduced in the *iRMX® I Nucleus User's Guide.*

CONTENTS

Chapter 1. Overview

Chapter 2. Using a Terminal with the iRMX® I Operating System

Chapter 3. Programming Considerations

Chapter 4. Configuration of the Terminal Handler

Index

Figures

Tables

1 OVERVIEW

1.1 INTRODUCTION

The Terminal Handler supports real-time, asynchronous I/O between a terminal and tasks running under the iRMX® I Nucleus. It can be used for applications that require only limited I/O through a terminal and is generally used in applications with no iRMX I Basic I/O System. The Terminal Handler features include the following:

- Line editing capabilities.
- Keystroke control over output, including output suspension and resumption, and deletion of data being sent by tasks to the Terminal Handler.
- Echoing of characters as they are entered into the Terminal Handler's line buffer.

An output-only version of the Terminal Handler is available for use in applications in which tasks send output to a terminal but do not receive input from it.

NOTE

The Terminal Handler supports character-by-character input from a terminal, rather than computer-to-computer input.

1.2 ORGANIZATION OF THIS MANUAL

This manual consists of four chapters:

- Chapter 1 -- Overview

  This chapter briefly describes the Terminal Handler and introduces some of its features.

- Chapter 2 -- Using a Terminal with the iRMX I Operating System

  This chapter provides information needed to use a terminal with the iRMX I Operating System.

- Chapter 3 -- Programming Considerations

  This chapter contains the information that a programmer needs to write tasks that send data to, or receive data from, the terminal.

- Chapter 4 -- Configuration of the Terminal Handler

  This chapter identifies and describes the configurable features, characteristics, and identifiers of the Terminal Handler.

USING A TERMINAL WITH THE iRMX® I OPERATING SYSTEM 2

2.1 INTRODUCTION

When using a terminal with the iRMX I Operating System, you must limit the maximum priority of tasks so they do not interfere with the proper functioning of the terminal. High-priority, processor-bound tasks can cause the Terminal Handler to drop input characters.

While using a terminal under control of the Terminal Handler, you either read an output message from the terminal's display or enter characters on the terminal's keyboard. Normal input characters are those destined for input messages sent to tasks. Special input characters direct the Terminal Handler to take special actions. The special characters are RUBOUT, CARRIAGE RETURN, LINE FEED, ESCAPE, CONTROL-C, CONTROL-O, CONTROL-Q, CONTROL-R, CONTROL-S, CONTROL-X, and CONTROL-Z. (These special characters are described in detail later in this chapter.) The output-only version of the Terminal Handler does not support any of the special characters. The remainder of this chapter discusses handling these two types of characters.

NOTE

This chapter contains several references to mailboxes and request messages used by tasks to communicate with the terminal. Chapter 3 contains more information about these.

2.2 HOW NORMAL CHARACTERS ARE HANDLED

The destination of a normal character, when entered, depends on whether there is an input request message at the Terminal Handler's input request mailbox. If there is an input request message, the character echoes at the display and then goes into the input request message. If there is no input request message, the character is deleted.

2.3 HOW SPECIAL CHARACTERS ARE HANDLED

Most special characters are for line-editing. Table 2-1 lists the special characters and their functions. Text following the table describes the functions in more detail. Each description is divided into two parts: internal effects and external effects. Internal effects are not visible; external effects are shown on the display. Note that in these descriptions, "the current line" is the edited data, entered since the last end-of-file character.

Table 2-1. Special Character Summary

Special Character	Function
RUBOUT	Deletes previously typed character
CONTROL-R	Displays current line with editing
CONTROL-X	Deletes the current line
CONTROL-Z	Sends an empty message
CARRIAGE RETURN	Signals end of line
LINE FEED	Signals end of line
ESCAPE	Signals end of line
CONTROL-S	Suspends output
CONTROL-Q	Resumes suspended output
CONTROL-O	Kills or restarts output
CONTROL-C	Calls an application program

2.3.1 RUBOUT -- Deletes a Previously Typed Character

Internal Effects: Deletes the most recently entered (but not yet deleted) character from the current line. If the current line is empty, there is no internal effect.

External Effects: If the current line is empty, the BEL character (07H) is sent to the terminal. Otherwise, the character is "rubbed out" in one of two rubout modes (see Chapter 4).

2.3.2 CONTROL-R -- Displays the Current Line

Internal Effects: None.

External Effects: Sends a CARRIAGE RETURN and LINE FEED to the terminal, followed by the current line. If the current line is empty, the previous line is sent to the display, where it can be edited and submitted as a new input message.

2.3.3 CONTROL-X -- Deletes the Current Line

Internal Effects: Empties the current line.

External Effects: Sends the sequence (#, CARRIAGE RETURN, LINE FEED) to the terminal.

2.3.4 CONTROL-Z -- Sends an Empty Message

Internal Effects: Puts a zero in the ACTUAL field of the input request message currently being processed. The message is then sent to the appropriate response mailbox.

External Effects: None.

2.3.5 CARRIAGE RETURN, LINE FEED, or ESCAPE -- Signals the End of a Line

Internal Effects: Puts either the ASCII end-of-transmission character (0AH for CARRIAGE RETURN or LINE FEED) or the ESCAPE character (1BH) in the current line. Each of these characters signals the end of a message, so the input request message currently being constructed is sent to the appropriate response mailbox.

External Effects: If the end-of-line indicator is either CARRIAGE RETURN or LINE FEED, both CARRIAGE RETURN and LINE FEED are sent to the terminal. If the indicator is ESCAPE, there is no effect on the display.

2.4 OUTPUT CONTROL

Output request messages sent to output mailboxes can be processed in one of three modes:

- Normal Mode -- Messages are outputted as described in Chapter 3 (this is the default mode).
- Queueing Mode -- Messages are queued at the output mailbox where they remain until the terminal operator permits processing of the messages.
- Suppression Mode -- Messages are discarded.

2.4.1 CONTROL-Q -- Resuming Suspended Output (Normal Mode)

Negates a CONTROL-S output request sent to the output mailbox. The suppressed output is displayed in the order it had before the CONTROL-S had been selected. If this output is too fast, you can stop it with another CONTROL-S.

2.4.2 CONTROL-S -- Suspends Output (Queueing Mode)

Puts output in the queueing mode.

2.4.3 CONTROL-O -- Kills or Restarts Output (Suppression Mode)

If output is in the normal mode, CONTROL-O puts it in the suppression mode. If output is in the suppression mode, CONTROL-O restores it to the normal mode. If output is in the queueing mode, CONTROL-O has no effect. Internally, the request messages that tasks send while output is being suppressed are returned to those tasks just as if the output had not been suppressed.

2.5 PROGRAM CONTROL

The remaining control character affects system behavior.

2.5.1 CONTROL-C -- Calls an Application Program

CONTROL-C invokes a user-written procedure with no parameters named RQ$ABORT$AP. This procedure, which must be compiled under the COMPACT and ROM controls, can perform any actions that suit the application. Often, as its name suggests, RQ$ABORT$AP aborts an application. If it is written by the user (and it need not be), RQ$ABORT$AP is not required to have a RETURN statement.

CONTROL-C is the same as CONTROL-Z; that is, it returns the current input request message with its ACTUAL field set to zero, even if the application system does not contain an RQ$ABORT$AP procedure.

PROGRAMMING CONSIDERATIONS 3

3.1 INTRODUCTION

The iRMX I Terminal Handler supports terminal input and output by providing mailbox interfaces. Figure 3-1 shows the use of these mailboxes. In the figure, an arrow pointing from a task to a mailbox represents an RQ$SEND$MESSAGE system call. An arrow pointing from a mailbox to a task indicates an RQ$RECEIVE$MESSAGE system call.

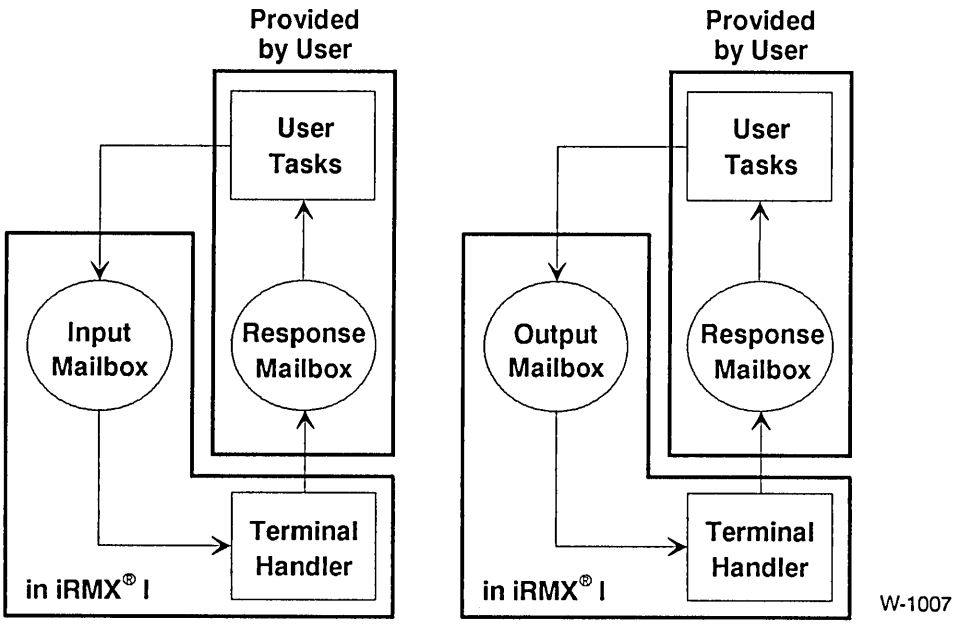


Figure 3-1. Input and Output Mailbox Interfaces

The protocol that tasks observe is much the same for input and output. In each case, the task initiates I/O by sending a request message to a mailbox. An input request mailbox (default name RQTHNORMIN) and an output request mailbox (default name RQTHNORMOUT) are provided. These mailboxes are cataloged in the root job directory.

For multiple terminals, one input and one output mailbox is cataloged for each Terminal Handler. (See Chapter 4 for more information about multiple versions of the Terminal Handler.) Figure 3-2 illustrates the protocol for finding the root job token and for obtaining the input and output mailbox tokens.

```
/*************************************************************************
 *  This example illustrates the protocol for finding the root job token *
 *  and for obtaining the input and output mailbox tokens.               *
 *************************************************************************/

   DECLARE rtjb$token                  WORD;
   DECLARE root$job                    LITERALLY '3';
   DECLARE status                      WORD;
   DECLARE input$mbx$token             WORD;
   DECLARE wait$forever                LITERALLY '0FFFFH';


 /*By setting the input parameter to three, the GET$TASK$TOKEN system call
   will return the root job's TOKEN.*/

   rtjb$token = RQ$GET$TASK$TOKENS     (root$job,
                                       @status);

 /*The following LOOKUP$OBJECT system calls use the default mailbox names.*/

   input$mbx$token = RQ$LOOKUP$OBJECT  (rtjb$token,
                                       @(10, 'RQTHNORMIN'),
                                       wait$forever,
                                       @status);

   output$mbx$token = RQ$LOOKUP$OBJECT (rtjb$token,
                                       @(11, 'RQTHNORMOUT'),
                                       wait$forever,
                                       @status);
```

Figure 3-2. Protocol for Obtaining Root Job and Mailbox Tokens

Refer to the *iRMX® I Nucleus User's Guide* for more information about the individual system calls used in the previous example.

When a task sends a message to the Terminal Handler mailbox, the Terminal Handler processes the request and then sends a response message back to the requesting task. The task waits at a response mailbox for the message. Thus, whether a task does input or output, it first sends and then receives. The full details of the input and output protocols are described later in this chapter.

For both input and output, a task sends a message segment to the Terminal Handler. The format of a request message is depicted in Figure 3-3. The numbers in that figure are offsets, in bytes, from the beginning of the segment. The field names have different meanings for input and for output. For both input and output, the first four fields are WORD values. The MESSAGE CONTENT field can be up to 132 bytes long for input and up to 65,527 bytes long for output.

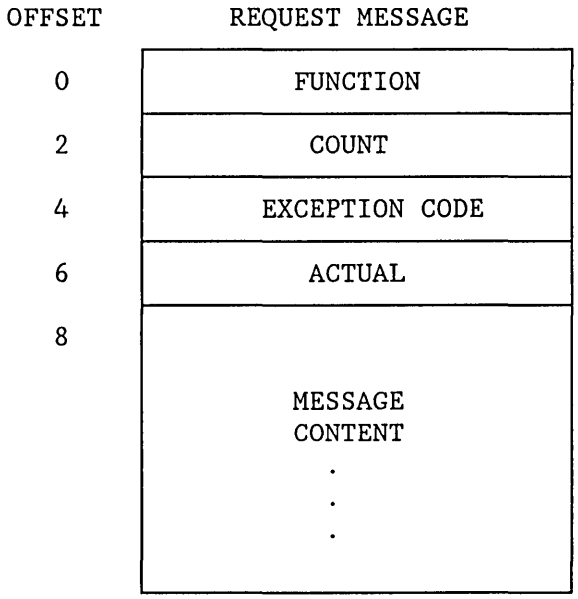


Figure 3-3. Request Message Format

NOTE

In the following discussions, the names F$WRITE and F$READ are literal names for the particular WORD values 5 and 1, respectively.

3.2 OUTPUT

When transmitting output, a task first prepares an output request message.

The task fills in the following fields before sending the message:

FUNCTION	F$WRITE
COUNT	The number of bytes (not to exceed 65527) in the MESSAGE CONTENT field
MESSAGE CONTENT	The bytes to be output

After preparing the message segment, the task sends it to the output request mailbox. Messages sent to this mailbox are processed in a first-in-first-out manner. Processing a message involves sending the characters in the MESSAGE CONTENT field to the terminal until a total of COUNT characters have been sent. There is one exception: when the Terminal Handler encounters the end-of-transmission character (0AH), it sends a CARRIAGE RETURN and a LINE FEED to the terminal.

When sending the output request message, the task specifies a user-supplied response mailbox. If no response mailbox is specified, the Terminal Handler deletes the segment that contained the message. Note though, that if the system call DISABLE$DELETION was used to make the segment containing the output message immune to deletion, the Nucleus will put the Terminal Handler into the asleep state because the Terminal Handler cannot execute deletion of the segment. This situation eliminates the Terminal Handler as a functioning task.

In addition to transmitting the message to the terminal, the Terminal Handler fills in the remaining fields in the output request message. The requesting task can wait indefinitely at the response mailbox (that is, it can call the RQ$RECEIVE$MESSAGE system call with a time limit of 0FFFFH) immediately after sending the output request. By observing this protocol, the task can learn of the success or failure of the output attempt by reading these fields:

- EXCEPTION CODE -- the encoded result of the output operation:
  - E$OK -- the operation was successful.
  - E$PARAM -- the FUNCTION field in the message did not contain F$WRITE.
  - E$BOUNDS -- the COUNT field in the message is too big for the segment; that is, COUNT + 8 is greater than the length of the segment containing the message.

- ACTUAL -- the actual number of bytes output.

In summary, the output protocol is as follows:

- Prepare the output request message segment, filling in the FUNCTION, COUNT, and MESSAGE CONTENT fields.
- Send the segment, via the RQ$SEND$MESSAGE system call, to the output request mailbox. You should specify a response mailbox in the system call.
- Wait indefinitely, via the RQ$RECEIVE$MESSAGE system call, at the response mailbox. When received, the message contains the results of the transmission in the EXCEPTION CODE and ACTUAL fields.

3.3 INPUT

The protocol for obtaining input is similar to that for output. A message is prepared and sent to a request mailbox; then, after the data has been input, the message is received at a response mailbox. There is a significant difference, however, between input and output protocols. Because the input is contained in the message segment at the response mailbox, you must designate a response mailbox and then wait there.

CAUTION

When multiple tasks use the same mailbox for input from the terminal, a task may get input intended for another task.

A task needing input first prepares an input request message. It must fill in the FUNCTION and COUNT fields before sending its request. The FUNCTION field must contain F$READ. The COUNT field reflects the maximum possible number of input characters in the input message. The value of COUNT must not exceed 132; moreover, COUNT + 8 must not exceed the length of the input request message segment.

When sending the input request message, the task must specify the response mailbox in its call to RQ$SEND$MESSAGE. The Terminal Handler obtains characters from the terminal and places them in the MESSAGE CONTENT field. The message is terminated by an end-of-line character (CARRIAGE RETURN, LINE FEED, or ESCAPE). The lone exception is when the end-of-line character has been "normalized" by being preceded by a CONTROL-P then the end-of-line character is treated as a normal character.

NOTE

If more than COUNT characters are entered before the end-of-line character, the extra characters are ignored and CONTROL-G is activated. This character usually causes the terminal to emit a beep tone.

After the message is complete, the Terminal Handler fills in the EXCEPTION CODE and ACTUAL fields as follows:

- EXCEPTION CODE -- the encoded result of the input operation:
  - E$OK --- the operation was successful.
  - E$PARAM -- either the FUNCTION field in the message did not contain F$READ or the COUNT field was greater than 132.
  - E$BOUNDS -- COUNT + 8 is greater than the length of the message segment.
- ACTUAL -- the number of bytes actually entered and placed in the MESSAGE CONTENT field.

The requesting task must wait indefinitely (that is, it must make a RQ$RECEIVE$MESSAGE system call with a time limit of 0FFFFH) at the designated response mailbox immediately after sending the input request.

In summary, the input protocol is as follows:

- Prepare the input request message segment, filling in the FUNCTION and COUNT fields.
- Send the segment, via the RQ$SEND$MESSAGE system call, to the input request mailbox. In the call, specify a response mailbox.
- Wait indefinitely, via the RQ$RECEIVE$MESSAGE system call, at the response mailbox. When received, the message segment will contain the results of the input operation in the MESSAGE CONTENT, EXCEPTION CODE, and ACTUAL fields.

CONFIGURATION OF THE TERMINAL HANDLER 4

4.1 INTRODUCTION

The Terminal Handler is a configurable layer of the iRMX I Operating System. It contains several options that you can adjust to meet your specific needs. To make configuration choices, Intel provides three kinds of information:

- A list of configurable options
- Detailed information about the options
- Procedures enabling you to specify your choices

The rest of this chapter provides the first category of information. To obtain the second and third categories, refer to the *iRMX® I Interactive Configuration Utility Reference Manual* and the *Guide to the iRMX® I Interactive Configuration Utility.*

4.2 CONFIGURABLE OPTIONS

Some Terminal Handler features, characteristics, and identifiers are configurable. Configurability is important for applications with unusual characteristics, such as a component hardware environment or multiple terminal handlers. The following sections describe the configurable options available on the Terminal Handler.

4.2.1 Selecting a Version of the Terminal Handler

The iRMX I Terminal Handler is available in two versions:

- Input and output
- Output only

The input-and-output version enables you to enter characters at the terminal as well as receive data. The output-only version is useful in applications in which tasks send output to a terminal but do not receive input from the terminal.

4.2.2 Baud Rate

The Terminal Handler supports the following baud rates:

- 110
- 150
- 300
- 600
- 1200
- 2400
- 4800
- 9600 (default)
- 19200

4.2.3 Rubout

You can delete a character from the buffer in one of two ways:

- Echo mode (default)
- Replace mode

In the echo mode, the character being deleted from the current line is re-echoed to the display. For example, entering "CAT" and then pressing RUBOUT three times results in the display "CATTAC".

In the replace mode, the deleted character is replaced on the display with the blanking character. For example, entering "CAT" and then pressing RUBOUT three times deletes "CAT" from the display.

The blanking-character and the default RUBOUT mode can be specified when you generate your system. If they are not specified, they default to a blank (20H) and mode (REPLACE).

4.2.4 USART

The USART (Universal Synchronous/Asynchronous Receiver/Transmitter) is a device that, depending on the application, can be used either to convert serial data to parallel data or to convert parallel data to serial data. The Terminal Handler requires an 8251A USART as a terminal controller. You can specify:

- The port address of the USART (default value is 0D8H).
- The interval between the port addresses for the USART.
- The number of bits of valid data per character that can be sent from the USART (default value is 7).

4.2.5 Mailbox Names

You can change the default names of both the input mailbox (RQTHNORMIN) and the output mailbox (RQTHNORMOUT). The new names must not be over 12 alphanumeric characters long. If you wish to run multiple terminals, each input mailbox must have a different name.

4.2.6 Interrupt Levels

You can specify the interrupt levels used by the Terminal Handler for input and output by selecting a value that corresponds to a particular interrupt value. The default value for the input interrupt level is 68H, and the default value for the output interrupt level is 78H. The Terminal Handler assumes the interrupt controller is either the 8259A or the 80130.

4.3 CREATING MULTIPLE VERSIONS OF THE TERMINAL HANDLER

Your iRMX I system can contain multiple versions of the Terminal Handler. For example, you may have two tasks using the Terminal Handler and want to communicate with these tasks from separate terminals. When creating multiple versions of the Terminal Handler follow these rules:

- Each Terminal Handler must use different input and output mailbox names.
- Each Terminal Handler must use a unique USART.
- Each Terminal Handler must use different interrupt levels.
- The code for the Terminal Handlers must be located in different, nonoverlapping areas; each Terminal Handler must have its own data area.
- Each Terminal Handler must have a separate job.

Refer to the *iRMX® I Interactive Configuration Utility Reference Manual* for detailed information about these rules.

INDEX

B

C

E

I

K

L

M

O

P

Q

R

S

T

U

V

intel®

iRMX® I Terminal Handler
Reference Manual
462930-001

REQUEST FOR READER'S COMMENTS

Intel's Technical Publications Departments attempt to provide publications that meet the needs of all Intel product users. This form lets you participate directly in the publication process. Your comments 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 this publication. If you have any comments on the product that this publication describes, please contact 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 suggestions for improvement.

3. Is this the right type of publication for your needs? Is it at the right level? What other types of 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)

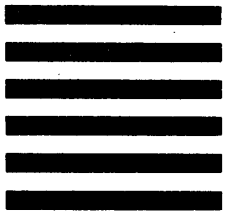
Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS . . .

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

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

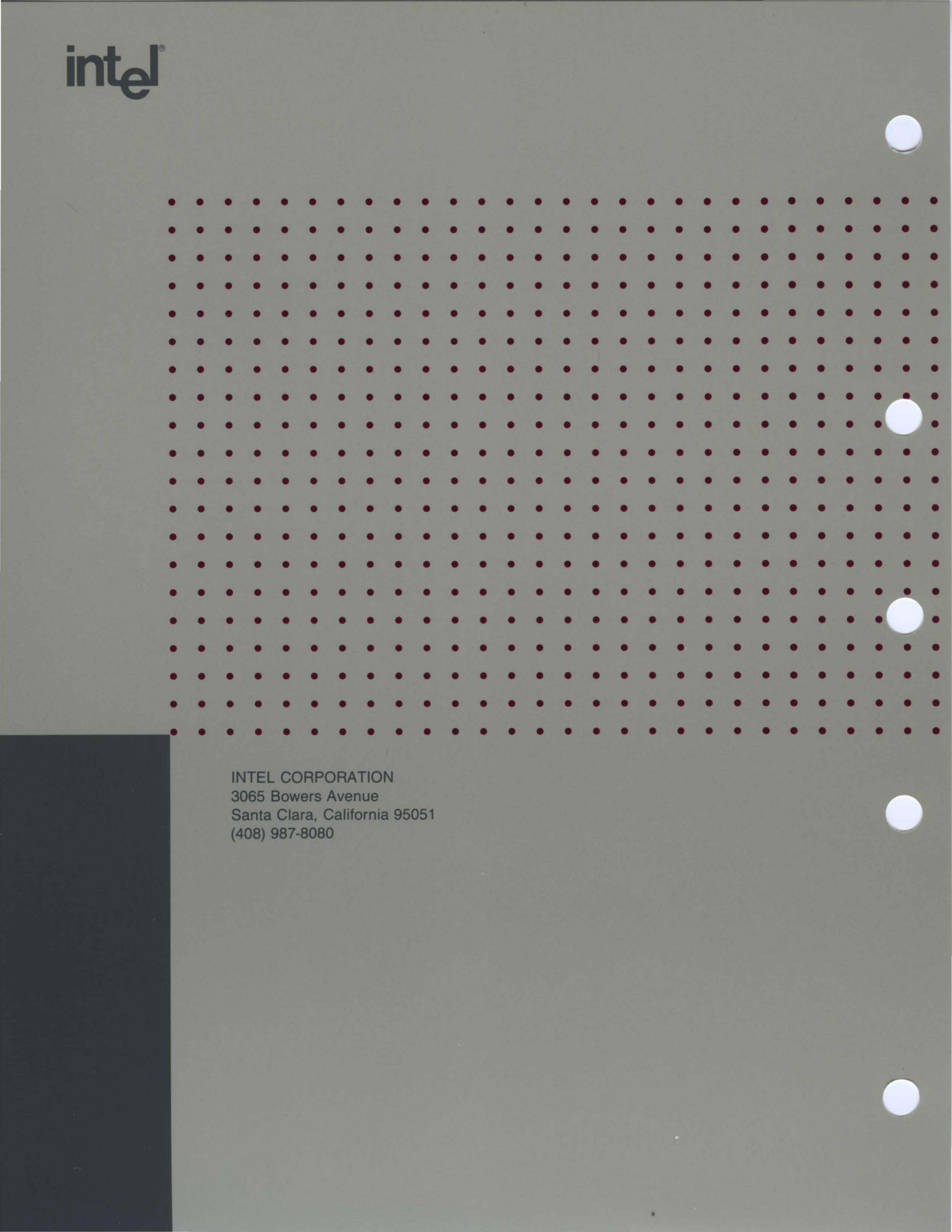
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



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



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