iRMX™ 86 TERMINAL HANDLER REFERENCE MANUAL

CONTENTS

***

CHAPTER 1 OVERVIEW OF THE TERMINAL HANDLER

The Terminal Handler supports real-time, asynchronous I/O between an operator's terminal and tasks running under the iRMX 86 Nucleus. It is intended for use in applications which require only limited I/O through a terminal, and it generally is used in applications that do not include the iRMX 86 I/O System. The features of the Terminal Handler 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.
- 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 the terminal.

NOTE

The terminal handler is intended primarily to support character-by-character input from a terminal, rather than computer-to-computer input.

ORGANIZATION OF THIS MANUAL

This manual consists of four chapters:

- Chapter 1 --Overview of the Terminal Handler

  This chapter discusses the purpose of the Terminal Handler and introduces some of the features.

- Chapter 2 -- Using a Terminal with the iRMX 86 Operating System

  This chapter provides information that the operator needs in order to use a terminal with the iRMX 86 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

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

***

CHAPTER 2
USING A TERMINAL WITH THE iRMX™ 86 OPERATING SYSTEM

When you are using a terminal with the iRMX 86 Operating System, you must limit the maximum priority of your tasks or they could interfere with the proper functioning of your terminal. High priority processor-bound tasks can cause the Terminal Handler to drop input characters.

While using a terminal that is under control of the Terminal Handler, an operator either reads an output message from the terminal's display or enters characters by striking keys on the terminal's keyboard. Normal input characters are those destined for input messages that are 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. The output-only version of the Terminal Handler does not support any of the special characters. In the remainder of this chapter, the handling of these two types is discussed, and the significance of each of the special characters is explained.

NOTE

This chapter contains several references to mailboxes and request messages used by tasks to communicate with the terminal. If you are puzzled by such a reference, look in Chapter 3 for an explanation.

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 is echoed to the terminal's display and goes into the input request message. If there is not an input request message, the character is deleted.

HOW SPECIAL CHARACTERS ARE HANDLED

Table 2-1 lists the special characters and summarizes the effects of each of them. The following text comprises complete descriptions of the effects of the special characters. In these descriptions, there are several references to "the current line." The current line consists of the data, with editing, that has been entered since the last end-of-file character.

Table 2-1. Special Character Summary

Special Character	Effect
RUBOUT	Deletes previously entered character.
Carriage Return	Signals end of line.
Line Feed	Signals end of line.
ESCape	Signals end of line.
control-C	Calls an application program.
control-O	Kills or restarts output.
control-Q	Resumes suspended output.
control-R	Displays current line with editing.
control-S	Suspends output.
control-X	Deletes the current line.
control-Z	Sends empty message.

The following descriptions concern the special characters needed when entering data at the terminal. Most of these characters are for line-editing. Each description is divided into two parts: internal effects and external effects. The difference is that internal effects are those that are not directly visible, whereas external effects are immediately shown on the terminal's display.

RUBBING OUT A PREVIOUSLY-TYPED CHARACTER (RUBOUT)

Internal Effects: Causes the most recently entered but not yet deleted character to be deleted 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 accordance with one of two available rubout modes. See Chapter 4 for a description of rubout modes.

DISPLAYING THE CURRENT LINE (CONTROL-R)

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 line-edited and submitted as a new input message.

DELETING THE CURRENT LINE (CONTROL-X)

Internal Effects: Empties the current line.

External Effects: Causes the sequence (#, Carriage Return, Line Feed) to be sent to the terminal.

SENDING AN EMPTY MESSAGE (CONTROL-Z)

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.

SIGNALLING THE END OF A LINE OF INPUT (CARRIAGE RETURN, LINE FEED, OR ESCAPE)

Internal Effects: Puts either the ASCII end-of-transmission character (OAH in the case of 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, however, there is no effect on the display.

OUTPUT CONTROL

Output request messages that are sent to output mailboxes can be processed in one of three ways:

- They can be outputted as described later in Chapter 3.
- They can be queued at the output mailbox where they remain until an operator at the terminal takes action to permit processing of the messages.
- They can be discarded.

In the descriptions that follow, these methods of dealing with output requests are called the normal mode, the queueing mode, and the suppression mode, respectively. Initially, output is in the normal mode.

SUSPENDING OUTPUT (CONTROL-S)

Puts output in the queueing mode.

RESUMING OUTPUT (CONTROL-Q)

Negates the effects of control-S by allowing the display of output requests that are sent to the output mailbox. The output that has been suppressed is displayed (very quickly) in the order in which it would have been displayed earlier if the control-S had not been pressed. If you are overwhelmed by this output, you can stop it again by pressing control-S.

DELETING OR RESTARTING OUTPUT (CONTROL-O)

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.

PROGRAM CONTROL

The remaining control character affects system behavior.

CALLING A USER-WRITTEN PROCEDURE MANUALLY (CONTROL-C)

Control-C invokes a parameter-less, user-written procedure named RQ$ABORT$AP. This procedure, which must be compiled under the COMPACT control, 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 also causes the effects produced by control-Z; that is, it returns the current input request message with its ACTUAL field set to zero. This is the case even if the application system does not contain an RQ$ABORT$AP procedure.

***

CHAPTER 3
PROGRAMMING CONSIDERATIONS

The iRMX 86 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.

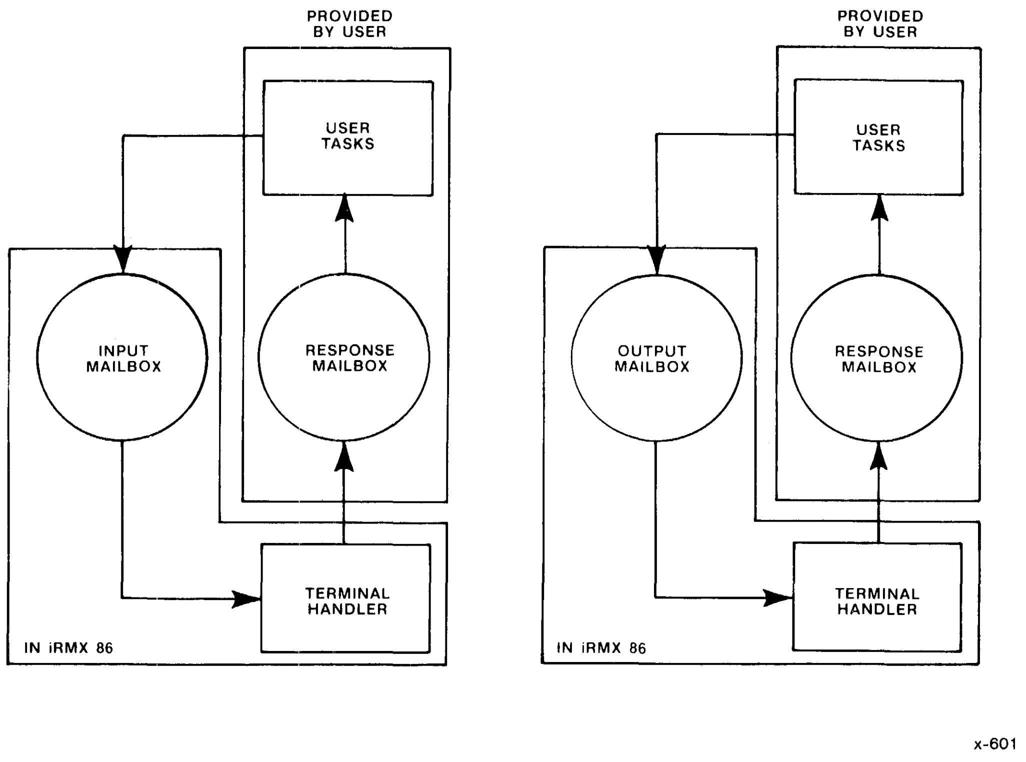

Figure 3-1. Input And Output Mailbox Interfaces

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. In the case of multiple terminals, one input and one output mailbox will be 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 'OFFFFH';


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

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

/*The following LOOKUP$OBJECT primitives 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 86 NUCLEUS REFERENCE MANUAL for more information concerning the individual primitives 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. Output is discussed first because it is somewhat easier to understand.

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 3-2. 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 in length for input and up to 65527 bytes in length for output.

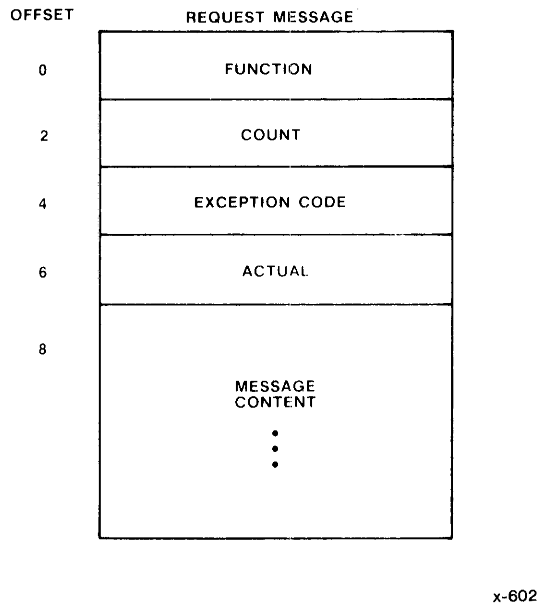

Figure 3-3. Request Message Format

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

OUTPUT

The first thing a task does when transmitting output is prepare an output request message. The task must fill in the following fields prior to sending the message:

FUNCTION --- F$WRITE.

COUNT --- the number of bytes (not to exceed 65527) in the MESSAGE CONTENT field.

MESSAGE CONTENT --- the bytes that are to be output.

Having prepared the message segment, the task must send 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 (OAH), 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 will delete the segment that contained the message. But note, if the the system call DISABLE$DELETION was used by the application engineer to make the segment containing the output message immune to ordinary deletion, the Nucleus will put the Terminal Handler into the asleep state because the Terminal Handler cannot execute deletion of the segment. This situation effectively 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 OFFFFH) immediately after sending the output request. By observing this protocol, the task can learn of the success or failure of the output attempt. The fields that provide this information are the following:

- 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 protocol observed by tasks doing output 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. It is advisable, but not necessary, to 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.

INPUT

The protocol for obtaining input is much the same as that for outputting. 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, it is necessary to designate a response mailbox and then wait there.

CAUTION

When multiple tasks use the same mailbox for input from the terminal, it is possible for a task to get input that is intended for another task.

A task needing input first prepares an input request message. It must fill in the FUNCTION and COUNT fields prior to 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 prior to the end-of-line character, the extra characters are ignored, and the control-G character 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, which is one of the following:

  - 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.

***

CHAPTER 4
CONFIGURATION

The Terminal Handler is a configurable layer of the 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 to allow you to specify your choices.

The balance of this chapter provides the first category of information. To obtain the second and third categories of information, refer to the iRMX 86 CONFIGURATION GUIDE.

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.

SELECTING A VERSION OF THE TERMINAL HANDLER

The iRMX 86 Terminal Handler is available in two different versions:

- Input and Output
- Output-only

The input and output version allows 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.

BAUD RATE

You can set the baud rate for the Terminal Handler to any of the following values:

110
150
300
600
1200
2400
4800
9600
19200

The default baud rate for the Terminal Handler is 9600.

Baud Count

The baud count provides a way to calculate internal timer values given the clock input frequency. If your system's programmable interval timer (PIT) has a clock input frequency other than 1.2288 megahertz, you must set the baud count. The default value for the baud count is 4.

RUBOUT MODE AND BLANKING CHARACTER

As previously mentioned, there are two ways to rubout a character:

- Copying mode
- Blanking mode

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

In the blanking mode, the deleted character is replaced on the CRT screen with the blanking character. For example, entering "CAT" and then striking RUBOUT three times deletes all three characters from the display.

The copy mode is the default mode. The default blanking character for the blanking mode is a space (20H).

USART

The USART is a device that, depending upon 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 a 8251A USART as a terminal controller. You can specify

- The port address of the USART. The default value for the port address 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. The default value for the number of bits is 7.

PIT

The Terminal Handler requires a PIT as an input to the USART. You can specify the following information about the programmable interval timer (PIT):

- The port address of the PIT. The default value for the port address is 0D0H.
- The interval between the port addresses for the PIT.
- The number of the PIT counter connected to the USART clock input. The default value is 2.

MAILBOX NAMES

You can change the default names of both the input mailbox (RQTHNOTMIN) and the output mailbox (RQTHNORMOUT). The new names must not be over 12 alphanumeric characters in length.

INTERRUPT LEVELS

You can specify the interrupt levels used by the Terminal Handler for input and output. You choose interrupt levels 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.

CREATING MULTIPLE VERSIONS OF THE TERMINAL HANDLER

Your iRMX 86 system can contain multiple version of the Terminal Handler. This may be desirable if, for example, you have two tasks that use the Terminal Handler and you want to communicate with these tasks from separate terminals. In order to create multiple versions of the Terminal Handler, you must obey the following 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, non-overlapping areas; each Terminal Handler must have its own data area.
- Each Terminal Handler must have a separate job.

Refer to the iRMX 86 CONFIGURATION GUIDE for detailed information about the previously described rules. If you adhere to these rules, you can create multiple versions of the Terminal Handler in your application system.

***

INDEX

Primary references are underscored.

***