Rocket

Rocket Model 204 Editing Guide

Version 7 Release 5.0

October 2014
204–75–EDIT–01

Notices

Edition

Publication date: October 2014
Book number: 204–75–EDIT–01
Product version: Version 7 Release 5.0

Copyright

Trademarks

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Corporate Information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA

Contacting Technical Support

If you have current support and maintenance agreements with Rocket Software and CCA, contact Rocket Software Technical support by email or by telephone:

Email: m204support@rocketsoftware.com

Telephone :

North America	+1.800.755.4222
United Kingdom/Europe	+44 (0) 20 8867 6153

Alternatively, you can access the Rocket Customer Portal and report a problem, download an update, or read answers to FAQs. You will be prompted to log in with the credentials supplied as part of your product maintenance agreement.

To log in to the Rocket Customer Portal, go to:

www.rocketsoftware.com/support

Contents

About this Guide

This guide describes the characteristics, formats, and command syntax of the editors that you can use during a Model 204 session.

Audience

This guide is for programmers and users who have no programming experience. Familiarity with User Language is presumed. However, to use this guide, no further technical knowledge is required.

A note about User Language and SOUL

Model 204 version 7.5 provides a significantly enhanced, object-oriented, version of User Language called SOUL. All existing User Language programs will continue to work under SOUL, so User Language can be considered to be a subset of SOUL, though the name "User Language" is now deprecated. In this guide, the name "User Language" has been replaced with "SOUL."

Model 204 documentation set

To access the Rocket Model 204 documentation, see the Rocket Documentation Library (http://docs.rocketsoftware.com/), or go directly to the Rocket Model 204 documentation wiki (http://m204wiki.rocketsoftware.com/).

Documentation conventions

This guide uses the following standard notation conventions in statement syntax and examples:

Convention	Description
`TABLE`	Uppercase represents a keyword that you must enter exactly as shown.
`TABLE` *`tablename`*	In text, italics are used for variables and for emphasis. In examples, italics denote a variable value that you must supply. In this example, you must supply a value for *tablename*.
`READ [SCREEN]`	Square brackets ([ ]) enclose an optional argument or portion of an argument. In this case, specify READ or READ SCREEN.
`UNIQUE | PRIMARY KEY`	A vertical bar (|) separates alternative options. In this example, specify either UNIQUE or PRIMARY KEY.
`TRUST | NOTRUST`	Underlining indicates the default. In this example, NOTRUST is the default.
`IS {NOT | LIKE}`	Braces ({ }) indicate that one of the enclosed alternatives is required. In this example, you must specify either IS NOT or IS LIKE.
`item ...`	An ellipsis (. . .) indicates that you can repeat the preceding item.
`item ,...`	An ellipsis preceded by a comma indicates that a comma is required to separate repeated items.
`All other symbols`	In syntax, all other symbols (such as parentheses) are literal syntactic elements and must appear as shown.
`nested-key` `::=` *`column_name`*	A double colon followed by an equal sign indicates an equivalence. In this case, *nested-key* is equivalent to *column_name*.
`Enter your account:` `sales11`	In examples that include both system-supplied and user-entered text, or system prompts and user commands, boldface indicates what you enter. In this example, the system prompts for an account and the user enters **sales11**.
File > Save As	A right angle bracket (>) identifies the sequence of actions that you perform to select a command from a pull-down menu. In this example, select the Save As command from the File menu.
E`DIT`	Partial bolding indicates a usable abbreviation, such as E for EDIT in this example.

1

Model 204 Full-Screen Editor

Overview

The full-screen editor enables you to use the special capabilities of the IBM 3270 and compatible terminals.

Two editors

Model 204 provides the following editors, which you can use to enter, change, or add text to Model 204 requests and procedures:

Full-screen editor	Available for full-screen terminals.
Line editor	Available for line-at-a-time terminals. Optionally, you can invoke this editor from a full-screen terminal.

This chapter describes how to use the full-screen editor. Chapter 2 describes the line editor.

Editing capabilities

Special features of the full-screen editor include:

- Full-screen mode

  You are always in full-screen mode, even when text is being entered or added to a procedure. You can position the cursor at any text on the display screen and delete, insert, or change characters by typing at that point.

- Special screen display

  The display screen contains a display area and, optionally, either a prefix or a suffix area for entry of prefix commands to perform editing operations.

(Refer to page 8.) You can display a column scale at the center of the screen to facilitate the positioning of text in specific columns.

- Automatic text wrapping

  Lines that are longer than the normal display line are automatically wrapped onto subsequent display line(s).

- Global text searches

  You can search throughout a procedure for any string of characters. You can replace characters and specify wildcard characters to match all or part of the specified string. This feature is described on page 22.

- Prefixes and targets

  You can specify both editing operations (prefixes) and the lines on which the edits are to be performed (targets) by entering abbreviated commands in the prefix or suffix area of the screen. Prefixes and targets are described on page 8.

- PF key functions

  Use PF3 to quit, PF7 to scroll backward, PF8 to scroll forward, and PF9 to repeat commands while editing in full-screen mode.

Invoking the full-screen editor

Model 204 requirements

Before invoking the full-screen editor, set the Model 204 LOUTPB parameter to a minimum value of 3000. The LOUTPB parameter specifies the length of the output page buffer for 3270 terminals that use the full-screen capabilities. For more information about LOUTPB, refer to the Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/LOUTPB_parameter

Invoking the editor

You can invoke the full-screen editor in the following ways:

- Directly from Model 204 command level
- Indirectly in a User Language (SOUL) request

To invoke the full-screen editor from a terminal, enter the EDIT command. For additional information about the EDIT command, refer to the Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/EDIT_command

Using the EDIT command

To invoke the full-screen editor, use the EDIT command. The format of the EDIT command is as follows:

Syntax `EDIT(SCREEN)[oldproc,newproc]`

Specifying the type of editor

You can include a parameter that indicates which of the Model 204 editors to invoke. To request the full-screen editor, enter:

```
EDIT (SCREEN)
```

To request the line editor, enter:

```
EDIT (LINE)
```

If you do not specify either SCREEN or LINE, the appropriate editor for your terminal type is invoked. For example, if your terminal is an IBM 3270 and you enter EDIT while at command level or while executing a SOUL request, the full-screen editor is invoked automatically. The full-screen editor is not available from an IBM 3270 terminal that is connected to Model 204 through an access method that does not support full-screen input and output.

Specifying an input procedure

Use the *oldproc* parameter to name a temporary or permanent procedure to be used as input for the current editing session. If you do not specify an oldproc option, the full-screen editor creates a temporary procedure (Model 204 temporary procedure 0). Note that you cannot use the full-screen editor to create a new and permanent procedure if you specify a nonexistent input procedure.

Specifying an output procedure

Use the *newproc* parameter to specify an output procedure name. You can include an output procedure name in the EDIT command or specify a name when you exit from the procedure with an END, GO, or SAVE command. After the editing session terminates, Model 204 saves the procedure under the specified newproc name. If you specify a newproc parameter in the EDIT command and enter a procname parameter when exiting from the procedure, the procname parameter specified when exiting overrides the name specified in the EDIT command.

Rules for procedure names

The *oldproc* and *newproc* names can consist of any combination of letters, numbers, and symbols, except:

- Blank or space
- Comma
- Semicolon
- Minus sign
- Equal sign
- Single quote
- Carriage return

The maximum length of a procedure name is 255 characters.

Results of using the EDIT command

After you invoke the full-screen editor, use any of the commands described in this chapter. When you enter the full-screen editor, the first portion of the procedure to be edited appears in the display area and the cursor appears at the beginning of the command line. To reposition the cursor to the command line, press the HOME key (usually ALT-BACKTAB).

When you issue an EDIT command, the full-screen editor writes the following message to the Model 204 audit trail:

```
M204.0685: EDITING INTO newproc
```

Invoking the editor indirectly

You can invoke the full-screen editor indirectly by including the EDIT command in a SOUL request by using the TERMINAL option in the following format:

Syntax `EDIT (TERMINAL)`

TERMINAL option of EDIT

The TERMINAL option invokes the full-screen editor from within a procedure and connects the Editor to your terminal for input. For more information about the TERMINAL option, refer to the Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/EDIT_command

Display screen format

The typical format of the display screen during an editing session is shown in Figure 1-Figure 1-1.. The areas numbered in the diagram are described below the diagram.

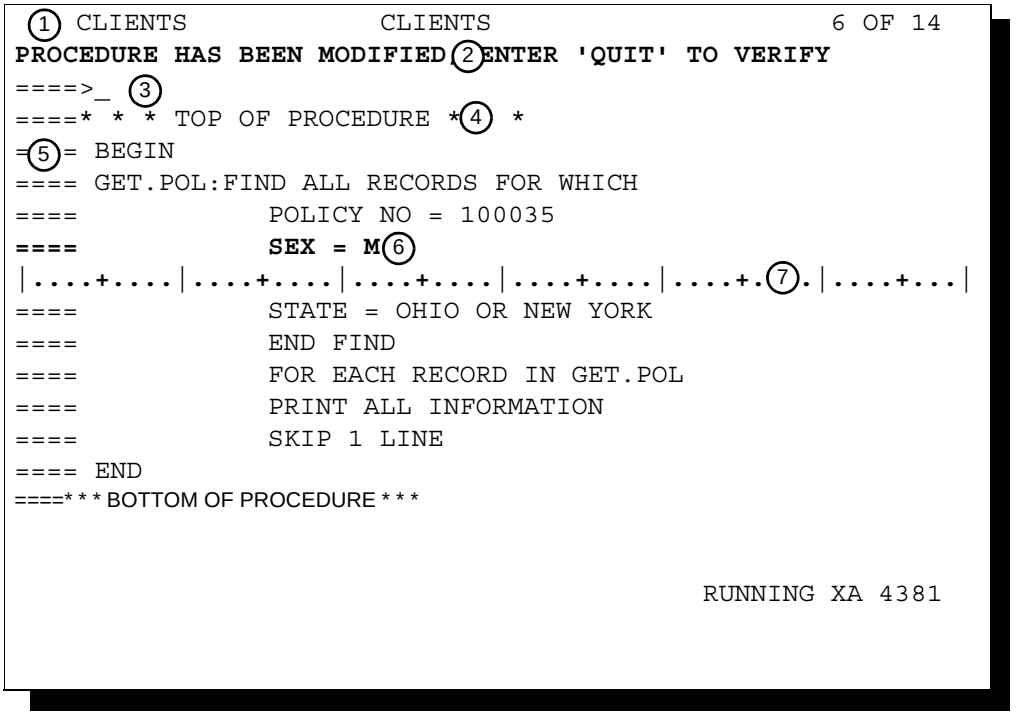

Figure 1-1. Sample Display Screen

Screen components

The components of the display screen are:

1. Header Line

   The top line of the screen is reserved for procedure identification. The header line contains the following information:

   – Old procedure name (before editing)

   – New procedure name (after editing)

   – Line number of the current line within the procedure

   – Current maximum line number within the procedure

2. Message Line

   The second line is reserved for error and status messages.

3. Command Line

   The third line is reserved for the entry of screen commands. This line begins with the following command prompt:

   ```
   ===>
   ```

   You can enter a command in uppercase or lowercase letters or a combination of the two.

Refer to Appendix A for command abbreviations. Appendix A also lists defaults for full-screen commands.

Enter one command at a time. If you enter more than one command, only the first command is executed.

4. Display Area

   The display area contains a portion of the text that is being edited. The text on the screen is considered a window in which only part of the procedure currently being edited appears.

   To manipulate displayed procedures, first position the cursor and then add, change, or delete text. To execute changes, press the Enter key.

   To position other parts of the procedure in the edit window, you can use the display commands or PF keys (refer to page 12) to scroll forward and backward through the procedure.

5. Prefix or Suffix Area

   This area can either precede or follow the display area on the screen. The columns typically display the following special characters:

   ```
   =====
   ```

   You can replace one or more of these equal signs with a special prefix command to perform an editing function, such as inserting or deleting text on the corresponding line in the display area. Refer to page 10 for more information about prefix commands.

6. Current Line

   The line being edited normally appears at the center of the screen and is highlighted (that is, displayed more brightly than the surrounding text).

   The *line pointer* points to the current line. The line pointer is repositioned to a new current line when you scroll forward and backward and when you execute screen commands. You can position the line pointer at a new current line by specifying the set line prefix command (*/*) (refer to page 11).

7. Column Scale

   The column scale indicates the column position of the procedure text that appears. You can display a column scale immediately below the center line (normally the current line) in the display area. The scale is useful when text must be entered in particular columns.

Requesting a column scale

The SCALE command specifies whether or not the full-screen editor displays a column scale in the center of the display area on the screen. The format of the SCALE command is:

Syntax `[SET] SCALE {OFF | ON}`

SCALE OFF is the default.

If you specify SCALE OFF, the column scale is not displayed. If you specify SCALE ON, the scale is displayed. The column scale is shown in the illustration of display screen characteristics in Figure 1-1 on page 5.

Positioning the prefix area

The PREFIX command positions the prefix area on the left (prefix) or right (suffix) side of the display screen. PREFIX LEFT is the default. The format of the PREFIX command is as follows:

Syntax `[SET] PREFIX {LEFT | RIGHT}`

The full-screen editor positions the prefix area at the end of each line of the display area when you specify PREFIX RIGHT, and at the beginning of each line when you specify PREFIX LEFT.

Figure 1-1 on page 5 shows the positioning of the prefix area on the left side of the display screen. For a description of the prefix commands that can be entered in this area, refer to page 8.

Editing modes

The editing modes are:

- Command mode (used primarily for editing)
- Input mode (used primarily for entering new text)

Command mode

When you invoke the full-screen editor, you are placed in command mode. To position the cursor to the command line, press the HOME key (usually ALT/BACKTAB).

In command mode, you enter commands for the full-screen editor either on the command line or in the prefix area.

Input mode

Enter input mode to type new lines of text. To enter input mode, specify an I* prefix command. You cannot use the prefix commands in input mode.

To return to command mode from input mode, press the Enter key twice without any intervening text.

Using targets

Some prefix and full-screen editor commands operate on a line or set of lines called a *target*.

For example, if you want to search for a text string within lines 10 and 20 of a procedure, lines 10 through 20 are considered target lines.

Specify a target on the appropriate line in the prefix or suffix area of the screen.

When specifying a target, you can use uppercase and lowercase interchangeably. Targets are specified as one of the following:

P	Precedes the target
F	Follows the target
R(*n*)	Target range
RR	Target range block

Using targets with prefix commands

The prefix commands are described in the rest of this chapter, along with command line commands, by functional group. Appendix A also provides a summary of full-screen editor commands

P target command

Use the P target command to define a target above the line on which you enter the P. For example, to move a group of lines within a procedure to precede line 20, place a P target on the line 20 prefix.

F target command

Use the F target command to define a target below the line on which you enter the F. For example, if you wanted to move a group of lines within a procedure to follow line 20, place an F on the line 20 prefix.

Specifying range targets

Several commands let you include a range specification that indicates the range and direction of the target lines. The SAVE, LOCATE, and REPLACE commands define a set of target lines by including range specifications. For LOCATE, if one of these specifications is omitted, the full-screen editor processes the set of lines from the current line to the end of the procedure and then wraps to the top of the procedure and continues to the current line.

R(*n*) Target command

Use the R(*n*) target command to define a target in a range of *n* lines beginning with the line on which you enter the R(*n*). For example, if you entered R6 on line 15, the range begins on line 15 and ends with line 20.

RR Target command

Use the RR target command to define a target in a range of lines beginning with the line on which you enter the first RR and ending on the line on which you enter the second RR. For example, to specify a range from line 15 to 20, place one RR target on line 15 and another RR target on line 20.

Range target characters

Range specifications are summarized in Table 1-1.

Table 1-1. Range specifications

This specification...	Has this effect...
ALL	Begin at the procedure top and end at the procedure bottom; equivalent to TOP TO BOTTOM.
BOTTOM [TO] CURRENT	Proceed backward through the procedure; begin at the bottom of the procedure and end at the current line.
BOTTOM [TO] TOP	Proceed backward through the procedure; begin at the procedure bottom and end at the top of the procedure.
CURRENT [TO] BOTTOM	Begin at the beginning of the current line and end at the procedure bottom.
CURRENT [TO] TOP	Proceed backward through the procedure; begin at the current line and end at the procedure's beginning.
RANGE	If R*n* is specified as a prefix target command, begin at the beginning of the line specified by R*n* and end at the end of the line that is *n* – 1 lines below the R line. If RR is specified as a prefix target command, begin at the beginning of the line that contains the first RR and end at the line that contains the second RR.
TOP [TO] BOTTOM	Begin at the procedure top and end at the procedure bottom; equivalent to ALL.
TOP [TO] CURRENT	Begin at the top of the procedure; end at the bottom of the procedure

Note: You can use the first initial of top, current, bottom, and range. For example, TOP TO CURRENT is the same as T TO C.

Clearing targets

Targets remain in effect until one of the following events occurs:

- Commands that use the targets complete execution.
- CLEAR command is executed (refer to page 11).

If you specify a target range, the full-screen editor does not automatically clear the range when the command completes; it preserves the range so that it can be used multiple times. To cancel a target range, enter a CLEAR command.

Prefix commands

Prefix commands and targets execute full-screen editor operations. A *prefix command* is an abbreviated command that specifies an editing operation to be performed on a particular line or set of lines.

Enter the command in the prefix area of the line that you want to change. For example, to delete a line, enter a D in the prefix area of the line and press Enter.

The prefix commands are described in the rest of this chapter, along with command line commands, by functional group. Appendix A also provides a summary of full-screen editor commands.

Summary of prefix commands

Table 1-2 summarizes the prefix commands that you can enter in the prefix area of the display screen. Some prefix commands must be used with a target command. You can enter prefix characters in uppercase or lowercase and in any position in the prefix area.

Table 1-2. Prefix commands

Command	Meaning
`E`	Extends the line by inserting a continuous line that contains blanks or nulls.
`I*`	Enters input mode.
`I[`*n*`]`	Inserts one or more lines containing blanks or nulls following this line.
`"[`*n*`]`	Duplicates one or more lines.
`""[`*n*`]`	Duplicates a block.
`M[`*n*`]`	Moves one or more lines.
`MM`	Moves a block.
`C[`*n*`]`	Copies one or more lines.
`CC`	Copies a block.
`D[`*n*`]`	Deletes one or more lines.
`DD`	Deletes a block.
`U`	Restores the line(s) or block most recently deleted.
/	Makes this line the current line.

Clearing the prefix area

The CLEAR command performs the following operations:

- Clears all prefix lines in the prefix or suffix area on the display screen.
- Clears any display area text that has been typed since the last time you pressed Enter (or a PF key).
- Aborts any pending prefix operations.
- Cancels any specified targets.

For more information about prefixes and targets, see page 8.

Pressing the CLEAR, PA1, or PA2 key on the 3270 terminal has the same effect as entering the CLEAR command.

Command-line commands

In addition to prefix commands, which you enter in the left-hand column (prefix area) of the line(s) you want to edit, the Model 204 full-screen editor provides several command-line commands.

Current line

Several command-line commands take effect from the current line. The current line is the line of the procedure on which the full-screen editor acts.

Use the set line prefix command (/) to set the current line.

Setting the current line

To define a new current line, type the set line command (/) in the prefix area of that line. This command is executed before any command-line commands specified on the same screen.

This feature is especially useful with the LOCATE command (see page 20). For example, suppose that you want to locate the next occurrence of a string, but not the occurrence on the line following the current line (the line on which LOCATE would normally start its search). You can use the / prefix to reposition the current line so that LOCATE begins its search below the unwanted occurrence.

Where to enter command-line commands

Enter command-line commands on the third line down from the top of the screen, which is the command line. It has the following prompt

```
===>
```

Entering commands

When entering command-line commands, follow these guidelines:

- Enter one command at a time. Press Enter after the command. If you enter more than one command, only the first command is executed.
- You can enter a command in uppercase or lowercase, or a combination of the two.
- You can abbreviate some commands; abbreviations are indicated in this chapter and Appendix A.
- Several commands have defaults; those defaults are described in this chapter and Appendix A.

Repeating commands

Placing an equal sign (=) on the command line repeats the most recently executed command.

This command executes a command any number of times and is useful for performing a repetitive operation, such as searching for a string, without reentering the command each time.

You can also repeat a command by pressing PF9.

Moving around a procedure

The full-screen editor display commands move through a procedure and display needed portions of it on the screen. Display commands move the current line pointer and control the appearance of portions of the procedure, but they do not change the contents of the procedure itself.

Display commands are summarized in Table 1-3 and are discussed in detail on the pages that follow.

Table 1-3. Display commands

Command	Meaning
TOP	Scrolls to the first line of the procedure.
BOTTOM	Scrolls to the last line of the procedure.
FORWARD [*n*]	Scrolls forward a specified number of screens in the procedure.
BACKWARD [*n*]	Scrolls backward a specified number of screens in the procedure.
+*n*	Scrolls forward a specified number of lines in the procedure.

Table 1-3. Display commands

Command	Meaning
-n	Scrolls backward a specified number of lines in the procedure.

Moving to the top of the procedure (TOP)

The TOP command moves the line pointer to the first line of the procedure that is being edited. After you issue a TOP command, the full-screen editor displays the TOP OF PROCEDURE indicator on the first line of the display area. It displays the first line of the procedure on the second line. The remaining lines of the procedure appear on subsequent lines. The first line of the procedure becomes the current line and is highlighted in the display.

The TOP command is helpful when inserting a line or set of lines at the beginning of the procedure. In a procedure that fills many screens; using the TOP command is often more efficient than scrolling backward using the BACKWARD command.

Moving to the bottom of the procedure (BOTTOM)

The BOTTOM command moves the line pointer to the last line of the procedure that is being edited. After you issue a BOTTOM command, the full-screen editor displays the BOTTOM OF PROCEDURE indicator on the last line of the screen. The preceding lines of the procedure appear on the preceding lines of the screen. The last line of the procedure becomes the current line and is highlighted in the display.

The BOTTOM command is helpful when adding a line or set of lines to the end of the procedure. In a procedure that fills many screens, using the BOTTOM command is often more efficient than scrolling forward using the FORWARD command.

Scrolling forward (FORWARD)

The FORWARD command scrolls forward in the procedure by moving the line pointer down a specified number of screens. The format of the FORWARD command is as follows:

Syntax **FO**RWARD [*n*]

If you enter FORWARD, the full-screen editor advances one entire screen. If you enter FORWARD followed by a number (*n*), the full-screen editor advances the specified number of screens.

After you issue a FORWARD command, the new current line is displayed highlighted at the center of the screen. The remainder of the screen is filled with as many as possible of the lines preceding and following the current line. The

bottom line of the previous screen display becomes the top line of the new screen display.

If the current line is the first line of the procedure when you issue the FORWARD command (that is, you are positioned at the top of the procedure), the full-screen editor advances only one half a screen and displays the new current line in the center of the display screen.

You can also press PF8 to scroll forward.

Scrolling backward (BACKWARD)

The BACKWARD command scrolls backward in the procedure, moving the line pointer up a specified number of screens. The format of the BACKWARD command is as follows:

Syntax **BA**CKWARD[n]

If you enter BACKWARD, the full-screen editor scrolls backward one entire screen. If you enter BACKWARD followed by a number (*n*), the full-screen editor scrolls backward the specified number of screens.

After you issue a BACKWARD command, the new current line is displayed highlighted at the center of the screen. The remainder of the screen is filled with as many as possible of the lines that precede and follow the current line.

If the current line is the last line of the procedure when the BACKWARD command is issued, the full-screen editor scrolls backward only half a screen and displays the new current line in the center of the display screen. Otherwise, the BACKWARD command causes the top line of the previous screen display to become the bottom line of the new screen display. In either case, the BACKWARD command causes the current line to be backed up by the specified number of screens.

You can also press PF7 to scroll backward.

Scrolling a specified number of lines (+/–*n*)

The scroll commands (+*n* and –*n*) scroll forward or backward, moving the line pointer a specified number of lines in the procedure. These commands are useful for moving a portion of the procedure up or down.

If you enter a number, optionally preceded by (+) the full-screen editor advances the specified number of lines. For example, if you enter either 5 or +5 the Editor advances the current line forward five lines toward the end of the procedure and adjusts the display on the screen.

If you enter –, followed by a number, the Editor moves backward the specified number of lines. For example, if you enter –10, the Editor moves backward 10 lines toward the beginning of the procedure and adjusts the display on the screen.

In general, the current line is displayed highlighted at the center of the screen. However, if the current line is less than half a screen from the top or bottom of the screen when you issue the command, the full-screen editor shifts the display to fill the screen.

Note that +*n* and –*n* scroll a specified number of lines in the procedure. Because a single procedure line can occupy more than a single physical line on the display screen, these commands might actually appear to scroll forward or backward a greater number of display lines than specified.

Searching for and replacing text

The full-screen editor LOCATE command allows you to search for text strings in a procedure.

The REPLACE command allows you to locate and change text strings.

These two commands share many characteristics, including:

- Rules for text string searches
- Use of special search characters
- Option to specify a range of procedure lines

Uppercase and lowercase in searches

With the exception of the following special search characters, the full-screen editor searches for an exact match to the string specified in the LOCATE command. The case (upper and lower) in which the string is specified and the placement of blanks are significant. Therefore, if you specify the command,

```
LOCATE /Model 204
```

the Editor locates an occurrence of Model 204, but not MODEL 204 or MODEL204 in the procedure.

Note that although lowercase characters are entered in a procedure, Model 204 does not necessarily process these characters as lowercase. Unless the Model 204 command, *LOWER, was specified before entering the full-screen editor, characters are translated automatically into uppercase when they are transmitted to Model 204. For additional information about using uppercase and lowercase characters, refer to the *UPPER and *LOWER commands in the Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/*UPPER_command

http://m204wiki.rocketsoftware.com/index.php/*LOWER_command

Specifying a range

You can limit the scope of a text string search by indicating a range.

The range entry indicates the target lines and the order of the lines to be searched for a match. You can include any of the range specifications summarized on page 8. If you omit the range, the full-screen editor searches the current line.

Special search characters

You can use special search characters when specifying a search string with the LOCATE and REPLACE commands.

These special characters are:

- Arbitrary (wildcard) character
- Repeat character
- End-of-line character

Wildcard character

An arbitrary or wildcard character is matched by any character. For example, the ampersand character (&) is your wildcard character, and you specify the following search string:

MODEL&

The following strings in the procedure can be located:

```
MODELA
MODEL2
MODEL&
```

Setting the wildcard character (ARBCHAR)

The ARBCHAR command specifies a wildcard character that matches zero or more characters when it is included in a search string. The format of the ARBCHAR command is as follows:

Syntax `[SET] ARBCHAR {OFF | char}`

where:

char	Represents the wildcard or arbitrary character.

ARBCHAR OFF is the default.

Suppose that you specify:

```
ARBCHAR *
```

Then you use the asterisk in the LOCATE command:

```
LOCATE /PROC*
```

The full-screen editor searches for strings that contain the letters PROC, followed by any character. The following strings are found:

```
PROCA
INPROC3
PROC *.
```

The following strings are not found:

```
PRO12
PRO CABC
```

To turn off the wildcard character capability, enter:

```
ARBCHAR OFF
```

If you enter ARBCHAR OFF, none of the characters in the specified search strings in subsequent LOCATE and REPLACE commands are considered wildcard characters, and all strings in the procedure must match the search string exactly.

Repeat character

The repeat character indicates that the previous character in the string can be repeated any number of times. For example, your repeat character is an exclamation point (!) and you specify the following search string:

```
Format *!
```

Any of the following strings in the procedure are located:

```
Format *
Format **
Format ****************
```

Setting the repeat character (REPEAT)

The REPEAT command specifies a repeat character that matches zero or more occurrences of the repeating character when it is included in a search string. The format of the REPEAT command is as follows:

Syntax `[SET] REPEAT {OFF | char}`

where:

char	Represents the repeat character.

REPEAT OFF is the default.

Suppose that you specify:

```
REPEAT &
```

Then you use the & in the LOCATE command:

```
LOCATE /MODEL*&
```

The full-screen editor searches for strings that contain the characters MODEL, followed by any number of asterisks. The following strings are selected:

```
MODEL*
MODEL***
MODEL**
```

The following strings are not selected:

```
MODEL1
MOD***
MODEL *
```

You can use repeat characters in conjunction with wildcard characters. For example, suppose that the wildcard character is an exclamation point (!) and the repeat character is and ampersand (&) and you specify the following search string:

```
/PRE!&
```

The full-screen editor locates any string that begins with the letters PRE, followed by any number of any other characters.

To turn off the repeat character capability, enter:

```
REPEAT OFF
```

If you enter REPEAT OFF, none of the characters in the specified search strings in subsequent LOCATE or REPLACE commands are considered repeat characters.

End-of-line character

The end-of-line character indicates the end of one line and the beginning of the next line. When the end-of-line character is used in a search string, it specifies a search for a string that occurs at the beginning or end of a line. For example, the percent sign (%) is your end-of-line character and you specify the following search string:

```
%prefix
```

Any line beginning with the word prefix is located:

```
prefix area!!!
prefixes!!!
prefix 12!!!
```

The percent sign (%) is your end-of-line character and you specify the following search string:

```
prefix%
```

Any line ending with the word prefix is located.

Specify an end-of-line character only at the beginning or at the end of the search string.

Setting the end-of-line character

The LINEND command specifies an end-of-line character. When you use an end-of-line character in a search string, it represents a character that indicates the end of one line and the beginning of the next line. The Editor searches for a string that occurs at the beginning or end of a line.

The format of the LINEND command is as follows:

Syntax `[SET] LINEND {OFF | char}`

where:

char	Represents the end-of-line character.

LINEND OFF is the default.

Suppose that you specify:

```
LINEND #
```

Then you use the pound sign (#) in a search string:

```
LOCATE /#PART
```

The full-screen editor locates any line that begins with the letters PART.

You must specify the end-of-line character at the start or end of a search string.

To turn off the end-of-line character capability, enter:

```
LINEND OFF
```

If you enter LINEND OFF, none of the characters in the search strings specified in subsequent LOCATE or REPLACE commands are considered end-of-line characters.

Combining special search characters

You can specify combinations of characters in the search string. For example, the following search string locates any string that contains HELP, followed by any number of characters, followed by ME:

```
HELP&!ME
```

For example:

```
HELP123ABME
HELP ME
```

Searching for a string (LOCATE)

The LOCATE command searches for a particular string in the procedure or in a subset of the procedure. See page 22 for more information about string specifications.

Two formats of the LOCATE command

There are two formats of the LOCATE command.

With LOCATE keyword

The first format is as follows:

Syntax `[-][LOCATE] /string[/ [IN]range]`

where the delimiting character (/ in the example above) can be any character not appearing in the string, with the exception of a blank character. For example:

```
LO *ELEPHANT* IN CURRENT TO TOP
```

If you do not specify a range, you do not need to include a delimiter after the string. If you specify an alphabetic delimiter, include a space between the LOCATE or LO keyword and the delimiter. If the delimiter is nonalphabetic, the space is not required.

See page 8 for more information about using a range; see page 9 for a list of range specifications.

Without LOCATE keyword

The second format, which does not require the LOCATE keyword, is as follows:

Syntax `[-] /string`

where a slash (/) must precede the string. If you do not specify a range, you do not need to include a slash following the string.

If the full-screen editor cannot find a match for the specified string, it displays the following message:

```
TARGET STRING NOT LOCATED
```

Searching forward

To search forward, enter one of the following formats:

```
LOCATE /string[/ [IN] range]

/string
```

The full-screen editor starts the search at the line following the current line and advances forward, stopping when it locates the string or reaches the end of the procedure. If the current line is the last line of the procedure, the full-screen editor wraps to the top of the procedure, begins the search at the top of the procedure, and ends at the current line.

Searching backward

To backward, enter one of the following formats:

```
-LOCATE /string[/ [IN] range]

-/string
```

When you precede the command with a minus sign, the full-screen editor starts the search at the line preceding the current line and searches backward toward the beginning of the procedure, stopping when it locates the string or reaches the top of the procedure. If the current line is the top of the procedure, the full-screen editor wraps to the bottom of the procedure, begins the search at the bottom line, and ends at the current line.

Replacing a string (REPLACE)

The REPLACE command replaces a string in the procedure with a substitute string. REPLACE provides the following capabilities:

- Displays each target string to verify the change before proceeding.
- Changes more than one occurrence of a specified string in a single line.

The format of the REPLACE command is as follows:

Syntax `REPLACE/string1/string2 [n | *] [VERIFY [IN] range]`

If you include the *n*, VERIFY, or range options in the REPLACE command, specify the options in the order shown. For example:

```
R /GROUP/PERM GROUP/ * IN CURRENT TO TOP
```

Specifying a string

Specify strings as follows:

string1	String to be searched for and changed (old string).
string2	New string to replace *string1*. The new string need not be the same length as the old string.

You can use as a delimiter any nonblank character (/ in the example above) that does not appear in *string1* or *string2*. If you do not specify the number of occurrences, a VERIFY option, or a range, you do not need to include a delimiter after *string2*.

Controlling the number of strings to be replaced

You can replace all occurrences of a string or only the first occurrence in a line. The *n* specification in the REPLACE command represents the number of occurrences of the search string (*string1*) to be replaced in each line. If you omit an *n* specification, only the first occurrence of the string is replaced in a line.

If you specify a value of *, all occurrences of the string are replaced in each line.

Using the VERIFY option

The VERIFY option verifies all string replacements before they are made. Each line that contains a string that matches the specified string is highlighted and is delimited on the scale line by the characters of the proper length, on the column scale line directly beneath the potential match. For example:

```
<=====>
```

The column scale must be turned on to display the scale line characters. The following prompt appears on the command line each time a string is matched:

```
CHANGE?
```

Respond with any of the following choices:

This choice...	Performs this action...
YES, Y, or Enter	Changes the string
NO or N	Does not change the string
ALL	Turns off the VERIFY option and makes all remaining changes
STOP or S	Terminates the REPLACE command

If you enter any other response, the full-screen editor displays the CHANGE? prompt again.

Specifying a range

You can specify a range of procedure lines to which the search applies. See page 8 for more information about using a range; see page 9 for a list of range specifications.

Inserting text

You can insert text several ways using the full-screen editor. Use the following commands to insert text:

This command...	Performs this action...
`E`	Extends a line
`I`	Inserts separate lines
`I*`	Enters input mode
`GET`	Inserts a saved procedure

You can also insert text using the REPLACE command to substitute a string with additional text for an existing string (see page 22).

Alternatively, you can insert text by duplicating, copying, and moving lines or blocks of text (see page 25).

Extending a line

The E (extend) prefix command inserts a physical line immediately following the line that contains the E, forming a logical line extension. The line is filled with the fill character (that is, blanks or nulls) specified in the FILL command. For more information about the FILL command and fill characters, see “Specifying a fill character” on page 24.

Inserting lines

The I (insert) prefix command inserts one or more lines following the line containing the I. Unlike E lines, these lines are treated as separate lines, not as continuations of existing lines in the procedure.

The format of the I prefix command is:

Syntax `I[`*`n`*`]`

If you enter I alone, the full-screen editor inserts one line following the line containing the I. If you enter I followed by a number, the full-screen editor inserts the specified number of lines. For example, I5 inserts 5 blank lines after the line that contains the command.

Inserted lines are filled with the fill character (blanks or nulls) specified in the FILL command. For more information about the FILL command and fill characters, see “Specifying a fill character” on page 24.

Specifying a fill character

The FILL command specifies a fill character. When the full-screen editor displays a line of text, it pads the line on the right with either null or blank characters. The format of the FILL command is as follows:

Syntax `[SET] FILL {NULL | BLANK}`

If you specify FILL NULL, the full-screen editor pads with null characters on the right. If you specify FILL BLANK, the full-screen editor pads with blank characters on the right.

The character selected as the fill character is inserted in lines generated using the E (extend) and I (insert) prefix commands.

Note: The command line on the display screen is always filled with null characters, regardless of the specified fill character.

Entering input mode

The I* prefix command places the full-screen editor in input mode. The line that contains I* becomes the current line and appears at the top of the screen. The rest of the screen is blank. Any lines following the I* line in the original procedure are not displayed.

For input lines, the cursor appears in column 1 of the first blank line. You can now enter text. If you press Enter, the last line that you typed is moved to the top of the screen and the cursor appears on the next (blank) line. To exit from input mode, either press Enter twice without any intervening text editing, or enter any full-screen editor command on the command line.

Inserting a saved procedure

The GET command inserts a previously saved procedure after the current line. You can insert a procedure that is stored in either the default procedure file or in a file specified in the command (with the *filespec* option). The last line of the inserted procedure becomes the current line and is positioned at the center of the display screen.

Before you issue the GET command, position the cursor at the appropriate line. For example, if you want to add one procedure to the end of another, use the BOTTOM command to advance the cursor to the end of the procedure before the procedure is added.

The format of the GET command:

Syntax `[IN filespec] GET procname`

where:

filespec	Is the name of the procedure file in which the combined procedure is to be stored. This file is used in place of the Model 204 default procedure file. The filespec can identify a Model 204 file or a temporary or permanent group.
	The format for filespec is:
	`[[{PERM | TEMP} GROUP] | FILE] name`
	For example:
	`FILE PAYROLL`
	`TEMP GROUP EMPINFO`
	`ACCOUNT`
	`If neither FILE nor GROUP is specified, FILE is the default.`

procname	Is the name of the procedure to be inserted after the current line. This procedure can be stored in any Model 204 file that you currently have open.

Duplicating, copying, and moving text

You can save keystrokes and reorganize text by using the full-screen editor commands for the following tasks:

- Duplicating lines of text (" or "")
- Copying lines of text (C or CC)
- Moving lines of text (M or MM)

Groups of lines are also referred to as blocks. You can also delete and undelete lines of text (see page 27).

Duplicating lines or blocks

You can duplicate either one line or a block of lines.

Duplicating one or more lines

The duplicate (") prefix command duplicates the line on which it is entered. The format of the duplicate command is as follows:

Syntax `"[n]`

If you enter ", the full-screen editor inserts one copy of the line. If you enter " followed by a number, the full-screen editor inserts the specified number of duplicate lines. For example, "5 duplicates a line five times.

Duplicating a block of text

The duplicate block ("") prefix command duplicates a group of lines. Entering "" duplicates the block of lines beginning with the line on which the first "" is specified and ending with the line on which the second "" is specified. The format of this command is as follows:

Syntax `""[n]`

If you enter "", the full-screen editor inserts one copy of the block of lines. The block appears immediately following the line that contains the second "" (the last line of the block). If you enter "" followed by a number, the full-screen editor duplicates the block the specified number of times.

Copying lines or blocks

The C (copy) prefix command copies one or more lines to the position indicated by the P (precedes) or F (follows) prefix target. If you specify P, the line(s) are inserted just before the line that contains the P. If you specify F, the lines are inserted just after the line that contains the F. For more information about prefix targets, refer to page 8.

The format of the C prefix command is as follows:

Syntax `C[n]`

If you enter C, the full-screen editor copies only the line that contains the C. If you enter C followed by a number, the full-screen editor copies the specified number of lines, beginning with the line containing the C*n*. When a line is copied, it is preserved in its original position as well as being duplicated in the new position.

The CC (copy block) prefix command copies a group of lines without requiring you to count them. CC copies the block of lines that begins with the line on which the first CC is specified and that ends with the line on which the second CC is specified. The lines are copied to the position indicated by the P (precedes) or F (follows) prefix target, as discussed for C.

Moving lines or blocks

The M (move) prefix command moves one or more lines to the position indicated by the P (precedes) or F (follows) prefix target (refer to page 8). If you specify P, the line(s) are inserted just before the line containing the P. If you specify F, the line(s) are inserted just after the line that contains the F.

The format of the M prefix command is as follows:

Syntax `M[n]`

If you enter M, the full-screen editor moves only the line that contains the M. If you enter M followed by a number, the full-screen editor moves the specified number of lines, beginning with the line that contains the M*n*. When a line is moved, it is automatically deleted from its original position.

The MM (move block) prefix command moves a group of lines without requiring you to count them. MM moves the block of lines that begins with the line on which the first MM is specified and that ends with the line on which the second MM is specified. The lines are moved to the position indicated by the P (precedes) or F (follows) prefix target (refer to page 8).

Deleting and undeleting text

You can delete lines from a procedure, using the D or DD prefix commands. You can also retrieve the last entry that was deleted with the U prefix command. Groups of lines are also referred to as blocks.

Deleting lines or blocks

The D (delete) prefix command deletes one or more lines from the procedure. The format of the D prefix command is as follows:

Syntax `D[n]`

If you enter D, the full-screen editor deletes only the line that contains the D. If you enter D followed by a number, the full-screen editor deletes the specified number of lines, beginning with the line that contains the D. If the number of lines specified in the D command exceeds the number remaining in the procedure, the full-screen editor deletes all lines from the line on which the D command is specified to the end of the procedure.

When a line or set of lines is deleted, the lines are not immediately discarded; they are placed on the full-screen editor delete stack. Entries in this stack are deleted when the editing session is terminated. Each time a D command is executed, an entry is placed on the delete stack. A D*n* command, which deletes several lines, causes a single entry that contains the specified block of lines to be placed on the stack. However, a series of contiguous lines, each containing a D command, is not accumulated into a single entry on the delete stack. You can retrieve or undelete an entry by means of the U command, which is described in the section “Undeleting lines or blocks”.

The DD (delete block) prefix command deletes a group of lines without requiring you to count them. DD deletes the block of lines that begins with the line on which the first DD is specified and that ends with the line on which the second DD is specified. The block is placed on the delete stack as a single entity.

Undeleting lines or blocks

The U (undelete) prefix command retrieves the last entry that was deleted (that is, placed on the delete stack) and inserts it after the line that contains the U.

Exiting from the full-screen editor

To exit from the full-screen editor or to save edited procedures, use the commands listed in Table 1-4, which also lists the description of each command and where more information is located. When you exit from the full-screen editor, you are returned to the Model 204 command level.

Table 1-4. Exit and save commands

This command...	Performs this action...
END	Exits from the full-screen editor and saves the edited procedures.
GO	Exits from the full-screen editor, saves the edited procedure, and includes (executes) the edited procedure.
QUIT	Exits from the full-screen editor, might prompt if changes were made to the procedure, and does not save the edited procedure.
SAVE	Does not exit from the full-screen editor, but saves the edited procedure.

For more information about exit and save commands, refer to the Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/EDIT_command

Exiting from the editor and saving the procedure (END)

The END command exits from the full-screen editor and saves the edited procedure on disk. Specify the name under which the procedure is to be saved. You can store the procedure in either the default procedure file or in a file specified in the command.

The format of the END command is as follows:

Syntax `[IN filespec] END [procname]`

where:

filespec	Is the name of the procedure file in which the edited procedure is to be stored. This file is used in place of the default Model 204 procedure file. The filespec can identify a Model 204 file or a temporary or permanent group. The file specified by IN filespec must be open. The format of filespec is:
	`[[{PERM | TEMP} GROUP] | FILE] name`
	For example:
	`FILE PAYROLL`
	`TEMP GROUP EMPINFO`
	If you do not specify either FILE or GROUP, FILE is assumed. For more information about files and groups, see the Model 204 documentation wiki:
	• http://m204wiki.rocketsoftware.com/index.php/CREATE_command:_File
	• http://m204wiki.rocketsoftware.com/index.php/CREATE_command:_Permanent_group
	• http://m204wiki.rocketsoftware.com/index.php/CREATE_command:_Temporary_group

procname	Is the name under which the edited procedure is to be saved.

If you enter END, the procedure is saved under the name of the output procedure specified when you invoked the Editor. The procedure is stored in the Model 204 default procedure file.

Saving and including the procedure (GO)

Like the END command, the GO command exits from the full-screen editor and saves the edited procedure. Unlike END, GO automatically performs an implicit Model 204 INCLUDE command, which executes the Model 204 commands in the saved procedure.

For more information about the INCLUDE command, refer to the Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/INCLUDE_command

The format of the GO command is as follows:

Syntax `[IN filespec] GO [procname]`

GO command options are identical to END command options. If you enter GO, the full-screen editor saves the procedure under the name of the output procedure specified when you invoked the full-screen editor. The procedure is stored in the Model 204 default procedure file.

Exiting from the editor without saving the edited procedure (QUIT)

The QUIT command exits from the full-screen editor.

The syntax is as follows:

Syntax `QUIT`

If you change the procedure during the current editing session and exit by entering QUIT, the full-screen editor displays the following message, which indicates that the procedure was changed and requests that you to enter QUIT again, if you want to exit the procedure without saving the edits:

```
PROCEDURE HAS BEEN MODIFIED, ENTER 'QUIT' TO VERIFY.
```

Respond in one of the following ways:

- Enter QUIT on the command line to confirm that changes are not to be saved.
- Ignore the prompt and continue editing.

To exit without receiving the warning prompt, enter the following command:

```
QUIT QUIT
```

If you use QUIT to exit from a file that was examined but not modified, the full-screen editor does not prompt before exiting.

You can also press PF3 to quit. Your edits are not saved.

Saving the procedure (SAVE)

The SAVE command saves the edited procedure or a subset of that procedure under the specified name (*procname*) or the default procedure name. Unlike the END, GO, and QUIT commands, SAVE does not exit from the Editor. To ensure against the loss of changes due to a system failure, issue the SAVE command periodically while you edit a procedure.

The format of the SAVE command is as follows:

Syntax `[IN filespec] SAVE [procname | procname] [[IN] range]`

where:

filespec	Is the name of the procedure file in which the edited procedure is to be stored. This file is used in place of the Model 204 default procedure file. The filespec can identify a Model 204 file or a temporary or permanent group. The file specified with IN filespec must be open. It has the same syntax as described for the END command.

procname	Is the name under which the edited procedure is to be saved.
range	Is the range to be saved. Note that if you specify range, you must also specify procname. You can include any of the range specifications summarized on page 1-13. If you omit a range, the entire procedure is saved.

If you enter SAVE alone, the full-screen editor saves the entire procedure under the name of the output procedure specified when the Editor was invoked. The procedure is stored in the Model 204 default procedure file.

2

Model 204 Line Editor

Overview

The Model 204 line editor allows you to edit procedures on line-at-a-time terminals.

Two editors

Model 204 provides the following editors, which you can use to enter, change, or add text to Model 204 requests and procedures:

Full-screen editor	Available for full-screen terminals.
Line editor	Available for line-at-a-time terminals. Optionally, you can invoke this editor from a full-screen terminal.

This chapter describes how to use the line editor. Chapter 1 describes the full-screen editor.

Editing capabilities

The line editor provides the following editing functions:

- Positioning a pointer at any character or line in the text
- Displaying a line number or a specified portion of the text
- Replacing lines or characters in the text
- Inserting lines or characters
- Deleting lines or characters
- Combining several procedures into a new procedure

Invoking the line editor

You can invoke the line editor in the following ways:

- Directly from Model 204 command level
- Indirectly in a SOUL request

Entering the EDIT command

To invoke the line editor from a line-at-a-time terminal, enter the EDIT command. For full-screen terminals, invoke the line editor by using a special option of the EDIT command (see “Invoking the editor” on page 2). After invoking the line editor, you can execute any of the commands described in this chapter:.

For line-at-a-time terminals, to invoke the line editor, enter:

```
EDIT
```

Optionally, you can specify the name or number of the text to be edited. The system responds by displaying:

```
M204.0526: EDITING INTO name
```

The use and construction of line editor names and numbers are discussed below.

EDIT command restrictions

The following restrictions are for line editor input and commands:

- You can issue the EDIT command within a procedure but you must not included it within a request in the procedure. That is, the command must not fall within the scope of a BEGIN;...END sequence. Furthermore, an EDIT command cannot modify the procedure in which it occurs.
- If you include the EDIT command within a SOUL request entered at a terminal, invoking the editor closes any USE ddname data sets that you have open.
- EDIT is recognized in ad hoc requests even when the editor is not included in the system. If the line editor is not available, the following message appears:

  ```
  *** M204.1039: FEATURE NOT LINKED IN
  ```

Calling the line editor indirectly

If you call the editor while entering a request, the word EDIT followed by a space must be the first five characters of a line. This line is not included in the

request. Note that this affects the use of EDIT in field names. The following lines do not generate a valid request:

```
FIND AND PRINT COUNT
TITLE = WUTHERING HEIGHTS
TYPEFACE = GOTHIC
EDIT = PENDING (space after EDIT)
```

However, the following line is valid as part of a request:

```
EDIT=PENDING (no space)
```

Uppercase and lowercase in procedures

You can enter lowercase characters in a procedure. However, Model 204 does not necessarily process these characters as lowercase. Unless the Model 204 command, *LOWER, was specified before entering the line editor, characters are translated automatically into uppercase when they are transmitted to Model 204. For more information about lowercase and uppercase usage, refer to the discussion of the *UPPER and *LOWER commands in the Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/*UPPER_command

http://m204wiki.rocketsoftware.com/index.php/*LOWER_command

Editing an existing request, procedure, or other text

The format of the EDIT command is the following:

Syntax `EDIT [n | name]`

where*:*

n or *name*	Indicates the number or the name of the request, procedure, or other text to edit.

Follow these guidelines:

- If you specify EDIT or EDIT 0, temporary procedure 0 is edited.
- If *n* is a negative number (–*n*), temporary procedure –*n* is edited. –*n* can also represent a previous request.
- If *name* is an alphanumeric string, the named permanent procedure is edited. The alphanumeric string can contain any letter, number, or other symbol, except:
  - Space
  - Comma
  - Semicolon

- Minus sign
- Equal sign
- Single quote
- Carriage return

The name cannot begin with a minus sign or a zero. The maximum length of a name is 255 characters.

For more information about permanent and temporary procedures, refer to the Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/EDIT_command

Defining new procedures

Use the following form of the EDIT command to indicate that a new name is being defined for the edited text:

Syntax `EDIT [`*`name1`* `[,`*`m`* `|,`*`name2`*`]]`

where:

name1	Represents either a temporary or a permanent procedure that you want to change.
m	Is the new temporary procedure number.
name2	Is the new permanent procedure name.

Follow these guidelines:

- If *m* is 0, temporary procedure 0 is defined.
- If *m* is a negative number (–m), temporary procedure or previous request –*m* is defined.
- If *name2* is an alphanumeric string, permanent procedure *name2* defined.

If *name2* already exists, the system verifies that you want to replace it by responding:

```
M204.1076: DO YOU REALLY WANT TO EDIT INTO EXISTING PROCE-
DURE?
```

Respond with Y (yes) or N (no). If your response is N, the existing procedure is not affected and the system returns to command level. If your response is Y, the system replaces the existing procedure (*name2*) with the new definition. This message is not displayed for temporary procedures; the editor simply stores the new text in place of an earlier temporary procedure or previous request.

After replacing a procedure, the new definition of procedure *name2* is an edited copy of procedure *name1*. The original procedure *name1* is not changed. This form of the EDIT command allows a procedure to be copied, with or without changes.

For example:

```
EDIT -2, TEST
```

In this example, you want to edit temporary procedure –2, and to save the edited result under the procedure name, TEST. Temporary procedure –2 is not altered by the editing.

When you create a new procedure with the editor, be aware that, if a procedure already exists under that name, you must satisfy the security rules for updating the procedure. If you do not have the appropriate privileges to update an existing permanent procedure, the system displays the following message:

```
M204.1176: CAN'T EDIT INTO name
```

where:

name	Identifies the existing procedure.

For more information about security and file privileges, refer to the Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/LOGCTL_command:_Modifying_file_entries_in_the_password_table

http://m204wiki.rocketsoftware.com/index.php/PRIVDEF_parameter

Setting the end-of-line character

Use the LINEND parameter to set the logical end-of-line character. The default value of LINEND is a semicolon. To reset the value to any single character, refer to the Model 204 documentation wiki:

http://m204wiki.rocketsoftware.com/index.php/LINEND_parameter

When you issue the PROCEDURE command to create a procedure, the current value of LINEND is stored as part of the procedure and becomes the procedure's physical end-of-line character. This character is carried through all editing.

If a procedure that was created with LINEND set to semicolon is edited by a user who has set LINEND to some other character, any semicolon inserted during editing is treated as an end-of-line character the next time the procedure is included.

If a procedure that was created with a LINEND semicolon is included by a user who has a LINEND of some other character, both characters are treated as

end-of-line characters (semicolon for physical lines and the user's LINEND for logical lines).

Using line editor commands

This section describes some key concepts involved in using the line editor, as well as general guidelines for using line editor commands.

This section describes how to enter, repeat, and cancel line editor commands. It also provides a summary of editor restrictions.

Pointer

All commands to the editor utilize a pointer that points to one character in the character string. Some of the commands position this pointer; others add, delete, or print characters based on its position. When the pointer is at any character other than the first character of a line, the system prints a pound sign (#) in front of that character when printing the line. In the examples used in this section, the pointer is represented as an underscore (_) where the pound sign would not appear at the terminal. The underscore itself does not appear at your terminal.

When you invoke the editor, the pointer is positioned initially at the beginning of the text that is being edited.

Character strings

A procedure can be considered not only as a series of statements and commands, but also as a string of characters, each line ending with a carriage return character, which is sometimes represented by a semicolon (;) as shown in the following examples. For example, you enter the following text:

```
BUICKS: FIND ALL RECORDS FOR WHICH
 MAKE = BUICK
```

This text appears to the editor as the following character string:

```
BUICKS: FIND ALL RECORDS FOR WHICH; MAKE = BUICK;
```

Entering line editor commands

Enter line editor commands one after another, with one or more commands on a line. Commands are performed one line at a time. You can separate commands with blanks, but this spacing is not required. The following two command strings are equivalent:

```
JL2T
J L 2T
```

Commands are summarized in Appendix B. A sample procedure using line editor commands is provided at the end of this section.

Repeating commands

When the editor encounters a command, it normally executes the command just once. However, if a sequence of editor commands is included in parentheses, the editor performs these commands as many times as you specify. The format of this repetition is as follows:

Syntax *n*(*commands*)*commands*

where:

n	Indicates the number of times that the parenthesized commands are to be executed and can be any positive number.

If you omit *n*, a value of 100 is assumed and the sequence is executed 100 times. The commands outside the parentheses are executed only once.

As with a regular execution of a search or replace command, the editor displays the following message, if the specified string cannot be found or if the editor cannot locate *n* occurrences of the string in the text:

```
M204.0529: NOT FOUND. WILL IGNORE REST OF LINE
```

If you do not specify *n* outside the parentheses, a value of 100 is assumed, and this message appears if there are fewer than 100 occurrences of the specified string.

You might want the editor to find and replace as many occurrences of a string as there are in the text. If you do not want to specify the precise number in the command, append a colon (:) to repeat S and R commands and to suppress the printing of the NOT FOUND error message.

Canceling line editor commands

The editor ignores the current command if the following phrase is entered in the first seven columns of an input line:

```
*CANCEL
```

If the current command already was completely entered, *CANCEL functions as a comment and has no effect on editor operations. For example:

```
Y.NEW LINE (command already completed)
*CANCEL
```

However, if the current command was not completely entered, *CANCEL causes it to abort:

```
R/END/AND (user has not entered final delimiter)
*CANCEL
```

The system responds with the following message:

```
M204.0528: REST OF LINE IGNORED. CANCELED
           OPERATION = R
```

Line editor restrictions

The following restrictions apply to line editor input and commands:

- Trailing blanks immediately preceding the carriage return are deleted from every input line.
- Any character after the last carriage return in a procedure is ignored outside the editor.
- Input and command strings in parentheses cannot be longer than 255 characters.
- Single lines read from a procedure cannot be longer than 255 characters outside the editor. The length can be further restricted by the setting of the LIBUFF parameter. There is no restriction on the length of a line that is being edited. Usually this condition is not detected until an attempt is made to use the edited procedure.
- The total number of characters inserted by a single editing session cannot exceed the value of an installation's PAGESZ parameter minus 40 (just over 6000 for most installations).

Controlling the pointer and displaying text

The line editor pointer and display commands perform the following functions:

- Position the pointer at the beginning or end of a procedure or other text, or at a specified position, character, line, or string
- Display the current line number or a portion of the text that is being edited

Pointer and display commands are text retrieval commands; they are not used to modify text.

This section describes the commands that perform the following tasks:

- Initializing the pointer
- Positioning the pointer at the end
- Spacing the pointer by character
- Spacing the pointer by line
- Displaying the current line
- Displaying text

Initializing the pointer (J)

The J command positions the pointer at the first character of the text. The format of the J command is as follows:

Syntax `J`

For example, a procedure consists of the logical lines:

```
NOW IS THE TIME
FOR
ALL GOOD MEN
TO COME
```

The J command positions the pointer at the first character of the text:

```
NOW IS THE TIME;FOR;ALL GOOD MEN;TO COME;
```

If a number precedes the J, *n*J, the pointer is moved *n* lines forward (if *n* is positive) or backward (if *n* is negative) within the text. The *nJ* command is equivalent to J*n*L. Refer to page 42.

Positioning the pointer at the end (Z)

The Z command positions the pointer immediately past the last character in the string (the carriage return). Use the Z command to add new lines to the end of the existing text.

The format of the Z command is as follows:

Syntax `Z`

For example, the pointer is positioned as shown in the following example, when you issue the Z command:

```
NOW IS THE TIME;FOR;ALL GOOD MEN;TO COME;_
```

Spacing the pointer by character (C)

The C command moves the pointer a specified number of characters to the left or right within the text. The format of the C command is as follows:

Syntax *n*C

where:

n	Is the number of characters that the pointer is moved.

The direction of movement is based upon the value of *n*, as follows:

- If *n* is negative, the pointer is moved *n* characters to the left.

- If *n* is positive, the pointer is moved *n* characters to the right.
- If *n* is omitted from the command, *n* = 1 is assumed and the pointer is moved one character to the right.

For example, the pointer is set to the beginning of the string:

```
NOW IS THE TIME;FOR;ALL GOOD MEN;TO COME;
```

Then either of the commands, C or 1C, moves the pointer as follows:

```
N#OW IS THE TIME;FOR;ALL GOOD MEN;TO COME;
```

From this position, the command 15C results in:

```
NOW IS THE TIME;FOR;ALL GOOD MEN;TO COME;
```

The editor treats the carriage return as any character in the character string. To move the pointer back again, issue the following command:

```
-3C
```

This command moves the pointer as follows:

```
NOW IS THE TI#ME;FOR;ALL GOOD MEN;TO COME;
```

Spacing the pointer by lines (L)

The L command moves the pointer one or more lines forward or backward within the text. The format of the L command is as follows:

Syntax *n*L

where:

n	Is the number of lines to be moved.

The direction of movement is based upon the value of *n*, as follows:

- If *n* is negative, the pointer is moved to the (*n*+1)th previous carriage return.
- If *n* is positive, the pointer is moved past the *n*th carriage return.
- If *n* is omitted from the command, *n* = 1 is assumed and the pointer is moved just past the next carriage return.

For example, suppose that you do not know where the pointer is positioned and want to point to the beginning of the third line of the text. To reset the pointer to the beginning of the text (J), and then move the pointer forward or down past two carriage returns to the beginning of the third line (2L), enter:

```
J2L
```

The following command now advances the pointer to the beginning of the next line:

```
L
```

Including zero as *n* results in a special use of the L command:

```
0L
```

This command moves the pointer to the beginning of the current line.

Displaying the current line number (=)

The = command returns the line number of the line at which the pointer is currently positioned. For example, you enter the following text:

```
AUTOS:  FIND ALL RECORDS FOR WHICH
          MAKE = BUICK AND COLOR = BLUE
        END FIND
        FOR #EACH RECORD IN AUTOS
          PRINT MAKE AND YEAR TO 45
        END FOR
```

If you issue the = command, the current line number appears as follows:

```
3
```

The number displayed is the number of lines past the first line in the text. If the pointer is currently positioned at the first line, the = command displays a 0.

If the pointer is at the line past the last line, the = command prints the number that the line would have if it existed. For the previous text shown, the sequence Z= positions the pointer past the last line and displays:

```
6
```

Displaying text (T)

The T command displays one or several lines of text or an entire procedure. The formats of the T command are as follows:

T	Displays one line
*n*T	Displays n lines
HT	Displays all of the text being edited
*n*HT	Displays *n* surrounding lines

where:

n	Can be any integer.

To display the entire line on which the pointer is currently positioned, enter:

```
T
```

For example, to display the first line of text, enter:

```
JT
```

Using the example from the previous section, the editor would respond with:

```
AUTOS: FIND ALL RECORDS FOR WHICH
```

You can use the *n*T form of the T command to display more than the current line; *n* can be any number greater than 1. For example:

```
J2T
```

This command causes the editor to print two lines: the current line and the one following:

```
AUTOS:  FIND ALL RECORDS FOR WHICH
          MAKE = BUICK AND COLOR = BLUE
```

If *n* is greater than the number of lines in the text, the editor displays all lines from the current line to the end.

The HT command displays all the text that is currently being edited. The *n*HT command displays the *n* lines surrounding the line at which the pointer is currently positioned. *n*HT displays a group of *n* consecutive lines; the line currently pointed at is near the middle of the group.

The T command does not change the position of the pointer. After the previous example is performed, the pointer is still at the beginning of the text.

To display the lines preceding the pointer without resetting the pointer, use a negative number in the command, as follows:

```
-nT
```

This form of the T command displays the previous *n* lines, up to the pointer in the current line. For example:

```
2LT-2T
```

This command moves the pointer down two lines, prints the third line, and then prints the previous two lines:

```
_           END FIND
AUTOS:      FIND ALL RECORDS FOR WHICH
              MAKE = BUICK AND COLOR = BLUE
```

Using the numbers 0 and 1 with T result in special action by the editor, as follows:

- 0T displays the part of the line before the pointer.
- 1T displays the part of the line after the pointer, beginning with the character at which the pointer is positioned. Thus, 1T is not equivalent to T.

With the pointer positioned as in the above example, the commands 13CT1T0T move the pointer 13 characters to the right and then display:

```
            END F#IND
IND
            END F#
```

The 1T and 0T commands determine the location of the pointer in the text.

Searching for and replacing text

You can search for text strings, and, if desired, replace them with a different text string.

These commands provide another method for moving the pointer around in a procedure. For more information on repositioning the pointer, see the section, “Controlling the pointer and displaying text” on page 40.

Searching for characters (S)

The S command searches the text by starting at the current position of the pointer and moving the pointer just past the next occurrence of a specified string. The format of the S command is as follows:

Syntax *SdXYZd*

where:

XYZ	Is the string to be found. The search string *XYZ* can be up to 255 characters in length and can contain carriage returns.
d	Is any character (except a blank) that does not occur in the string XYZ. The *d* character delimits the string being searched for and ends the command.

Suppose that the text to be edited consists of the following lines:

```
AUTOS:  FIND ALL RECORDS FOR WHICH
          MAKE = BUICK AND COLOR = BLUE
        END FIND
        FOR EACH RECORD IN AUTOS
          PRINT MAKE AND YEAR TO 45
        END FOR
```

You want to add another condition to the FIND statement. For example:

```
MAKE = BUICK OR DODGE AND COLOR = BLUE
```

First, position the pointer immediately past MAKE = BUICK. There are several ways to do this. One positioning method requires counting lines and characters to the desired position, and entering one of the following:

```
JL23C
```

or

```
J L 23C
```

These commands reset the pointer to the start of the text, move down one line, and then space 23 characters (including the eleven blanks at the beginning of the line and the blanks surrounding the equal sign) into the second line. An easier way to accomplish the same result, however, would be to use the S command in one of the following ways. Note that the J command is included in these examples to ensure that the search begins at the beginning of the text:

JS/MAKE = BUICK/	(Slashes are delimiters.)
JS.E = BUICK.	(Periods are delimiters.)
JS=BUICK=	(Equal signs are delimiters.)

All these commands result in the positioning of the pointer as follows:

```
MAKE = BUICK# AND COLOR = BLUE
```

When an S command is performed, the editor searches for the first occurrence of the character string enclosed by the delimiters. The search begins at the current pointer position.

Setting a delimiter

You can use any character as a delimiter. However, the delimiter character cannot be included in the string.

The previous examples illustrate the use of three different delimiters. In the third example, an equal sign (=) is used as the delimiter. This is acceptable because = is not a part of the character string (BUICK). = could not be used in either of the first two examples, because it is included as a character in the string. Thus, the following command is not valid:

```
JS=MAKE = BUICK =
```

In this example, the editor assumes that the first and second occurrences of = are the delimiters and searches for MAKE (that is, everything up to the second occurrence of the delimiter). The text after the second =, BUICK, is recognized as a possible repetition factor, not as a part of the string. The final = is treated as a request for the line number. Refer to “Displaying the current line number (=)” on page 43.

Searching for other occurrences (*n*S)

The previous examples searched for only the first occurrence of a character string; however, the S command is not limited to searching for only the first occurrence. You can search for the *n*th occurrence of the string *XYZ* by prefixing the S command with a counter number, *n*. The format is as follows:

Syntax *nSdXYZd*

where:

n	Is any positive number.

Consider the following example:

```
AUTOS:  FIND ALL RECORDS FOR WHICH
          MAKE = BUICK AND COLOR = BLUE
        END FIND
        FOR EACH RECORD IN AUTOS
          PRINT MAKE AND YEAR TO 45
        END FOR
```

Suppose that you want to position the pointer immediately past MAKE in the PRINT statement. This is the second time that MAKE occurs in the text, so you enter:

```
J2S/MAKE/
```

J is entered to ensure that the search starts at the beginning of the text. If the desired string does not occur between the point at which the search was begun and the end of the text, the following error message is displayed:

```
M204.0529: NOT FOUND. WILL IGNORE REST OF LINE
```

The pointer is reset to its position just before the unsuccessful invocation of the S command.

You enter the following:

```
J3S/MAKE/
```

The pointer remains positioned after the second occurrence of MAKE, because there is not a third occurrence of MAKE.

The S command does not find overlapping occurrences of the specified character string. Consider the situation in which the editor searches for occurrences of AA in the string AAAA. The S command finds AAAA and AAAA, but not AAAA.

Replacing a character string (R)

The R command replaces one or more occurrences of a specified character string with another character string. The format of the R command is as follows:

*n*R*d*XYZ*d*ABC*d*

where:

n	Is a repetition factor.
XYZ	Is the string to be replaced.
ABC	Is the new string.
d	Is the delimiter.

Starting at the current pointer location, R searches for the next *n* occurrences of *XYZ* and replaces all those occurrences with *ABC*. If *n* is omitted from the command, *n* = 1 is assumed, and only the first occurrence of *XYZ* is replaced. Both of the strings, *XYZ* and *ABC*, can be up to 255 characters in length and can contain carriage returns. The delimiter, *d*, can be any character (except a blank) that does not occur in the strings.

After the R command is executed, the pointer is set to just past the last character of the last substitution of *ABC*. If the string to be replaced cannot be found in the text, the following error message is displayed:

```
M204.0529: NOT FOUND. WILL IGNORE REST OF LINE
```

The pointer remains where it was. This message also appears if you include *n* in the command, and the editor finds at least one, but not *n*, occurrence of the string to be replaced. In this case, after execution of the R command, the pointer is left positioned just past the last successful substitution.

All occurrences of *XYZ* that are found by the editor are properly replaced by *ABC*. For example, you enter:

```
3R 'FINE'FIND'
```

The editor finds only two occurrences of FINE in the text. The following events occur:

- Two occurrences of FINE are replaced by FIND.
- `NOT FOUND` message is displayed.
- Pointer is positioned just past the D of the second substituted FIND.

In the following text, the word COLOR is incorrectly typed as COLPR:

```
BUICKS:  FIND ALL RECORDS FOR WHICH
           MODEL = BUICK
         END FIND
```

```
FOR EACH RECORD IN BUICKS
  PRINT MODEL AND COLPR
END FOR
```

To correct the mistake, enter:

```
JR.COLPR.COLOR.
```

This command searches the text from the beginning for the string COLPR and, upon locating it, replaces it with the string COLOR. The resulting line is:

```
PRINT MODEL AND COLOR
```

The replacement string need not have the same length as the original string. The following command successfully changes both occurrences of MODEL to MAKE:

```
J2R/MODEL/MAKE/:
```

The text then consists of:

```
BUICKS:  FIND ALL RECORDS FOR WHICH
           MAKE = BUICK
         END FIND
         FOR EACH RECORD IN BUICKS
           PRINT MAKE AND COLOR
         END FOR
```

Editing text

You can replace, insert, and delete lines, characters, and character strings by using the change commands. Most of these commands require that the pointer be properly positioned at the text to be changed before you issue the change commands. Several of the commands explicitly reposition the pointer after making the specified change.

In addition to editing characters, lines and strings, you can use change commands to combine several existing procedures into a new procedure.

Replacing a line (X)

The X command replaces the entire current line with the line that you enter following the X. The format of the X command is as follows:

Syntax *Xline*

The command sets the pointer to the beginning of the next line. Consider the following sample text:

```
AUTOS:   FIND ALL RECORDS FOR WHICH
           MAKE = BUICK AND COLOR = BLUE
```

```
          END FIND
          FOR EACH RECORD IN AUTOS
            PRINT MAKE AND YEAR TO 45
          END FOR
```

To replace the second line of this example, position and check the pointer as follows:

```
JLT

_           MAKE = BUICK AND COLOR = BLUE
```

Enter:

```
X           YEAR = 82
```

This command replaces the entire original line with:

```
            YEAR = 82
```

You can enter the entire command string on a single line as:

```
JLTX          YEAR = 82
```

Note: To have these spaces appear in the line, enter spaces before a string.

The text now consists of the following lines:

```
AUTOS:    FIND ALL RECORDS FOR WHICH
            YEAR = 82
          END FIND
          FOR EACH RECORD IN AUTOS
            PRINT MAKE AND YEAR TO 45
          END FOR
```

Inserting a line (Y)

The Y command inserts a line between existing lines in the text. The format of the Y command is as follows:

Syntax `Yline`

The line that you enter following the Y is inserted after the previous carriage return (that is, just before the line at which the pointer is positioned). The pointer is positioned at the beginning of the line after the inserted line.

For example, you want to add the following text to the FIND statement of the original example:

```
YEAR = 82
```

You can insert the new condition either before or after the existing line by entering:

```
J2LY YEAR = 82
-2L3T
```

The above command places the new condition after the existing line. The T command displays the following lines:

```
    MAKE = BUICK AND COLOR = BLUE
    YEAR = 82
END FIND
```

You can include a line number before the Y to insert text before a line other than the current line. The effect of the command *n*Y is identical to that of *n*LY. *n* defaults to zero if it is not present in the command.

Deleting a line (K)

The K command deletes part of a line, or one or more lines. The formats of the K command are as follows:

This format...	Performs this action...
K	Deletes the entire current line.
1K	Deletes from the current pointer location to the end of the line, including the carriage return.
0K	Deletes from the beginning of the line up to the current character. The pointer is not moved and is left positioned at the character immediately after the deleted string.
*n*K	Deletes from the current pointer location to the end of the line, including the carriage return, and the *n*–1 following lines.
-*n*K	Deletes from the beginning of the line up to the current character, and the preceding *n* lines as well.
HK	Deletes all the text that is being edited.
*n*HK	Deletes the group of *n* lines surrounding the current line. If *n* is even, *n*HK deletes *n*/2 lines before the current line, the current line and (*n*/2) –1 lines after the current line. If *n* is odd, *n*HK deletes (*n*–1)/2 lines before the current line, the current line, and (*n*–1)/2 lines after the current line. The text deleted by this method is the same text that would be printed by *n*HT.

For example, the following commands return the sample text to its original form and display:

```
J2LK-1L2T
```

The result of these commands is:

```
_             MAKE = BUICK AND COLOR = BLUE
            END FIND
```

Inserting a character (I)

The I command inserts a specified character or string of characters just before the current pointer location. The format of the I command is as follows:

Syntax `I`*`dXYZd`*

where:

XYZ	Is the character string to be inserted. The inserted string, *XYZ*, can be as long as 255 characters in length and can contain carriage returns.
d	Is any character (except a blank) that does not occur in the string *XYZ*. The *d* character delimits the string that is being inserted and ends the command. Its function is identical to that of the delimiter in the S command.

After the I command is executed, the pointer is reset to just past the last character of the inserted string.

Suppose that you want to change the following statement:

```
PRINT 'EMPLOYEE ADVANCES TO DATE'
```

to

```
PRINT 'EMPLOYEE CASH ADVANCES TO DATE'
```

To make this correction, issue the command:

```
S/EMPLOYEE/I/ CASH/
```

The I command is not limited to adding characters within a line; you can also use it to insert any number of lines. Consider the following lines:

```
AUTOS:   FIND ALL RECORDS FOR WHICH
           MAKE = BUICK
         END FIND
         FOR EACH RECORD IN AUTOS
```

You can add additional retrieval conditions to the FIND statement. For example, you want to add:

```
OR MAKE = DODGE
AND YEAR = 82
AND COLOR = BLUE
```

Enter the following command string:

```
S/BUICK/I/ OR DODGE
           YEAR = 82
           COLOR = BLUE/J99T
```

This command string results in:

```
AUTOS:  FIND ALL RECORDS FOR WHICH
          MAKE = BUICK OR DODGE
          YEAR = 82
          COLOR = BLUE
        END FIND
        FOR EACH RECORD IN AUTOS
```

Note: The action of the S, I, and R commands is controlled by the delimiter, not by the carriage return. Thus, when the editor encounters I/ (or I followed by any delimiter), it treats carriage returns as part of the character string until the second delimiter is encountered.

Deleting a character (D)

The D command deletes the character at which the pointer is currently positioned and resets the pointer to the next character. The format of the D command is as follows:

Syntax `D`

The command also has the following forms:

*n*D
-*n*D

These formats delete the next (*n*) or previous (–*n*) characters in the text. Thus, to remove the second retrieval condition (that is, AND COLOR = BLUE) from the previous example, enter the following commands:

```
23C17DJ99T
```

Combining procedures

The I command combines several existing procedures, saved requests, or other text into a single new procedure. The format of this command is as follows:

Syntax `[n |'name'] I`

where:

n	Can be zero or a negative integer.

name	Is the name of a procedure, enclosed in single quotes.

The editor inserts temporary procedure *–n* or procedure *name* just before the character at which the pointer is currently positioned. The procedure that is being included remains unchanged. A carriage return included in the procedure name is ignored, but any other invalid character cause an error to occur.

In the following example, three procedures are defined, and then combined and defined as a new procedure. First, a temporary procedure is defined:

```
 PROCEDURE -2

*** M204.1144: DEFINE PROCEDURE

 BEGIN
 CATEGORY.A: FIND ALL RECORDS FOR WHICH
                 TOTAL PREMIUM IS GREATER THAN 300
                 AGENT = GOODRICH OR BATEMAN
                 CITY = SACRAMENTO
             END FIND END PROCEDURE

*** M204.1146: PROCEDURE DEFINITION ENDED
```

Next, the two procedures that are to be permanently retained are defined:

```
 PROCEDURE SUBSET

 *** M204.1144: DEFINE PROCEDURE

 COUNTER:    COUNT RECORDS IN CATEGORY.A
             PRINT COUNT IN COUNTER
 END PROCEDURE

 *** M204.1146: PROCEDURE DEFINITION ENDED



 PROCEDURE AGENTPRINT
 *** M204.1144: DEFINE PROCEDURE
             FOR EACH RECORD IN CATEGORY.A
             PRINT FULLNAME AND AGENT AT 30
END
END PROCEDURE
 *** M204.1146: PROCEDURE DEFINITION ENDED
```

Finally, the procedures are combined by means of the editor. First, enter:

```
EDIT -2, SACRAMENTO
```

This command invokes the line editor and indicates that editing is to be performed on temporary procedure –2 and that the result is to be stored as procedure SACRAMENTO.

The following command sets the pointer to the end of procedure –2, then attaches procedure SUBSET, and leaves the pointer set just past the new end:

```
Z'SUBSET'I
```

```
Use the following command to attach procedure AGENTPRINT:
```

```
'AGENTPRINT'I
```

The END command terminates line editor operations and stores the result in procedure SACRAMENTO:

```
END

*** M204.0542: EDIT COMPLETE - END
```

Use the following command to display the combined procedure:

```
DISPLAY SACRAMENTO

BEGIN
CATEGORY.A:  FIND ALL RECORDS FOR WHICH
                TOTAL PREMIUM IS GREATER THAN 300
                AGENT = GOODRICH OR BATEMAN
                CITY = SACRAMENTO
COUNTER:     COUNT RECORDS IN CATEGORY.A
             PRINT COUNT IN COUNTER
             FOR EACH RECORD IN CATEGORY.A
                PRINT FULLNAME AND AGENT AT 30
END
```

Leaving the line editor

Use the following commands to exit from the line editor:

END	Saves the results of the editing and returns to command level.
GO	Saves the results of the editing and performs an INCLUDE command. For more information about the INCLUDE command, refer to the Model 204 documentation wiki: http://m204wiki.rocketsoftware.com/index.php/INCLUDE_command
QUIT	Stops without modifying any text and returns to command level.

END and GO commands

The following commands cause the editor to save the results of the editing:

Syntax `[m |'name'] END`

```
[m |'name'] GO
```

where:

m	Can be 0 or a negative number.
name	Is an alphanumeric string (for example, TEST).
END	Returns control to command level, which implies that the next line entered must be a system control command.
GO	Causes the system to take the next line of input from the edited text.

The system responds with either:

```
M204.0542: EDIT COMPLETE - END
```

or

```
M204.0542: EDIT COMPLETE - GO
```

If *m* or *name* is included in END or GO, the effect is the same as calling the editor with the following command:

Syntax `EDIT [name1 [,m |,name2]]`

In both the END and GO commands, *m* can be zero or a negative integer, and *name* is the name of a procedure. The editor saves temporary procedure *m* or procedure *name* before it exits. A carriage return included in the procedure *name* is ignored, but any other invalid character causes an error to occur.

If procedure name already exists, the system displays the confirming message shown for EDIT in the preceding discussion.

After an END or GO command that specifies a procedure name or number is executed, *m* or *name* is an edited copy of the procedure that was just edited. The original procedure is not changed.

The *m* or *name* argument included in END or GO overrides the *m* or *name* included in the EDIT command. This is useful if a user who has no update privileges tries to edit a permanent procedure. In this case, entering END or GO causes the following message to be displayed:

```
M204.0539: CAN'T EDIT INTO PROCEDURE
```

The user remains in the editor. The user can then save the edited copy as a temporary procedure or under a new name. The name is shown in the message only if a name was specified in the END or GO command.

Line editor example

The following example illustrates an edit session using the line editor. This example uses the CLIENTS file:

```
EDIT POLICY

*** M204.0526: EDITING INTO POLICY

HT
BEGIN
GET.POL: FIND ALL RECORDS WHICH
            POLICY NO = 100035
            STATE - OHIO
          END FIND
          FOR EACH RECORD IM GET.POL.
            PRINT FULLNAME AND DATE OF BIRTH
```

The procedure has various errors in it, including typing errors, omissions, and incorrect specifications. Correct the following errors:

- Omission of the word FOR in the first statement
- Keying of – instead of = on the third line
- Misspelling of IN (as IM) on the fifth line
- Inclusion of a period after GET.POL

Set the pointer to the beginning of the procedure using the following command:

J

Correct the errors as follows:

32CT Space 32 characters to position the pointer and display the line.

```
GET.POL: FIND ALL RECORDS# WHICH
```

I/ FOR/ Insert FOR before WHICH.

R/-/=/ Replace occurrence of – with =.

R/IM/IN/ Replace occurrence of IM with IN.

8CD Space eight characters and delete the period.

-5T Display the previous four lines plus the current line (up to the pointer) for verification.

```
GET.POL:   FIND ALL RECORDS FOR WHICH
             POLICY NO = 100035
             STATE = OHIO
           END FIND
           FOR EACH RECORD IN GET.POL#
```

Next, insert additional retrieval conditions to:

- Change STATE = OHIO to STATE = OHIO OR NEW YORK.
- Insert SEX = M.

JS/OHIO/ Search for occurrence of OHIO.

I/ OR NEW YORK/ Insert words after OHIO.

LY SEX = M Skip to next line and insert new condition.

J5T Display the first five lines of the procedure for verification.

```
BEGIN
GET.POL:   FIND ALL RECORDS FOR WHICH
             POLICY NO = 100035
             STATE = OHIO OR NEW YORK
             SEX = M
```

Make additional changes to the procedure to:

- Print all key information from the record, not simply FULLNAME and DATE OF BIRTH.
- Skip between printed lines.

HT Display the entire procedure for verification.

```
BEGIN
GET.POL:   FIND ALL RECORDS FOR WHICH
             POLICY NO = 100035
             SEX = M
             STATE = OHIO OR NEW YORK
           END FIND
           FOR EACH RECORD IN GET.POL
              PRINT FULLNAME AND DATE OF BIRTH
```

J7LK Skip to and delete the seventh line.

Next, insert three new lines:

```
Y             PRINT ALL INFORMATION
Y             SKIP 1 LINE
YEND
```

Finally, insert the appropriate commands to open and close the file against which this retrieval is processed. OPEN and CLOSE commands are inserted, respectively, at the beginning and end of the current text:

JYOPEN CLIENTS Start at beginning and insert line.

```
ZYCHOSE CLIENTS                Skip to end and insert line.
-LXCLOSE CLIENTS               Replace incorrectly entered line.
HT                             Display all text for final time.

OPEN CLIENTS
BEGIN
GET.POL:   FIND ALL RECORDS FOR WHICH
              POLICY NO = 100035
              SEX = M
              STATE = OHIO OR NEW YORK
           END FIND
           FOR EACH RECORD IN GET.POL
              PRINT ALL INFORMATION
              SKIP 1 LINE
END
CLOSE CLIENTS

END                            Exit from editor, saving edited procedure

*** M204.0542: EDIT COMPLETE - END
```

Enter the final END statement after the procedure is printed to exit from the editor and to save the procedure. It is not part of the procedure.

A

Full-Screen Editor Command and Abbreviation Summary

Full-screen editor commands and abbreviations

Table A-1. Full-screen editor commands

Command	Abbreviation	Meaning
[+]*n*		Scrolls forward *n* lines in a procedure.
-*n*		Scrolls backward *n* lines in a procedure.
=		Repeats the most recently executed command.
[SET] ARBCHAR		Sets the arbitrary (wildcard) character for a search string. Default: OFF
BACKWARD[*n*]	BA[*n*]	Scrolls backward n screens in a procedure.
BOTTOM	B	Scrolls to the last line of a procedure.
CLEAR		Clears the prefix area, aborts pending prefix commands, and cancels any specified targets.
EDIT		Invokes the full-screen editor.
END		Exits from the full-screen editor and saves the edited procedure.
[SET] FILL		Sets the fill character. Default: NULL
FORWARD[*n*]	FO[*n*]	Scrolls forward *n* screens in a procedure.
GET		Inserts a saved procedure in the procedure being edited.
GO		Exits from the full-screen editor, saves the edited procedure, and includes the procedure.

Table A-1. Full-screen editor commands

Command	Abbreviation	Meaning
[SET] LINEND		Sets the line-end character.
[-] LOCATE	[-] LO	Locates a specified string in a procedure.
[SET] PREFIX		Controls the position of the prefix area on the screen. Default: LEFT
QUIT		Exits from the full-screen editor and does not save the edited procedure.
[SET] REPEAT		Sets the repeat character for search strings. Default: OFF
REPLACE	R	Locates and replaces a specified string in a procedure.
SAVE		Saves the edited procedure.
[SET] SCALE		Controls the display of the column scale on the screen.
TOP	T	Scrolls to the first line of a procedure.

B

Line-Editor Command Summary

Entry and exit commands

Table B-1. Entry and exit commands

Command	Meaning
EDIT [*n* | *name1* [,*m* |,*name2*]]	Edits procedure *n* or *name1*, defining *m* or *name2* as the result.
[*m* |'*name*'] END	Exits the line editor, saving the editing in procedure *m* rename'.
[*m* |'*name*'] GO	Exits the line editor, saving the editing in procedure *m* or '*name*' and executing an INCLUDE command.
QUIT	Exits the line editor without saving the editing.
*CANCEL (in columns 1-7)	Cancels the current editor command, if not completed.

Pointer and display commands

Table B-2. Pointer and display commands

Command	Meaning
J	Positions the pointer at the first character of the text.
Z	Positions the pointer just past the last character of the text.
C	Moves the pointer one character to the right within the text.
*n*C	Moves the pointer *n* characters to the right.
-*n*C	Moves the pointer *n* characters to the left.
L	Moves the pointer to the beginning of the next line.
*n*L	Moves the pointer to the beginning of the *n*th line.
-*n*L	Moves the pointer to the beginning of the *n*th previous line.

Table B-2. Pointer and display commands (Continued)

Command	Meaning
0L	Moves the pointer to the beginning of the current line.
S*dXXYd*	Searches for the next occurrence of *XYZ* (delimited by *d*) in the text and positions the pointer just past it.
*n*S*dXYZd*	Searches for the *n*th occurrence of *XYZ* in the text and positions the pointer just past it.
–	Displays the number of the line at which the pointer is currently positioned.
T	Displays the line in which the pointer is currently positioned.
*n*T	Displays *n* lines, beginning with the current line.
-*n*T	Displays the previous *n* lines and the current line up to the pointer.
0T	Displays the part of the line before the pointer.
1T	Displays the part of the line after the pointer, beginning with the current character.
HT	Displays all the text that is being edited.
*n*HT	Displays the *n* lines surrounding the current line.

Change commands

Table B-3. change commands

Command	Meaning
X*line*	Replaces the current line with *line*.
Y*line*	Inserts *line* just before the current line.
K	Deletes the current line.
*n*K	Deletes *n* lines from the current pointer location through the *n*-1 subsequent lines.
-*n*K	Deletes the previous *n* lines through the current pointer location.
0K	Deletes the part of the line before the pointer.
1K	Deletes the part of the line after the pointer beginning with the current character.
HK	Deletes all the text being edited.
*n*HK	Deletes the *n* lines surrounding the current line.
D	Deletes the current character.
*n*D	Deletes the next *n* characters, including the current character.
-*n*D	Deletes the previous *n* characters.
I*dXYZd*	Inserts string *XYZ* just before the pointer and positions the pointer just past the inserted string.
R*dXYZdABCd*	Replaces the next occurrence of the string *XYZ* with *ABC* and positions the pointer just past the replaced string.
*n*R*dXYZdABCd*	Replaces the next *n* occurrences of the string *XYZ* with *ABC* and positions the pointer just past the last replacement.
(*commands*) commands	Repeats the parenthesized commands 100 times.
n (*commands*) commands	Repeats the parenthesized commands *n* times.
{*n* | '*name*'} I	Inserts procedure *n* or '*name*' just before the pointer location.

Index

Symbols

A

B

C

D

E

F

G

T

U

W

X

Y

Z