Rocket Model 204 Language Support Summary

Version 7 Release 5.0

September 2014
204–75–LANG-01

Notices

Edition

Publication date: September 2014
Book number: 204–75–LANG-01
Product version:

Copyright

Trademarks

Examples

License agreement

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

Audience

This guide is for anyone who needs to understand the facilities that perform language-specific processing for Model 204 data display, sequencing, and collating using single-byte character sets other than U.S. English or Japanese double-byte character set (DBCS) support.

Adding other languages

Adding a language to those already developed for Model 204 language processing is a cooperative venture between a customer and Rocket Software. If you are interested, please consult your sales representative. You will be asked to become a beta site for testing that language, which includes answering a questionnaire to analyze your needs and the character set used to display the language. You will work closely with Rocket Software development engineers and Technical Support.

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/).

Language support documentation

For a thorough discussion of the decisions surrounding language support, consult the following documents:

- *Canadian Alphanumeric Ordering Standard for Character Sets of CSA Standard CAN/CSA-Z243.4*, Canadian Standards Assoc., Rexdale (Toronto), Ontario, Canada, 1992.
- *The Unicode Standard: Worldwide Character Encoding, Version 1.0, Volume 1*, The Unicode Consortium, Addison-Wesley, Reading, MA, 1991.

Documentation conventions

This manual 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 |` <u>`NOTRUST`</u>	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.
EDIT	Partial bolding indicates a usable abbreviation, such as E for EDIT in this example.

1

Model 204 Language Support Feature

Overview of language support

Model 204 contains a language support feature for customers who sort and display Model 204 data using single-byte character sets other than U.S. English or Japanese double-byte character set (DBCS). This feature is included in the SOUL (User Language), HLI, and SQL interfaces. This chapter describes the facilities that perform language-specific processing.

Language support in computer data storage means being able to receive, store, and redisplay differing character sets and devising algorithms to handle the correct sorting procedures. Worldwide use of the computer to store, transmit, share, and compare data exposed the need to:

- Analyze the character sets used by written languages to determine which characters are shared and which characters are unique to a character set.
- Respect the collating sequence or order of precedence rules used by a written language.

Language support terminology

A ***character set*** is a set of symbols or marks used in a writing system, such as a letter of the alphabet. Character sets differ in the number of characters, the specific characters included, and their collating sequence.

Once a character set is identified, the next task is handling the collating sequence. ***Collating sequence*** is the sequence in which characters are ordered for sorting, merging, and comparing. Specifically it is the order assigned to the characters of a character set (in computers, for example, ASCII, U.S. English, and EBCDIC) used for sequencing purposes. Usage determines the correct collating sequence for each writing system. The commonplace examples of collating sequences are telephone directories and dictionaries.

See the section “Language support documentation” on page vii for a list of books pertaining to language support for character sets and collating sequences.

Collating sequence support

The language support feature in Model 204 currently sorts using the expected collating sequence for U.S. English and limited support of Japanese.

NLANG, the language support module

Model 204 modules are linked with a set of language support tables in the NLANG module that define written languages. A Model 204 supported language consists of translation tables and flag tables containing information about:

- Alphabetic characters, lowercase to uppercase
- Alphabetic characters, uppercase to lowercase
- LANGSORT tables
- Pattern matcher
- Characters you can enter at the keyboard
- Characters you can display on the terminal
- ASCII to EBCDIC translation

Supported languages

After installing Model 204, you can select one of five variations of the internal language table. The LANGUSER and LANGFILE parameter settings you select sets the terminal and print capabilities. The NLANG module contains internal language tables for the following languages:

- Cyrillic
- French Canadian
- Japanese
- Turkish

- US English

The internal language table provides the same input and output translation tables, uppercase or lowercase translation, and $ALPHA support as the coordinating Model 204 parameters, LANGUSER and LANGFILE, but does not determine collating sequences for sorting or B-tree indexes.

Language support parameters

After installation set the correct LANGFILE and LANGUSER parameter options to support applications for your language requirements. The value of the LANGFILE and LANGUSER parameters determine which internal language table in NLANG that Model 204 consults for collating sequence, character storage, and uppercase or lowercase translation.

IBM code pages

IBM assigns a code page number to correspond to various sets of characters. Each IBM code page assigns a particular set of character shapes to a corresponding binary code. Model 204 depends on the binary code definition in the IBM code page to handle language support.

Table 1-1 lists the Model 204-supported character sets and designated IBM code pages.

Table 1-1. Character sets supported in Model 204

Written language	Parameter value	Refers to IBM code page
Cyrillic	CYRILLIC	880
French Canadian	FRENCHC	037
Japanese	JAPAN	290
Turkish	TURKISH	1026
US English	US (the default)	1047

LANGFILE: Choosing a character set definition for a file

Class	FPARMS
Default	US, meaning U.S. English
Setting	During file creation or resettable by file manager
Meaning	Use the LANGFILE parameter to specify the language for file processing operations such as the ordering of data and processing LIKE and LANGLIKE patterns. The LANGFILE parameter determines the valid character set in a file.

The value of LANGFILE must be one of the following, listed in Table 1-2.

Table 1-2. Valid character sets

Written language	Model 204 LANGFILE value
Cyrillic	CYRILLIC
French Canadian	FRENCHC
Japanese	JAPAN
Turkish	TURKISH
US English	US (the default)

Note: You cannot specify a LANGFILE parameter setting other than US for sorted files (FILEORG X'01' setting).

LANGUSER: Setting the language definition of a user thread

Class:	USER
Default:	US, meaning U.S. English
Setting:	On the user's parameter line, resettable

Meaning: Use the LANGUSER parameter to specify the language that is in use by the thread's I/O device. Different terminals in the same Model 204 run can use different languages. HLI or SQL threads can use different languages from each other and from SOUL or terminal threads.

The value of LANGUSER must be one listed in Table 1-3:

Table 1-3. Valid languages for a thread's I/O device

Written language	Model 204 LANGUSER value
Cyrillic	CYRILLIC
French Canadian	FRENCHC
Japanese	JAPAN
Turkish	TURKISH
US English	US (the default)

Data Management Language enhancements

This section describes language support enhancements to Data Management Language.

SQL Server

SQL Server ordering operations for an SQL table use the collating sequence specified by the file’s LANGFILE parameter. Model 204 SQL does not permit joins across files that do not have the same LANGFILE parameter settings.

SQL language support requires an additional four bytes in QTBL per compiled query. You can set QTBL on the User 0 line or with the UTABLE command.

Statements that support language-specific ordering

The following SOUL statements and their corresponding HLI calls provide language-specific ordering:

- FIND (various inequality operators such as index and direct search)
- FOR EACH RECORD IN ORDER BY (including FROM and TO clauses)
- FOR EACH VALUE IN ORDER
- FOR EACH VALUE (in group context)
- SORT RECORDS/RECORD KEYS
- SORT VALUES
- Pattern matcher (LIKE or LANGLIKE clause) range specifications

Note: Sorted file operations (with FILEORG = X‘01’) are not supported.

Pattern matching using the LANGLIKE operator

The SOUL operator LANGLIKE supports parsing and evaluation of patterns according to the tables provided with the LANGUSER and LANGFILE parameters.

The LANGLIKE syntax is the same as LIKE syntax. See the Rocket Model 204 documentation wiki information on value loops for more details:

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

- The LIKE operator employs U.S. English for parsing the pattern and the value of LANGFILE for evaluating the pattern.
- The LANGLIKE operator uses the value of LANGUSER for parsing the pattern and the value of LANGFILE for evaluating the pattern.

The parsing language, LANGUSER, is used for checking the syntax of the pattern and for determining the value of:

- Special pattern escape character
- Hexadecimal character
- Alphabetic character

The evaluation language, LANGFILE, is used to match the pattern against the data. In particular, if a range of characters is defined in the pattern, then the collating sequence is determined by the evaluation language, LANGFILE.

Syntax

The format of the FIND statement used to perform pattern matching is:

```
FIND [ALL] RECORDS {FOR WHICH | WITH} fieldname

     IS [NOT] LANGLIKE 'pattern'
```

Where

The LANGLIKE keyword indicates that *pattern* is the set of characters to match, using LANGUSER and LANGFILE as previously described.

The *pattern* argument must be enclosed in quotation marks. The characters that you can use in a pattern and the methods of optimizing a pattern retrieval are described in the Rocket Model 204 documentation wiki information on record loops:

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

SOUL $functions for language support

The $functions in Table 1-4 include language-specific processing capabilities.

Table 1-4. $Functions for language-specific processing

$function	Description
$ALPHA	Verifies that a string is composed of only characters that are valid in the specified or default language.
$ALPHNUM	Verifies that a string is composed of only characters and digits 0 through 9, which are valid in the specified or default language.
$CHKPAT	Verifies the syntax of a pattern.
$LANGSPC	Returns a string containing the language-specific hexadecimal value of a special character on a particular terminal.
$LANGSRT	Transforms a string into a language-specific sequence value.
$LANGUST	Restores a transformed string back to its original value.
$LIKE	Controls parsing and evaluation languages used in pattern matching.
$LOWCASE	Translates an uppercase case or mixed-case string into a lowercase string.
$UPCASE	Translates a lowercase or mixed-case string into an uppercase string.

The following describes the syntax and usage for these $functions.

$ALPHA

The $ALPHA function verifies that a string is composed of only characters that are valid in the specified or default language.

- If the condition is true, 1 is returned.
- Otherwise, 0 is returned, because:
  - String contains digits, spaces, or punctuation marks
  - String is null
  - U.S. English characters in a string have accent marks

Syntax

```
$ALPHA(string[,'language'])
```

Where

The *string* argument represents the set of characters to verify, which must be one of the following:

- A literal enclosed in quotation marks.
- A %variable.
- A field name without quotation marks. In this case, the function call must be embedded in a FOR EACH RECORD loop where the current value of the field is verified.

The optional *language* argument specifies the language to use. The *language* argument is handled as follows:

- You can enter a literal enclosed in quotation marks or a %variable containing a valid language name. If the value you enter is not a supported language, the request is canceled with the following error message.

  ```
  M204.2340: INVALID LANGUAGE ARGUMENT: ‘language’ FOR $FUNC-
  TION: ALPHA
  ```

  See “LANGUSER: Setting the language definition of a user thread” on page 4 for the valid values.

- When you omit the *language* argument, Model 204 performs the validation in U.S. English, even if the value of the LANGUSER parameter is not US, and lowercase characters are not recognized.
- An asterisk enclosed in quotation marks (‘*’) instructs Model 204 to use the value of the LANGUSER parameter.

Examples

The following table has examples of the *string* and *language* arguments as literals:

Function code…	Returns…
`$ALPHA(‘JOHN’,‘US’)`	1

Function code…	Returns…
`$ALPHA('MÂCON','FRENCHC')`	1
`$ALPHA('MÂCON','US')`	0
`$ALPHA('JOHN SMITH','US')`	0
`$ALPHA('ÎLE D'ORLÉANS','FRENCHC')`	0
`$ALPHA('12A','US')`	0
`$ALPHA('12A','FRENCHC')`	0

In the following code example, the request sorts and prints the names of agents whose name contains nonalphabetic characters. The quoted asterisk in the $ALPHA call causes Model 204 to verify the contents of the field AGENT against whatever language is indicated by the value of the LANGUSER parameter.

```
RESET LANGUSER 'FRENCHC'
BEGIN
POL.HOLDERS:   FIND ALL RECORDS FOR WHICH
                  RECTYPE = POLICYHOLDER
               END FIND
               FOR EACH RECORD IN POL.HOLDERS
                  IF NOT $ALPHA(AGENT,'*') THEN
                     PLACE RECORD ON LIST BADNAME
                  END IF
               END FOR
ORDERED.LIST: SORT RECORDS ON LIST BADNAME BY AGENT
               FOR EACH RECORD IN ORDERED.LIST
                  PRINT AGENT
               END FOR
END
```

$ALPHNUM

The $ALPHNUM function verifies that a string is composed of only characters and digits 0 through 9 that are valid in the specified or default language.

- If the condition is true, 1 is returned.
- Otherwise, 0 is returned because:
  - String contains spaces or punctuation marks
  - String is null.

Syntax `$ALPHNUM(`*`string`*`[,`*`language`*`])`

Where The *string* argument represents the characters to verify, which must be one of the following:

- A string of characters enclosed in quotation marks.
- A %variable.
- A field name that is not enclosed in quotation marks. In this case, the function call must be embedded in a FOR EACH RECORD loop where the current value of the field is verified.

The optional *language* argument specifies the language to use. The *language* argument is handled as follows:

- When the *language* argument is omitted, Model 204 performs the validation in U.S. English, even if the value of the LANGUSER parameter is not US, and lowercase characters are not recognized.
- An asterisk enclosed in quotation marks (‘*’) instructs Model 204 to use the value of the LANGUSER parameter.
- You can enter the name of a valid language enclosed in quotation marks or a %variable containing a valid language. If the value you enter is not supported, the request is canceled with an error message. See “LANGUSER: Setting the language definition of a user thread” on page 4 for valid values.

Examples

The following examples illustrate the *string* and *language* arguments as literals enclosed in quotation marks:

Function code…	**Returns…**
`$ALPHNUM(‘JOHN’,‘US’)`	1
`$ALPHNUM(‘MÂCON’,‘FRENCHC’)`	1
`$ALPHNUM(‘MÂCON’,‘US’)`	0
`$ALPHNUM(‘JOHN SMITH’,‘US’)`	0
`$ALPHNUM(‘ÎLE D’ORLÉANS’,‘FRENCHC’)`	0
`$ALPHNUM(‘12A’,‘US’)`	1
`$ALPHNUM(‘12A’,‘FRENCHC’)`	1

In the following example, the request sorts by name and processes records whose designated field value does not meet the $ALPHNUM criteria. The second argument in the $ALPHNUM call instructs Model 204 to use French Canadian to perform the validation:

```
BEGIN
           %SEARCH = $READ(‘ENTER FIELD NAME’)
   FIND.RECS: FIND ALL RECORDS FOR WHICH
              RECTYPE = POLICYHOLDER
           END FIND
           PLACE RECORDS IN FIND.RECS ON LIST BAD
```

```
                FOR EACH RECORD IN FIND.RECS
                   IF $ALPHNUM(%%SEARCH,'FRENCHC') THEN
                      REMOVE RECORD FROM LIST BAD
                   END IF
                END FOR
        SORT.RECS: SORT RECORDS ON LIST BAD BY FULLNAME
                FOR EACH RECORD IN SORT.RECS
                    .
                    .
                    .
        END
```

$CHKPAT

The $CHKPAT function verifies the syntax of a pattern. If the pattern is valid, a null string is returned; otherwise, an error message string is returned.

Syntax

`$CHKPAT(`*`pattern`*`[,`*`language`*`])`

Where

The required *pattern* argument is the string of characters to verify as a valid pattern, which can be a literal enclosed in quotation marks or a %variable.

The optional *language* argument specifies the language to use. The *language* argument is handled as follows:

- When *language* is omitted, Model 204 performs the validation in U.S. English, even if the value of the LANGUSER parameter is not US, and lowercase characters are not recognized.
- An asterisk enclosed in quotation marks ('*') instructs Model 204 to use the value of the LANGUSER parameter.
- You can enter the name of a valid language enclosed in quotation marks or a %variable containing a valid language. If the value you enter is not supported, the request is canceled with an error message. See "LANGUSER: Setting the language definition of a user thread" on page 4 for valid values.

Without $CHKPAT, pattern syntax errors can cause cancellation of the request or require the coding of complex ON units.

Examples

For U.S. English:

```
%PAT='ABC*'

%X=$CHKPAT(%PAT)
IF %X NE '' THEN
 PRINT %X
 JUMP TO ERROR.RETURN
END IF
```

For French Canadian:

```
%PAT='pêché'

%X=$CHKPAT(%PAT,'FRENCHC')
IF %X NE '' THEN
 PRINT %X
 JUMP TO ERROR.RETURN
END IF
```

$LANGSPC

The $LANGSPC function returns a string containing the hexadecimal value of the specified character in the specified language. You can use $LANGSPC to scan user input for a special character in a language-independent manner. A print-out or display of the returned value will be the character representation based on the language argument.

You can also use the $LANGSPC function to ensure that any special character that has a different hexadecimal code value is displayed correctly.

Syntax

`$LANGSPC('`*`charname`*`'[,`*`language`*`])`

Where

The *charname* argument is a string containing one of the following values:

Valid *charname*	US character	Description
AT	@	At sign
BACKSLSH	\	Backslash
DOLLAR	$	Dollar sign
DQUOTE	"	Double quotation mark
EXCLAMAT	!	Exclamation point
NOT	¬	Not sign
RBRACE	]	Closing square brace or right square brace
SHARP	#	Number sign or pound sign
VERTICAL	|	Vertical bar

The optional *language* argument specifies which language to use to obtain the desired hexadecimal code for the specified character. The request is canceled with an error message if the name is not found in NLANG. The *language* argument is handled as follows:

- When you omit the *language* argument, Model 204 performs the validation in U.S. English, even if the value of the LANGUSER parameter is not US, and lowercase characters are not recognized.
- An asterisk enclosed in quotation marks ('*') instructs Model 204 to use the value of the LANGUSER parameter.

- You can enter the name of a valid language enclosed in quotation marks or a %variable containing a valid language. If the value you enter is not supported, the request is canceled with an error message. See “LANGUSER: Setting the language definition of a user thread” on page 4 for valid values.

Example In the following example, the %PATH variable, supplied by the user from the terminal, is searched for the backslash character in the code table designated by the user’s LANGUSER value:

```
%BACKSLASH IS STRING LEN 1
%BACKSLASH = $LANGSPC(‘BACKSLSH’,‘*’)
%DIR = $SUBSTR(%PATH,$INDEX(%PATH,%BACKSLASH)+1)
```

$LANGSRT

The $LANGSRT function translates a given string according to the specified language into a language-neutral hexadecimal string against which you can sort. A print-out or display of the returned value will be the character representation based on the language argument.

By determining whether one string is greater or less than another string, you can use the $LANGSRT function to compare two strings. First apply the $LANGSRT function to the strings and then compare them using the SOUL greater-than (GT) and less-than-or-equal-to (LE) operators.

Syntax `$LANGSRT('`*`string`*`'[,`*`language`*`])`

Where The *string* argument is a literal enclosed in quotation marks or a %variable containing the original data to be translated into collating sequence.

The optional *language* argument is the name of one of the defined languages, which specifies which collating sequence to use. The *language* argument is handled as follows:

- When you omit the *language* argument, Model 204 performs the validation in U.S. English, even if the value of the LANGUSER parameter is not US, and lowercase characters are not recognized.
- An asterisk enclosed in quotation marks (‘*’) instructs Model 204 to use the value of the LANGUSER parameter.
- You can enter the name of a valid language enclosed in quotation marks or a %variable containing a valid language. If you enter a value that is not supported, the request is canceled with an error message. See “LANGUSER: Setting the language definition of a user thread” on page 4 for valid values.

Note: The $LANGSRT function returns the string unchanged when the language is U.S. English.

Example The following procedure stores the value of NAME from each record into array %STR. The $LANGSRT function translates each value of NAME into a language specific collating sequence and stores the value into the array %SORTSTR. The procedure then calls a user written subroutine, MYSORT, that sorts the %SORTSTR array in ascending order. At this point the procedure invokes the $LANGUST function to translate the collating string back to its original form and prints the names in language specific order.

```
BEGIN
DECLARE SUBROUTINE MYSORT (STRING LEN 20 ARRAY(*))
%STR STRING LEN 20 ARRAY (20) NO FS
%SORTSTR STRING LEN 20 ARRAY (20) NO FS
*
FD1: IN DATA FIND ALL RECORDS
     END FIND
     %I = 1
     FOR EACH RECORD IN FD1
       %STR(%I) = NAME
       %I = %I + 1
     END FOR
     FOR %J FROM 1 TO %I-1
      %SORTSTR(%J) = $LANGSRT(%STR(%J),‘TURKISH’)
      END FOR
*
* SORT NAMES
*
CALL MYSORT(%SORTSTR)
*
     FOR %J FROM 1 TO %I-1
      %STR(%J) = $LANGUST(%SORTSTR(%J),’TURKISH’)
      PRINT %STR(%J)
     END FOR
END
```

$LANGUST

The $LANGUST function translates back to its original form a string previously translated by $LANGSRT processing, which is useful for applications that maintain sorted arrays of data and need to display the values.

Syntax `$LANGUST(‘`*`string`*`’[,`*`language`*`])`

Where The *string* argument is a literal enclosed in quotation marks or a %variable containing the data in collating sequence to be translated back to its original form.

The optional *language* argument is the name of one of the defined languages, specifying which collating sequence to use. The *language* argument is handled as follows:

- You can enter the name of a valid language enclosed in quotation marks or a %variable containing a valid language. If the value you enter is not supported, the request is canceled with an error message. See "LANGUSER: Setting the language definition of a user thread" on page 4 for valid values.
- An asterisk enclosed in quotation marks ('*') instructs Model 204 to use the value of the LANGUSER parameter.
- When you omit the *language* argument, Model 204 performs the validation in U.S. English, even if the value of the LANGUSER parameter is not US, and lowercase characters are not recognized.

Example If your site maintains more than one type of terminal and keyboard that store and display the same character set, individual characters might be assigned differing hexadecimal codes on different keyboards. You can translate the character equivalents back and forth as follows:

```
$LANGUST($LANGSRT(string,source-language),
          target-language)
```

A character without an equivalent converts to its base character. A special character without an equivalent converts to a space.

$LIKE

The $LIKE function provides user control over the parsing and evaluation languages used in pattern matching. It has two language arguments: one to assign the parsing language, LANGUSER, and one to assign the evaluation language, LANGFILE.

The LANGLIKE operator and $LIKE function in expressions coordinate to provide consistency between the FIND statement and the IF statement, and avoid complicating the interpretation of the evaluation language parameter.

Syntax

```
$LIKE(string,pattern[,parse-lang][,eval-lang])
```

Where The *string* argument represents the characters to verify. It must be one of the following:

- A literal enclosed in quotation marks.
- A %variable.
- A field name without quotation marks. In this case, the function call must be embedded in a FOR EACH RECORD loop where the current value of the field is verified.

The required *pattern* argument is the string of characters to verify, which you can specify as a literal enclosed in quotation marks or as a %variable.

The optional *parse-lang* argument specifies the language to use for parsing. The *parse-lang* argument is handled as follows:

- Omitting this argument instructs Model 204 to use U.S. English parsing rules, even if the value of the LANGUSER parameter is not US.
- An asterisk enclosed in quotation marks (‘*’) instructs Model 204 to use the value of the LANGUSER parameter.
- You can enter the literal name of a valid language enclosed in quotation marks. If you enter a name that is not supported, the request is canceled with an error message. See “LANGUSER: Setting the language definition of a user thread” on page 4 for valid values.

The optional *eval-lang* argument specifies the language to use for evaluation. Its requirements are identical to the *parse-lang* argument.

Example In the following example, we are matching the value of field NAME against the pattern (A-Z)*@ using US as the parsing language and TURKISH as the evaluation language.

- The parsing language, LANGUSER, determines the special characters that can be used in a pattern. The pattern is checked for syntax against these characters. The evaluation language, LANGFILE, is used when the pattern is matched against the data.
- The evaluation language, LANGFILE, determines the collating sequence and the definition of alphabetic characters.

In this example, the evaluation language is Turkish. therefore all character matching is done against the Turkish alphabet and the range operation, (A–Z), uses the collating sequence of the Turkish language.

```
BEGIN
FD1: IN DATA FIND ALL RECORDS
   END FIND
   %PAT=‘(A-Z)*@’
   FOR EACH RECORD IN FD1
    %RC = $LIKE(NAME,%PAT,‘US’,‘TURKISH’)
    IF %RC EQ 0 THEN
     PRINT ‘STRING: ’WITH NAME WITH ’DOES NOT MATCH PATTERN: -
           ‘  WITH %PAT
    END IF
   END FOR
END
```

$LOWCASE

The $LOWCASE function translates an uppercase or mixed-case string into a lowercase string. The translation affects only characters with uppercase and lowercase pairs, for example, A to a through Z to z in U.S. English. These are not strictly keyboard pairs. If the first character in the string is alphabetic, the character is converted to uppercase.

Syntax `$LOWCASE(`*`string`*`[,`*`language`*`])`

Where The *string* argument represents the characters to verify, which must be entered as follows:

- A literal enclosed in quotation marks.
- A %variable.
- A field name without quotation marks. In this case, the function call must be embedded in a FOR EACH RECORD loop where the current value of the field is verified.

The optional *language* argument specifies the language to use, which is handled as follows:

- Omitting the *language* argument instructs Model 204 to perform the validation in U.S. English, even if the value of the LANGUSER parameter is not US.
- An asterisk enclosed by quotation marks (’*’) instructs Model 204 to use the value of the LANGUSER parameter.
- You can enter a literal name of a valid language enclosed in quotation marks. If the name you enter is not supported, the request is canceled with an error message. See “LANGUSER: Setting the language definition of a user thread” on page 4 for valid values.

Example The following example returns the string ’Name and address’ in U.S. English:

```
$LOWCASE(‘NAME AND ADDRESS’)
```

The following example returns the string ‘Çà et là’ in French Canadian:

```
$LOWCASE(‘ÇÀ ET LÀ’,‘FRENCHC’)
```

$UPCASE

The $UPCASE function translates a lowercase or mixed-case string into an uppercase-only string. The translation affects only the uppercase letters of character pairs in the specified language.

Syntax `$UPCASE(`*`string`*`[,`*`language`*`])`

Where The *string* argument represents the characters to verify. which must be entered as follows:

- A literal enclosed by quotation marks.

- A %variable.
- A field name without quotation marks. In this case, the function call must be embedded in a FOR EACH RECORD loop where the current value of the field is verified.

The optional *language* argument specifies the language to use. The *language* argument is handled as follows:

- Omitting this argument instructs Model 204 to perform the validation for U.S. English, even if the value of the LANGUSER parameter is not US.
- An asterisk enclosed in quotation marks (‘*’) instructs Model 204 to use the value of the LANGUSER parameter.
- A literal name of a valid language enclosed in quotation marks. If the name you enter is not supported, the request is canceled with an error message. See “LANGUSER: Setting the language definition of a user thread” on page 4 for the valid values.

Examples The following examples return uppercase strings for mixed case entries.

Function code…	Returns…	Language
`$UPCASE(‘Name and address’)`	`‘NAME AND ADDRESS’`	U.S. English
`$UPCASE(‘Île d’Orléans’,‘FRENCHC’)`	`‘ÎLE D’ORLÉANS’`	French Canadian
`$UPCASE(‘Île d’Orléans’)`	`‘ILE D’ORLeANS`	U.S. English

Note: In U.S. English no accented characters have case translation.

Terminal interface requirements

Output validation on 3270 full-screen threads uses the list of displayable characters that is specified in the thread’s language table, specified by “LANGUSER: Setting the language definition of a user thread” on page 4 or by the default language, US.

If no such list is supplied, then no output validation is performed, regardless of the setting of the FSTRMOPT parameter.

If there is a list of displayable characters, then output validation is performed when the FSTRMOPT parameter setting allows it; that is, when the X’01’ bit is off.

The *UPPER and *LOWER commands, which set case translation, use the case translation rules specified in the thread’s language table. If no case translation rules are specified, then no case translation is performed, regardless of the *UPPER or *LOWER command setting.

Using the JAPAN language table

The JAPAN language table is designed to handle Katakana terminal display and to provide upward compatibility with DBCS support in previous releases of Model 204. In particular, case translation and 3270 output validation are disabled.

Using DBCSENV for uppercase translation

When the DBCSENV parameter is set to a nonzero value, the LANGUSER parameter is automatically set to JAPAN. See the *Model 204 DBCS Support Summary* for the use and setting of this parameter.

Uppercase translation depends on the DBCSENV parameter. In the non-DBCS environment, when an *UPPER command is in effect, Model 204 converts data received from the user to uppercase for:

- Full-screen editor commands
- Screen input items not specified as mixed case
- Line mode input (for example, command and $READ input)

Extended text lines for full-screen editor

Full-screen editor users in the Fujitsu environment can now input extended text lines of up to 255 display positions. If the storage requirement of such a line exceeds 255, the line is truncated cleanly. The screen is resent to the user with the truncated line highlighted, and an error message is displayed in the full-screen editor’s message window.

A

Control Characters

Control characters

Table A-1 lists the control characters found in IBM computers and the sequence in which they are sorted. The Character Name column lists the Bisync Communications Protocol Standard acronym in uppercase letters. Where the action is identified, the acronym is followed by a description in lowercase letters.

Table A-1.Control characters

Character Name	Character	U.S. English	Display	Japanese	Display	French Canadian	Display
NUL null		00		00		00	
SOH start of heading		01		01		01	
STX start of text		02		02		02	
ETX end of text		03		03		03	
SEL select		04		04		04	
HT horizontal tab		05		05		05	
RNL		06		06		06	
DEL delete		07		07		07	
GE		08		08		08	
SPS		09		09		09	

Table A-1.Control characters (continued)

Character Name	Character U.S. English	Display	Japanese	Display	French Canadian	Display
RPT	0A		0A		0A	
VT vertical tab	0B		0B		0B	
FF form feed	0C		0C		0C	
CR carriage return	0D		0D		0D	
SO shift out	0E		0E		0E	
SI shift in	0F		0F		0F	
DLE data link escape	10		10		10	
DC1 device control 1	11		11		11	
DC2 device control 2	12		12		12	
DC3 device control 3	13		13		13	
RES	14		14		14	
NL new line	15		15		15	
BS back space	16		16		16	
POC	17		17		17	
CAN cancel	18		18		18	
EM end of medium	19		19		19	
UBS	1A		1A		1A	
CU1 control unit 1	1B		1B		1B	
IFS file separator	1C		1C		1C	
IGS group separator	1D		1D		1D	
IRS record separator	1E		1E		1E	
IUS unit separator	1F		1F		1F	
DS	20		20		20	
SOS	21		21		21	
FS file separator	22		22		22	
WUS	23		23		23	

Table A-1.Control characters (continued)

Character Name	Character	U.S. English	Display	Japanese	Display	French Canadian	Display
BYP		24		24		24	
LF line feed		25		25		25	
ETB end of text block		26		26		26	
ESC escape		27		27		27	
SA		28		28		28	
SFE		29		29		29	
SM		2A		2A		2A	
CSP		2B		2B		2B	
MFA		2C		2C		2C	
ENQ enquiry		2D		2D		2D	
ACK acknowledge		2E		2E		2E	
BEL bell		2F		2F		2F	
		30		30		30	
		31		31		31	
SYN synchronous idle		32		32		32	
IR		33		33		33	
PP		34		34		34	
TRN		35		35		35	
NBS		36		36		36	
EOT end of transmission		37		37		37	
SBS		38		38		38	
IT		39		39		39	
RFF		3A		3A		3A	
CU3 control unit 3		3B		3B		3B	
DC4 device control 4		3C		3C		3C	

Table A-1.Control characters (continued)

Character Name	Character	U.S. English	Display	Japanese	Display	French Canadian	Display
NAK negative ack		3D		3D		3D	
		3E		3E		3E	
SUB substitute		3F		3F		3F	
Space		40	Y	40		40	Y
Non-breaking space						41	Y
Zero width space							
Zero width non-joiner							
Zero width joiner							
Left-to-right mark							
Right-to-left mark							

B

Special Characters

Special characters

Table B-1 lists the special characters found in text, such as punctuation marks, diacritic (or accent) marks, currency symbols, arithmetic and mathematical marks, building blocks for screen forms, and Optical Character Recognition characters.

The word processing software and printer used to create this document cannot display an image of all the characters listed. Please consult the *Unicode Standard Worldwide Character Encoding, Version 1.0, Volume 1* for a display of the missing characters.

Table B-1. Special characters

Character Name	Character	U.S. English	Display	Japanese	Display	French Canadian	Display
Underscore	_	6D	Y	6D		6D	Y
Nonspacing underscore	_						
Double underscore	‗					DF	Y
Macron	¯						
Soft hyphen (shy)	-						
Hyphen (minus sign)	-	60	Y	60		60	Y
EN dash	-						

Table B-1. Special characters (continued)

Character Name	Character	U.S. English	Display	Japanese	Display	French Canadian	Display
EM dash	—						
Horizontal bar	—						
Double horizontal bar	=					DD	Y
Comma	,	6B	Y	6B		6B	Y
Double comma	„						
Semicolon	;	5E	Y	5E		5E	Y
Colon	:	7A	Y	7A		7A	Y
Exclamation point	!	5A	Y	5A		4F	Y
Inverted exclamation	¡					AA	Y
Question mark	?	6F	Y	6F		6F	Y
Inverted question mark	¿	AB	Y	AB		AB	Y
Slash (solidus)	/	61	Y	61		61	Y
Nonspacing long slash overlay	/						
Period	.	4B	Y	4B		4B	Y
Ellipsis	…						
Middle dot-small bullet	·	B3	Y	B3		B3	Y
Cedilla	¸	9D	Y	9D		E0	Y
Ogonek							
Nonspacing ogonek							
Apostrophe	'	7D	Y	7D		7D	Y
Open quote	‘						
Close quote	’						
Quotation mark	"	7F	Y	7F		7F	Y
Double open quote	“						
Double close quote	”						
Left pointing single guillemet	‹						

Table B-1. Special characters (continued)

Character Name	Character	U.S. English	Display	Japanese	Display	French Canadian	Display
Right pointing single guillemet	›						
Left pointing guillemet	«						
Right pointing guillemet	»						
Opening parenthesis	(	4D	Y	4D		4D	Y
Superscript opening parenthesis	$^{(}$						
Closing parenthesis	)	5D	Y	5D		5D	Y
Superscript closing parenthesis	$^{)}$						
Opening square bracket	[	BA	Y	BA		BC	Y
Closing square bracket	]	BB	Y	BB		BE	Y
Opening curly brace	{	C0	Y	C0		51	Y
Closing curly brace	}	D0	Y	D0		54	Y
Section	§	B5	Y	B5		B5	Y
Paragraph (pilcrow)	¶	B6	Y	B6		B4	Y
Copyright	©	B4	Y	B4		BA	Y
Registered trademark	®	AF	Y	AF		B9	Y
Trademark	™						
At sign	@	7C	Y	7C		7C	Y
Generic currency sign	¤						
Euro-currency sign	₠						
Deutsche mark							
Florin	ƒ						
Franc	₣					A0	Y
Lira	₤						
Mill (usa 1/10th cent)	₥						
Kroner							
Peseta	Ᵽ						

Table B-1. Special characters (continued)

Character Name	Character	U.S. English	Display	Japanese	Display	French Canadian	Display
Rupee	Rs						
Won	₩						
New sheqel							
Cent	¢	4A	Y	4A		B0	Y
Dollar	$	5B	Y	5B		5B	Y
Pound	£	B1	Y	B1		B1	Y
Yen	¥	B2	Y	B2		B2	Y
Asterisk	*	5C	Y	5C		5C	Y
Back slash (reverse solidus)	\	E0	Y	E0		BD	Y
Ampersand	&	50	Y	50		50	Y
Number sign	#	7B	Y	7B		7B	Y
Numero	№						
Percent sign	%	6C	Y	6C		6C	Y
Permille sign	‰						
Superscript minus sign	⁻						
Plus sign	+	4E	Y	4E		4E	Y
Superscript plus sign	⁺						
Plus/minus sign	±	8F	Y	8F		8B	Y
Acute accent	´	BE	Y	BE		5A	Y
Grave accent	`	79	Y	79		79	Y
Breve							
Nonspacing breve							
Circumflex	^	B0	Y	B0			
Caret	ˆ						
Hacek (caron)	v						
Ring above	°						

Table B-1. Special characters (continued)

Character Name	Character	U.S. English	Display	Japanese	Display	French Canadian	Display
Nonspacing ring above	°						
Diaeresis	¨	BD	Y	BD		A1	Y
Double acute							
Nonspacing double acute							
Tilde	~						
Dot	·						
Nonspacing dot	·						
Division sign	÷					E1	Y
Multiplication sign	×	BF	Y	BF		BF	Y
Not equal to sign	≠						
Double less-than sign	«	8A	Y	8A		8C	Y
Less-than sign	<	4C	Y	4C		4C	Y
Less-than or equal sign	≤						
Equals sign	=	7E	Y	7E		7E	Y
Greater-than or equal sign	≥						
Greater-than sign	>	6E	Y	6E		6E	Y
Double greater-than sign	»	8B	Y	8B		AC	Y
Not sign	¬	5F	Y	5F		5F	Y
Vertical bar	|	4F	Y	4F			
Pipe	¦	6A	Y	6A		BB	Y
Double vertical bar	‖						
Dagger	†					9D	Y
Double dagger	‡						
Degree	°	90	Y	90		90	Y
Degree fahrenheit	°F						
Degree centigrade	°C						

Table B-1. Special characters (continued)

Character Name	Character	U.S. English	Display	Japanese	Display	French Canadian	Display
Open bullet	∘	9F	Y	9F		9F	Y
Large bullet	•						
Micro (Greek mu)	µ					B6	Y
Ohm							
End of proof (qed)							
Forms light down and right							
Forms light down and horizontal							
Forms light down and left							
Forms light vertical and right							
Forms light vertical and horizontal							
Forms light vertical and left							
Forms light up and right							
Forms light up and horizontal							
Forms light up and left							
Forms light vertical	|						
Forms light horizontal	_						
Forms heavy horizontal							
Left arrow	←						
Right arrow	→						
Non-spacing right harpoon above							
Up arrow	↑						
Down arrow	↓						
Eighth note							
Forms light diagonal upper right to lower left	/						
Forms light diagonal upper left to lower right	\						

Table B-1. Special characters (continued)

Character Name	Character	U.S. English	Display	Japanese	Display	French Canadian	Display
Black lower right triangle							
Black lower left triangle							
OCR hook							
OCR chair							
OCR fork							
OCR inverted fork							
OCR belt buckle							
OCR bow tie							
OCR branch bank identification							
OCR amount of check							
OCR dash (on us)							
OCR customer account number							
OCR double backslash							

C

Latin Alphabet, Diacritics, Ligatures, and Numerals

Latin alphabet, diacritics, ligatures, and numerals

Table C-1 lists the characters used to build words in U.S. English, French Canadian, and other written languages utilizing the Latin and extended Latin character set

The word processing software and printer used to create this document cannot display an image of all the characters listed. Consult *The Unicode Standard: Worldwide Character Encoding, Version 1.0, Volumn 1* for a display of the missing characters.

Table C-1. Latin alphabet, diacritics, ligatures, and numerals

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
	a	81	Y	81			Y	
	A	C1	Y	C1		C1	Y	
Aelingus (lc)	æ	9C	Y	9C		9C	Y	Y
Aelingus (uc)	Æ	9E	Y	9E		9E	Y	Y
Superscript a feminine ordinal end	ª					9A	Y	
Acute accent (lc)	á					45	Y	
Acute accent (uc)	Á					65	Y	

Table C-1. Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
Grave accent (lc)	à					4A	Y	
Grave accent (uc)	À					64	Y	
Breve (lc)	ă							
Breve (uc)	Ă							
Circumflex (lc)	â					42	Y	
Circumflex (uc)	Â					62	Y	
Ring (lc)	å					47	Y	
Ring (uc)	Å					67	Y	
Diaereses (lc)	ä					43	Y	
Diaereses (uc)	Ä					63	Y	
Tilde (lc)	ã					46	Y	
Tilde (uc)	Ã					66	Y	
Ogonek (lc)								
Ogonek (uc)								
Macron (lc)	ā							
Macron (uc)	Ā							
	b	82	Y	82		82	Y	
	B	C2	Y	C2		C2	Y	
	c	83	Y	83		83	Y	
	C	C3	Y	C3		C3	Y	
Acute accent (lc)	ć							
Acute accent (uc)	Ć							
Circumflex (lc)	ĉ							
Circumflex (uc)	Ĉ							
Caron (lc)	č							
Caron (uc)	Č							

Table C-1.   Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	**Hex code**	**Display**	**Japanese**	**Display**	**Hex code**	**Display**	**Ligature**
Dot (lc)	ċ							
Dot (uc)	Ċ							
Cedilla (lc)	ç					48	Y	
Cedilla (uc)	Ç					68	Y	
Spanish ch	ch							
Spanish CH	CH							
	d	84	Y	84		84	Y	
	D	C4	Y	C4		C4	Y	
Small letter d with caron (lc)	ď							
Caron (uc)	Ď							
Bar [eth] (icelandic)	đ	8C	Y	8C		8E	Y	
Bar [ETH] (icelandic)	Đ	AC	Y	AC		AE	Y	
	e	85	Y	85		85	Y	
	E	C5	Y	C5		C5	Y	
Acute accent (lc)	é					C0	Y	
Acute accent (uc)	É					71	Y	
Grave accent (lc)	è					D0	Y	
Grave accent (uc)	È					74	Y	
Breve (lc)	ĕ							
Breve (uc)	Ĕ							
Circumflex (lc)	ê					52	Y	
Circumflex (uc)	Ê					72	Y	
Caron (lc)	ě							
Caron (uc)	Ě							
Diaereses (lc)	ë					53	Y	
Diaereses (uc)	Ë					73	Y	

Table C-1. Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
Dot (lc)	ė							
Dot (uc)	Ė							
Ogonek (lc)								
Ogonek (uc)								
Macron (lc)	ē							
Macron (uc)	Ē							
	f	86	Y	86		86	Y	
	F	C6	Y	C6		C6	Y	
	g	87	Y	87		87	Y	
	G	C7	Y	C7		C7	Y	
Breve (lc)	ğ							
Breve (uc)	Ğ							
Dot (lc)	ġ							
Dot (uc)	Ġ							
Lappish, Latvian small letter g cedilla (lc)	ģ							
Cedilla (uc)	Ģ							
	h	88	Y	88		88	Y	
	H	C8	Y	C8		C8	Y	
Circumflex (lc)	ĥ							
Circumflex (uc)	Ĥ							
Turkish small letter i with no dot (lc)								
	i	89	Y	89		89	Y	
	I	C9	Y	C9		C9	Y	
Turkish capital letter I dot (uc)	İ							
Acute accent (lc)	í					55	Y	
Acute accent (uc)	Í					75	Y	

Table C-1. Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
Grave accent (lc)	ì					58	Y	
Grave accent (uc)	Ì					78	Y	
Breve (lc)								
Breve (uc)	Ĭ							
Circumflex (lc)	î					56	Y	
Circumflex (uc)	Î					76	Y	
Diaereses (lc)	ï					57	Y	
Diaereses (uc)	Ï					77	Y	
Ogonek (lc)								
Ogonek (uc)								
Macron (lc)								
Macron (uc)	Ī							
Dutch ij (lc)	ij							
Dutch IJ (uc)	IJ							
	j	91	Y	91		91	Y	
	J	D1	Y	D1		D1	Y	
	k	92	Y	92		92	Y	
	K	D2	Y	D2		D2	Y	
Cedilla (lc)	ķ							
Cedilla (uc)	Ķ							
	l	93	Y	93		93	Y	
	L	D3	Y	D3		D3	Y	
Middle dot (lc)	l·							
Middle dot (uc)	Ŀ							
Acute accent (lc)								
Acute accent (uc)	Ĺ							

Table C-1. Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
Caron (lc)	ľ							
Caron (uc)	Ľ							
Polish small letter l oblique (lc)	ł							
Oblique (uc)	Ł							
Latvian small l cedilla (lc)	ļ							
Cedilla (uc)	Ļ							
	m	94	Y	94		94	Y	
	M	D4	Y	D4		D4	Y	
	n	95	Y	95		95	Y	
	N	D5	Y	D5		D5	Y	
Acute accent (lc)	ń							
Acute accent (uc)	Ń							
Caron (lc)	ň							
Caron (uc)	Ň							
Tilde (lc)	ñ					49	Y	
Tilde (uc)	Ñ					69	Y	
Latvian small letter n cedilla (lc)	ņ							
Cedilla (uc)	Ņ							
Spanish & Portuguese n with tilde (lc)	ñ							
Spanish & Portuguese N with tilde (uc)	Ñ							
	o	96	Y	96		96	Y	
	O	D6	Y	D6		D6	Y	
Small letter oe	œ							
Capital letter OE	Œ							
Superscript masculine ordinal end	º	9B	Y	9B		9B	Y	
Acute accent (lc)	ó					CE	Y	

Table C-1. Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
Acute accent (uc)	Ó					EE	Y	
Grave accent (lc)	ò					CD	Y	
Grave accent (uc)	Ò					ED	Y	
Breve (lc)	ŏ							
Breve (uc)	Ŏ							
Circumflex (lc)	ô					CB	Y	
Circumflex (uc)	Ô					EB	Y	
Diaereses (lc)	ö					CC	Y	
Diaereses (uc)	Ö					EC	Y	
Double acute (lc)								
Double acute (uc)								
Tilde (lc)	õ					CF	Y	
Tilde (uc)	Õ					EF	Y	
Oblique (lc)	ø					70	Y	
Oblique (uc)	Ø					80	Y	
Macron (lc)	ō							
Macron (uc)	Ō							
	p	97	Y	97		97	Y	
	P	D7	Y	D7		D7	Y	
	q	98	Y	98		98	Y	
	Q	D8	Y	D8		D8	Y	
	r	99	Y	99		99	Y	
	R	D9	Y	D9		D9	Y	
Acute accent (lc)	ŕ							
Acute accent (uc)	Ŕ							
Caron (lc)	ř							

Table C-1. Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
Caron (uc)	Ř							
Cedilla (lc)	ş							
Cedilla (uc)	Ş							
	s	A2	Y	A2		A2	Y	
	S	E2	Y	E2		E2	Y	
German small letter sharp s (Greek beta)	β					59	Y	Y
Acute accent (lc)	ś							
Acute accent (uc)	Ś							
Caron (lc)	š							
Caron (uc)	Š							
Cedilla (lc)	ş							
Cedilla (uc)	Ş							
	t	A3	Y	A3		A3	Y	
	T	E3	Y	E3		E3	Y	
Lappish small letter t bar (lc)	ŧ							
Bar (uc)	Ŧ							
Caron (lc)	ť							
Caron (uc)	Ť							
Romanian small letter t cedilla (lc)	ţ							
Cedilla (uc)	Ţ							
Icelandic thorn (lc)	þ					B7	Y	
Icelandic thorn (uc)	Þ					B8	Y	
	u	A4	Y	A4		A4	Y	
	U	E4	Y	E4		E4	Y	
Acute accent (lc)	ú					DE	Y	
Acute accent (uc)	Ú					FE	Y	

Table C-1. Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
Grave accent (lc)	ù					6A	Y	
Grave accent (uc)	Ù					FD	Y	
Breve (lc)	ŭ							
Breve (uc)	Ŭ							
Circumflex (lc)	û					DB	Y	
Circumflex (uc)	Û					FB	Y	
Ring (lc)	ů							
Ring (uc)	Ů							
Diaereses (lc)	ü					DC	Y	
Diaereses (uc)	Ü					FC	Y	
Double acute (lc)								
Double acute (uc)								
Tilde (lc)	ũ							
Tilde (uc)	Ũ							
Ogonek (lc)								
Ogonek (uc)								
Macron (lc)	ū							
Macron (uc)	Ū							
	v	A5	Y	A5		A5	Y	
	V	E5	Y	E5		E5	Y	
	w	A6	Y	A6		A6	Y	
	W	E6	Y	E6		E6	Y	
	x	A7	Y	A7		A7	Y	
	X	E7	Y	E7		E7	Y	
	y	A8	Y	A8		A8	Y	
	Y	E8	Y	E8		E8	Y	

Table C-1. Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
Acute accent (lc)	ý					8D	Y	
Acute accent (uc)	Ý					AD	Y	
Diaereses (lc)	ÿ					8A	Y	
Diaereses (uc)	Ÿ							
	z	A9	Y	A9		A9	Y	
	Z	E9	Y	E9		E9	Y	
Acute accent (lc)	ź							
Acute accent (uc)	Ź							
Caron (lc)	ž							
Caron (uc)	Ž							
Dot (lc)	ż							
Dot (uc)	Ż							
Swedish diaeresis (lc)	ä							
Swedish diaeresis (uc)	Ä							
Swedish ring (lc)	å							
Swedish ring (uc)	Å							
Danish aelingus (lc)	æ							
Danish aelingus (uc)	Æ							
Duplicate character (1 of 4)	à							
Numeral zero	0	F0	Y	F0		F0	Y	
Superscript zero	0							
One quarter	¼					AF	Y	
One half	½					8F	Y	
Three quarters	¾					CA	Y	
Numeral one	1	F1	Y	F1		F1	Y	
Superscript one	1					DA	Y	

Table C-1. Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
Roman numeral one	i							
Numeral two	2	F2	Y	F2		F2	Y	
Superscript two	2					EA	Y	
Roman numeral two	ii							
Numeral three	3	F3	Y	F3		F3	Y	
Superscript three	3					FA	Y	
Roman numeral three	iii							
Numeral four	4	F4	Y	F4		F4	Y	
Superscript four	4							
Roman numeral four	iv							
Numeral five	5	F5	Y	F5		F5	Y	
Superscript five	5							
Roman numeral six	v							
Numeral six	6	F6	Y	F6		F6	Y	
Superscript six	6							
Roman numeral six	vi							
Numeral seven	7	F7	Y	F7		F7	Y	
Superscript seven	7							
Roman numeral seven	vii							
Numeral eight	8	F8	Y	F8		F8	Y	
Superscript eight	8							
Roman numeral eight	viii							
Numeral nine	9	F9	Y	F9		F9	Y	
Superscript nine	9							
Roman numeral nine	ix							
Roman numeral 10	x							

Table C-1. Latin alphabet, diacritics, ligatures, and numerals (cont.)

Language		U.S. English		Japanese		French Canadian		
	Character	Hex code	Display	Japanese	Display	Hex code	Display	Ligature
Roman numeral 11	**xi**							
Roman numeral 12	**xii**							
Roman numeral 13	**xiii**							
Roman numeral 14	**xiv**							
Roman numeral 15	**xv**							
Roman numeral 50	**l**							
Roman numeral 100	**c**							
Roman numeral 500	**d**							
Roman numeral 1000	**m**							
EO end of		FF		FF		FF		

Index