Rocket | M204

Rocket Model 204 DBCS Support Summary

Version 7 Release 4.0

May 2012
204-0704-DBCS-01

Notices

Edition

Publication date: May 2012

Book number: 204-0704-DBCS-01

Product version: Rocket Model 204 DBCS Support Summary Version 7 Release 4.0

Copyright

Trademarks

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 regulation should be followed when exporting this product.

Contact information

Web Site: www.rocketsoftware.com

Rocket Software, Inc. Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451–1468
USA
Tel: +1.617.614.4321
Fax: +1.617.630.7100

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 vii
Model 204 documentation set vii
Documentation conventions vii

Installing the DBCS Software

In this chapter 1
Overview 1
Reference materials 1
Installation steps 2
Installing DBCS in an IBM z/OS environment 2
Supported z/OS operating environments 2
Installing DBCS using INS204 2
Installing DBCS in z/OS using the standard JCL library 3
Installing DBCS in an IBM z/VM environment 4
Supported z/VM operating environments 4
Installation steps 4
Dictionary/204 support for DBCS 4
Determining if a field in Dictionary/204 is BINARY or DBCS/MIXED DBCS 4

Using the DBCS Feature

In this chapter 5
Overview 5
Pure DBCS data 6
MIXED DBCS data 6
DBCS naming conventions 6
Environment-specific requirements 6
Using DBCS in non-full-screen mode 7
Using continuation characters 8
Unsupported DBCS editors 8
Full-screen procedure editor with DBCS 8
LOCATE and REPLACE command limitations 8
Printing in a DBCS environment 9
Redirecting output 9
Font sizes in DBCS 10
PRINT statement syntax 11

User Language Programming in a DBCS Environment

In this chapter 13
Overview 13
Valid DBCS program elements 14
DBCSMODE parameter 14
DBCS data types 15
Assignment rules for DBCS data 15
Concatenating strings 15

Printing data 16
Nonnumeric literals 16
%Variables 17
Truncation of %variables in Fujitsu environments 17
%Variable syntax 18
Image items 18
Image item syntax 18
Screen items 18
Default data types for screen items 18
Screen item syntax 19
User Language storage allocation 20
Case translation for EBCDIC and Katakana 20
Case translation syntax 20
Options with MIXED DBCS screen items 21
Options with pure DBCS screen items 21
Sample screen definition 21
Indirectly referenced DBCS screen items 21
$Functions 22
User Language $functions 22
DBCS $functions 22
Host Language Interface IFDECL function 23
IFDECL function syntax 23
Notes and tips 23
Coding example 24
Retrieval conditions 24

File Management in a DBCS Environment

In this chapter 25
Overview 25
Defining DBCS fields 25
DEFINE FIELD command syntax 26
Preallocating fields 26
Displaying DBCS fields 27
Initializing files 27
Redefining field attributes 27
Using the File Load utility 27
Read and Load a Field statement 28
DBCS mode bits 29
Rules for using File Load with DBCS 29

System Management in a DBCS Environment

In this chapter 31
Overview 31
DBCSENV parameter 31
DBCS parameter 32
Resetting the DBCS parameter 33
Non-DBCS operator terminals 33
Minimum LOUTPB requirements 33
DBMS-specific messages 33

Index

About this Guide

This guide describes the double-byte character set (DBCS) feature for Model 204.

Audience

This guide is for installers, users, programmers, file managers, and system managers who work on systems that are implementing the Model 204 DBCS feature.

Model 204 documentation set

The complete commercially released documentation for the latest version of Model 204 is available for download from the Rocket M204 customer portal.

To access the Rocket Model 204 documentation:

1. Navigate to:

   www.rocketsoftware.com/m204

2. From the drop-down menu, select **Products > Model 204 > Documentation**.

3. Click the link to the current release and select the document you want from the list.

4. Click the .zip file containing the document.

5. Choose whether to open or save the document:

   – Select **Open** and double-click the pdf file to open the document.

   – Select **Save as** and select a location to save the zip file to.

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*.

Convention	Description
`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.
EDIT	Partial bolding indicates a usable abbreviation, such as E for EDIT in this example.

1

Installing the DBCS Software

In this chapter

This chapter has the following sections:

- Overview
- Installation steps
- Installing DBCS in an IBM z/OS environment
- Installing DBCS in an IBM z/VM environment
- Dictionary/204 support for DBCS

Overview

Model 204 is distributed with all of the materials necessary to install and use Model 204, plus any separately purchased features. Object and source modules that support separately purchased features are distributed in an encrypted (encoded) form. Until they are decrypted, they are useless to any utilities that process them such as the assembler and the linkage editor. The DBCS feature is implemented in a separate object module for this reason.

Reference materials

To prepare and install the DBCS feature, use the Rocket Model 204 installation guide that applies to your environment (z/OS or z/VM).

Decryption keys are provided by Technical Support. Decryption keys are 8-byte numeric strings; when using the decryption utility, be careful to specify the entire eight bytes, including any leading zeros. The decryption utility rejects any key that is nonnumeric or not exactly eight bytes in length.

Installation steps

Installing the DBCS feature requires the following steps for all operating systems:

1. Unload the DBCS module. Because this takes place while the Model 204 base product is installed, it is not discussed in detail here.
2. Decrypt the DBCS module. This step requires the decryption key from Technical Support.
3. Linkedit (or generate) the load modules.

Installing DBCS in an IBM z/OS environment

Supported z/OS operating environments

See the *Rocket Model 204 Installation Guide for IBM z/OS* for the z/OS operating systems that support the current version of Model 204.

Installing DBCS using INS204

To install the DBCS feature using INS204, follow these steps:

1. Use the INSPARMS parameter file to specify all of the site-dependent information for your installation and to specify the features and interfaces to install.
2. Obtain the DBCS decryption key from Technical Support.
3. To install the DBCS feature, modify the DBCS statement. Set DBCS=Y and specify the decryption key in the KEY= field. For example:

   ```
   DBCS=Y KEY=nnnnnnnn ** Install DBCS support
   ```

4. Modify the INSTALL-M204 INSPARMS statement as follows:
   - If you are installing the Model 204 base product at this time, specify:

     ```
     INSTALL-M204=I ** Install Model 204
     ```

   - If Model 204 is already installed and you are adding DBCS, specify:

     ```
     INSTALL-M204=R ** Relink Model 204
     ```

INS204 generates all of the jobs necessary to install Model 204 with all features specified, including DBCS. The jobs generated that apply to the DBCS installation are:

- M204DECR, which runs the decryption utility for all features selected, including DBCS.
- M204LINK (generated by INSTALL-M204=I), which runs the linkedit steps for ONLINE, BATCH204, IFAM1, and IFAM4. The 'INCLUDE OBJLIB(DBCS)' statement is automatically generated.
- M204RLNK (generated by INSTALL-M204=R), which runs the linkedit steps for ONLINE, BATCH204, IFAM1, and IFAM4, including previously linked modules, and adding the 'INCLUDE OBJLIB(DBCS)' statement.

For more information on using INS204, see the *Rocket Model 204 Installation Guide for IBM z/OS.*

Installing DBCS in z/OS using the standard JCL library

Decrypting DBCS

To decrypt DBCS, follow these steps:

1. Use the standard JCL library member M204DECR to decrypt the DBCS object module.
2. Edit the job to add a job card and submit.

See the *Rocket Model 204 Installation Guide for IBM z/OS* for a description of the error messages and return codes used by the decryption utility.

Linkediting the load modules

The DBCS module is included in the ONLINE, BATCH204, IFAM1,and IFAM4 modules. Follow these steps to linkedit the load modules:

1. Do one of the following:
   - If you are linking these load modules for the first time, use the M204LINK job to linkedit the modules.
   - If you are adding DBCS to an existing set of load modules, use the M204RLNK job to include the previously linked module, add the DBCS module, and replace the load module.
2. Make sure that the steps that link the four main configurations have an include statement for the DBCS module as follows:

   ```
   INCLUDE OBJLIB(DBCS)
   ```

For more information about the M204LINK and M204RLNK jobs, see the *Rocket Model 204 Installation Guide for IBM z/OS.*

Installing DBCS in an IBM z/VM environment

Supported z/VM operating environments

See the *Rocket Model 204 Installation Guide for IBM z/VM* for the z/VM operating systems that support the current version of Model 204.

Installation steps

Decrypting DBCS

Decrypt the DBCS module using the M204CRYP command. The format of the command is:

```
M204CRYP DECODE DBCS key
```

where *key* is the 8-byte numeric decryption key for the DBCS feature, which is provided by Technical Support. For more information, see the *Rocket Model 204 Installation Guide for IBM z/VM*.

For Help on this command, issue:

```
HELP 204 M204CRYP
```

Dictionary/204 support for DBCS

Dictionary/204 supports the DBCS and MIXED DBCS data types with the restrictions described in this section.

Determining if a field in Dictionary/204 is BINARY or DBCS/MIXED DBCS

BINARY fields are specified in the METADATA field attribute record as being neither STRING nor FLOAT nor DBCS nor MIXED DBCS. For example:

```
STRING=N
FLOAT=N
DBCS=N
MDBCS=N
```

Therefore, if you do not explicitly check for DBCS and MIXED DBCS, you might inadvertently assume that a DBCS or a MIXED DBCS field is BINARY.

If your site uses both BINARY and DBCS or MIXED DBCS fields, rewrite your applications to include a check for DBCS and MIXED DBCS.

Note: This upward incompatibility is an issue only if your site uses DBCS or MIXED DBCS.

2

Using the DBCS Feature

In this chapter

This chapter has the following sections:

- Overview
- Environment-specific requirements
- Using DBCS in non-full-screen mode
- Full-screen procedure editor with DBCS
- Printing in a DBCS environment

Overview

Kanji is the Japanese name for one of the subsets of Chinese characters used in Japan. Several mainframe vendors (IBM and Fujitsu) support terminals and printers that display Kanji characters.

Because Kanji has more than 256 characters, each character requires two bytes. The coding schemes for various vendors differ; in general, however, all 2-byte coding schemes are called double-byte character set (DBCS) schemes.

Pure DBCS data

Pure DBCS data contains only double-byte data. Model 204 cannot enter or read English or Katakana in DBCS fields. A DBCS character always occupies two output character positions on the terminal screen.

MIXED DBCS data

MIXED DBCS fields can contain both double- and single-byte data. The two types of data are differentiated within the field by escape, or shift sequences.

A shift sequence, which is a series of one or more control bytes, defines whether a series of bytes is interpreted as EBCDIC or DBCS. Shift sequences are used on terminals where both EBCDIC and DBCS characters can be used in the same field or item.

Users can toggle between the two character sets by shifting into or out of EBCDIC from DBCS. Different vendors have defined different shift sequences, some one byte and some two bytes long. Shift sequences take up an output position only on IBM printers.

DBCS naming conventions

Although DBCS and MIXED DBCS are referred to throughout this document, you can also enter all DBCS and MIXED DBCS data types and options as KANJI or MIXED KANJI. For example, you can define a field as either STRING DBCS or STRING KANJI, and you can define an image item as either MIXED DBCS or MIXED KANJI.

Environment-specific requirements

DBCS terminals (in the IBM mainframe compatible environment) are implemented as enhanced 3270 terminals, but other types of terminals are also available.

The terminals listed in Table 2-1 support Model 204 DBCS.

Table 2-1. Valid device types for DBCS

Terminal type	Description
IBM 5500 series	Terminals that use the DBCS specific features of the IBM 3270 Data Stream. IBM 5500 series also support all three data types (pure EBCDIC, pure DBCS, and MIXED DBCS).
TEPCO	Similar to IBM 3270 terminals, which support the DBCS extended attribute (but not MIXED DBCS input). TEPCO also support pure EBCDIC and pure DBCS, but not MIXED DBCS. They are always used with IBM mainframes and often in combination with IBM 5500 series terminals.

Table 2-1. Valid device types for DBCS (Continued)

Terminal type	Description
Fujitsu	Similar to 3270 terminals, which support only MIXED DBCS fields. They do not recognize the IBM 3270 DBCS extended attribute for DBCS, because all fields are MIXED DBCS. Shift sequences are one byte long and do not occupy output positions on the terminal. Fujitsu terminals are used only with Fujitsu mainframes. Model 204 supports Fujitsu data streams.

Note: Due to significant differences among hardware environments, a single Model 204 run can access only one type of DBCS terminal. The only exception to this is the TEPCO environment, where both TEPCO and IBM 5500 series terminals are supported.

Using DBCS in non-full-screen mode

In a variety of circumstances, you might need to enter DBCS characters in non-full-screen mode at the terminal. These circumstances include:

- $ENTER function
- $READ and $READINV functions
- Command-level input
- Operator responses
- CCAIN
- Line-at-a-time emulation of full-screen terminals

The DBCSENV and DBCS parameters control the terminal field attribute for non-full-screen input. If the DBCSENV parameter indicates a 5500 series or Fujitsu terminal, and the terminal has DBCS capability, then all line-oriented terminal input is performed using MIXED DBCS data. Otherwise, line-oriented input is performed with EBCDIC data. MIXED DBCS does not have case translation. See Chapter 5 for a description of the DBCSENV and DBCS parameters.

Use $ENTER to assign data to DBCS and MIXED DBCS variables using STRING to DBCS and STRING to MIXED DBCS rules. However, $ENTER does not check for truncation of trailing shift sequences or splitting of byte-pairs. Whenever possible, use $READ instead of $ENTER to enter DBCS data.

Use $READ and $READINV to read in DBCS and MIXED DBCS data from the terminal. For IBM 5500 series terminals, $READ-style input is either EBCDIC or MIXED DBCS for an entire request. For Fujitsu environments, all input is MIXED DBCS. DBCS input to $READ/$READINV is not supported for the TEPCO environment.

Using continuation characters

You must use an EBCDIC character to continue a line, by placing the character in the input character continuation column. DBCS input in the input character continuation column (as determined by the INCCC parameter) is not valid. If DBCS input is in INCCC (including the trailing shift), any input after the end of that DBCS string is discarded, and the line is not continued.

Unsupported DBCS editors

DBCS is not supported for the line editor or for Line-at-a-time emulation of full screen terminals.

Full-screen procedure editor with DBCS

The full screen editor allows MIXED DBCS input on the command line and in the general input area. Escape sequences are ignored in the following cases:

- First character of a search target is a shift out
- Last character of a search target is a shift in

The REPLACE command preserves pairing of shift sequences and drops contiguous shift sequences. The Model 204 terminal interface eliminates contiguous shift sequences from user input.

The LINEND character must be set to a character that cannot appear within a DBCS character byte-pair. If LINEND is set to a value that can occur within DBCS data, then LINEND does not function properly.

LOCATE and REPLACE command limitations

The LOCATE and REPLACE commands have the following limitations:

- If you have enabled any of the pattern-matching characters (ARBCHAR, REPEAT, or LINEND), you cannot enter them in an EBCDIC portion of a search string, if the search string also contains DBCS characters, or you will receive the following error message:

  ```
  PATTERN CHAR INVALID IN MIXED DBCS LOCATE/REPLACE
  ```

- If the replacement string ends in DBCS characters, then the replacement count, if specified, must be 1. If you enter a replacement count other than 1, you will receive the following error message:

  ```
  FINAL PORTION OF REPLACEMENT IS DBCS AND REPLACEMENT
  COUNT IS NOT 1.
  ```

- ARBCHAR does not match DBCS characters.

Printing in a DBCS environment

The PRINT statement treats a terminal field in which data is displayed as a MIXED DBCS field. That is, all DBCS data is enclosed in shift sequences, even if the source of the data is a pure DBCS item.

The PRINT statement can send output to a variety of devices in conjunction with the USE command. However, Model 204 interprets the shift sequences in accordance with the user's terminal type, as follows:

Environment	Results
IBM	Shift sequences each take up a single output position on both the terminal and in printed output. On printed output the sequences are translated into blanks by the printer.
Fujitsu	Shift sequences do not take up output positions.

When you use the PRINT statement to print DBCS data to a terminal, the following rules apply:

- Column alignment using the AT and TO clauses is based on displayable output positions.
- Values truncated by using both AT and TO on the same print item are truncated based on displayable output positions. The integrity of the shift sequences is preserved.

These rules allow you to align columns correctly, whether or not shift sequences take up output positions on the terminal. The following sections discuss how to align columns correctly for displaying on a terminal and printing to a printer.

Redirecting output

You can redirect output from PRINT via the USE command to do the following:

- Send output to a printer. In this case, you want AT and TO to be interpreted as displayable output position.
- Build a disk data set. If you plan to ultimately route the data to a printer, then you want AT and TO to be interpreted as displayable output position. If you build a disk data set for any other reason (for example, to add input to a File Load program) you want AT and TO to be interpreted by byte position.

Two parameters, DBCSUPOS and DBCSOPOS, control the interpretation of AT and TO.

DBCSUPOS parameter for DBCS USE positioning

The DBCSUPOS parameter changes the interpretation of AT and TO for USE positioning. You can position text either by byte position or by displayable

output position for USE data set pointers with the user resettable parameter, DBCSUPOS. This parameter applies only to Fujitsu environments.

DBCSUPOS has the following settings:

For this setting...	Formatting is by...
0	Byte position
1	Displayable output position

The default is DBCSUPOS=0, which is compatible with standard text positioning.

If you send the PRINT output to a disk data set via USE and DBCSUPOS=1, then you must set the LRECL of the data set equal to the number of output positions per record multiplied by 5/3 for Fujitsu. This setting ensures that enough bytes are available on each record to hold the desired number of output positions (including the shifts).

DBCSOPOS parameter for DBCS output positioning

The DBCSOPOS parameter changes the interpretation of AT and TO for output positioning. You can position text by byte position or by displayable output position for non-USE data sets with the user-resettable parameter, DBCSOPOS. This parameter applies only to Fujitsu environments.

DBCSOPOS has the following settings:

For this setting...	Formatting is by...
0	Byte position
1	Displayable output position

The default is DBCSOPOS=1.

Font sizes in DBCS

In Fujitsu environments, two font sizes are available for DBCS:

- LARGE (6 pitch)
- SMALL (7.5 pitch)

Each font size is equal to two 12 pitch or 15 pitch EBCDIC characters, respectively.

Use the *FONT option to specify font size. You can specify the *FONT option once for each PRINT statement, as well as in SET HEADER and SET TRAILER statements.

The default font size is LARGE (6 pitch). If you specify SMALL, the alternate font escape sequence, determined by the DBCSENV parameter, is substituted for default escape sequences. Terminals display only one font size.

All DBCS characters, regardless of font, occupy two output positions for disk output.

In addition to the keywords LARGE and SMALL, you can also specify the size as a %variable.

You can use the “PRINT X...” form of the PRINT statement to mix large and small characters on a single line.

PRINT statement syntax

The following PRINT statement syntax is specific only to the DBCS environment:

```
PRINT *FONT={LARGE | SMALL | %variable} [print specifications]
```

For other syntax related to the PRINT statement, see the *Rocket Model 204 User Language Manual.*

3

User Language Programming in a DBCS Environment

In this chapter

This chapter has the following sections:

- Overview
- Valid DBCS program elements
- DBCSMODE parameter
- DBCS data types
- %Variables
- Image items
- Screen items
- $Functions
- Retrieval conditions

Overview

Any elements of an application seen or used by the end user, such as field data, screens, or menus, can be entered or displayed on the screen in DBCS.

You can use DBCS (or KANJI) and MIXED DBCS (or MIXED KANJI) data in the following Model 204 programming constructs:

- %Variables
- Image items
- Fields
- Screen items

The rules for programming in the DBCS environment vary slightly from standard User Language programming. These variations are described in this chapter.

Valid DBCS program elements

The following Model 204 program elements can contain DBCS characters:

- Screen names
- Screen item names
- Image names
- Image item names
- %Variable names
- Statement labels
- List names
- Comments

DBCSMODE parameter

When DBCS applications written under Release 9.0 of Model 204 are run under later versions of Model 204, you must set the DBCSMODE parameter.

Set DBCSMODE to 1 to allow the application to remain compatible with Release 9.0 of Model 204 while running under a later version of Model 204.

Note: If DBCSMODE is not set to 1 when running Release 9.0 applications, unpredictable results might occur.

The valid settings for the DBCSMODE parameter are:

Setting	Meaning
0	Application was written under Version 2.1.0 or later.
1	Application was written under Release 9.0 and is being run under a later version of Model 204. This setting places the application in Release 9.0 compatibility mode.

Setting	Meaning
2	Users can enter DBCS data into EBCDIC screen items.

DBCS data types

The following data types are available to DBCS users:

- DBCS (or KANJI)
- MIXED DBCS (or MIXED KANJI)

DBCS and MIXED DBCS are treated as string (rather than numeric) for compiling expressions. You can define screen items, %variables, and image items as DBCS or MIXED DBCS. Nonnumeric literals that contain DBCS characters are always compiled as MIXED DBCS, whether or not they contain any EBCDIC characters.

Assignment rules for DBCS data

Table 3-1 shows the conversion rules for assignment statements involving DBCS and MIXED DBCS data types.

Table 3-1. DBCS data assignment conversion rules

Source	Target	Conversion rules
NUMERIC	MIXED DBCS	Same as NUMERIC and STRING.
	DBCS	Null result.
STRING	MIXED DBCS	Same as MIXED to MIXED.
	DBCS	Same as DBCS to DBCS.
MIXED DBCS	NUMERIC	Same as STRING to NUMERIC.
	STRING	Same as STRING to STRING.
	MIXED DBCS	Truncates enough extra bytes to avoid splitting a DBCS character or removing a trailing shift sequence.
	DBCS	If all DBCS characters, then removes shifts and truncates in byte-pairs; otherwise, null result.
DBCS	NUMERIC	Always results in 0.
	STRING	Same as STRING to STRING.
	MIXED DBCS	Truncation occurs in byte-pairs, and the result is placed inside shift sequences.
	DBCS	Truncation occurs in byte-pairs.

Concatenating strings

The following rules apply when concatenating strings, using the WITH operator:

- Numeric items are always converted to EBCDIC STRINGS before concatenating.
- Concatenating unlike items gives a MIXED DBCS result.
- Concatenating like items gives a result of that type.
- Adjacent shift sequences are eliminated from MIXED DBCS results, unless DBCSMODE is set to 1.
- MIXED DBCS results are truncated so that they avoid splitting a DBCS character or dropping a trailing shift sequence.
- DBCS results are truncated in byte-pairs.

These assignment and concatenation rules result in the automatic handling of shift sequences. As long as DBCS characters are restricted to DBCS and MIXED DBCS items, all DBCS characters appear within correctly paired shift sequences (when necessary), and are displayed properly on terminals and printers.

Note: If you print EBCDIC data that contains unmatched pairs of shifts to a terminal, you cause the terminal to hang (that is, stop processing). To avoid hanging the terminal, make sure that you have correctly paired shift sequences surrounding EBCDIC data.

Printing data

The following translation rules apply when using any form of the User Language PRINT statement:

Source	Target
EBCDIC	EBCDIC
MIXED DBCS	MIXED DBCS
PURE DBCS	MIXED DBCS

Nonnumeric literals

Nonnumeric literals can currently appear in several different contexts. In specifying a field value, a nonnumeric literal can appear as an unquoted string with quotes surrounding any portions of the string that contain reserved words or characters. Similarly, you can enter dummy string parameters specified in an INCLUDE statement without quotes. In all other contexts, you must quote nonnumeric literals.

The User Language compiler treats literals containing DBCS characters as MIXED DBCS items, regardless of whether they contain EBCDIC characters. If DBCSMODE=1, however, literals containing DBCS characters are treated as

EBCDIC literals. See the section “DBCSMODE parameter” on page 14 for more information.

Quotes are not required around DBCS field values. For MIXED DBCS field values, quotes must surround reserved words or characters that appear in the EBCDIC portion of the value. You must always enclose in quotes any DBCS and MIXED DBCS literals that appear elsewhere.

%Variables

STRING DBCS and STRING MIXED DBCS (or, as an alternative syntax, STRING KANJI and STRING MIXED KANJI) are DBCS data types for %variables.

If you declare a %variable as STRING (with no keyword), then Model 204 determines the %variable using the following rules:

- If the default type is STRING EBCDIC, STRING MIXED DBCS, or STRING DBCS, the %variable is given the default type.
- If the %variable type is numeric, the %variable is declared as STRING EBCDIC.

When you log in to any DBCS environment, the default %variable type is initially set to STRING EBCDIC. Set the default %variable type with the VARIABLES ARE statement or the RESET VTYPE command.

STRING DBCS or STRING KANJI %variables can contain only DBCS characters without shift sequences. The length of the DBCS field is always an even number of bytes. STRING DBCS %variables cannot contain DP specifications.

For STRING MIXED DBCS %variables, Model 204 assumes that the field contains both EBCDIC and DBCS characters and that all DBCS characters are contained within balanced shift sequence pairs. That is, if you begin the %variable definition in EBCDIC and shift into DBCS, you must shift back out of DBCS at the end of the definition. Specify the length of STRING MIXED DBCS %variables as the number of single-character display positions in the string.

Truncation of %variables in Fujitsu environments

MIXED DBCS %variables might be truncated in Fujitsu DBCS environments under certain conditions. If a %variable is declared with a display length greater than 109, then this value is converted using the DBCSLEN conversion formulas to determine the amount of storage needed:

For Fujitsu the formula is:

```
(5/3 * DISPLAY LENGTH)
```

%Variables with more than 109 display positions require a storage length of more than 255 bytes. This case is handled by truncating the value at 255 bytes,

which is the value that Model 204 uses for storage length. Whenever the display length is needed, the value is determined from the storage length using the DBCSLEN conversion formulas.

Due to this truncation at 255 bytes of storage, any variables declared with more than 109 display positions are truncated at 109 display positions.

%Variable syntax

The following syntax for %variables is specific only to the DBCS environment:

```
[DECLARE] %variable [IS] STRING [EBCDIC | MIXED {DBCS |
KANJI}| DBCS | KANJI] [options]
```

For other syntax relating to the DECLARE *%variable* statement, see the *Rocket Model 204 User Language Manual*.

Image items

DBCS and MIXED DBCS image items follow the same general rules as %variables. The following additions and exceptions apply:

- DBCS items cannot contain decimal point (DP) specifications.
- DBCS image items cannot contain PAD, STRIP, or JUSTIFY options.
- If you specify the PAD option for a MIXED DBCS image item, the pad character must be a single EBCDIC character.

Image item syntax

The following syntax for image items is specific only to the DBCS environment:

```
itemname IS [TYPE] STRING [EBCDIC | MIXED {DBCS | KANJI}
     | DBCS | KANJI] [options]
```

For other syntax relating to the image item statement, see the *Rocket Model 204 User Language Manual*.

Screen items

You can specify data type once for each screen item (TITLE, INPUT, PROMPT), or on a DEFAULT statement in the screen definition.

Default data types for screen items

The following data types are valid:

- EBCDIC
- DBCS (or KANJI)

- MIXED DBCS (or MIXED KANJI)

The default data for a screen item depends on the terminal. If the terminal is an IBM 5500 series or Fujitsu with DBCS capabilities, then the default type for screen items is MIXED DBCS. In all other cases, the default type is EBCDIC.

If the terminal is Fujitsu, then the data type is always compiled as MIXED DBCS. If you try to change the data type to EBCDIC, the change is ignored. Fujitsu does, however, allow you to specify a data type of DBCS. In this case, Model 204 validates that all data is DBCS, although it is still compiled as MIXED DBCS.

EBCDIC data type

The EBCDIC data type corresponds to existing screen items and is the default in most cases. Data sent to EBCDIC screen items is validated and any nondisplayable characters (such as DBCS characters) are replaced with question marks.

DBCS data type

If you specify a screen item as DBCS, only DBCS data can be displayed in the field. Shift sequences are not sent to the terminal and are not inserted by Model 204 on input. Specify the length of a DBCS item as the number of single-character screen positions in the item.

MIXED DBCS data type

If you specify a MIXED DBCS screen item, it can contain both EBCDIC and DBCS characters surrounded by shift sequences. Specify the length of the item as the number of single-character screen positions in the item.

In the TEPCO environment (where MIXED DBCS is not supported), specifying MIXED DBCS causes a compiler error.

Screen item syntax

The following syntax for screen items is specific only to the DBCS environment:

```
[DEFAULT] {PROMPT | INPUT | TITLE} [EBCDIC |
      MIXED {DBCS | KANJI} | DBCS | KANJI] [options]
```

For other syntax relating to the screen item statement, see the *Rocket Model 204 User Language Manual.*

User Language storage allocation

The User Language compiler allocates storage for MIXED DBCS screen items and %variables, depending on environment as follows:

Environment	Storage
IBM	LEN
Fujitsu	LEN * 5/3 (rounded up)

where LEN is the internal storage length of an item. The maximum storage is always 255 bytes.

This approach guarantees that enough storage is allocated in the screen item for the largest possible number of bytes (including escape sequences). These formulas were derived from a worst-case scenario of alternating single EBCDIC and DBCS characters.

Case translation for EBCDIC and Katakana

For EBCDIC screen input items, the following case translation options are available:

- UPCASE translates input to uppercase.
- NOCASE suppresses case translation.

You can specify UPCASE and NOCASE on the DEFAULT INPUT and INPUT statements. If you do not specify a case translation option, then case translation depends on whether the *LOWER command has been processed. These case translation options allow the user to enter Katakana characters (which map to lowercase Roman letters) into pure EBCDIC screen items. In addition, Katakana characters that do not map to lowercase EBCDIC characters are not translated to question marks.

Case translation is not performed in DBCS or MIXED DBCS fields regardless of the case option specified. Any adjacent shift sequences are eliminated before data is sent to the terminal and when Model 204 receives input from the user. Consecutive shift sequences (for example, two shift out sequences) are reduced to a single shift out or shift in upon receiving operator input. Model 204 supplies missing trailing shift in sequences.

Case translation syntax

The following syntax for case translation is specific only to the DBCS environment:

```
[DEFAULT] input [EBCDIC] [UPCASE | NOCASE]
```

For other syntax relating to the case translation statement, see the *Rocket Model 204 User Language Manual.*

Options with MIXED DBCS screen items

The following options are supported for MIXED DBCS screen items:

- If you specify PAD for MIXED DBCS screen items, the pad character must be a single EBCDIC character.
- You must enclose in quotes any MIXED DBCS values that contain a space or comma (in either EBCDIC or DBCS positions).

Options with pure DBCS screen items

The following options are not supported for pure DBCS screen items:

- VERIFY
- NUMERIC
- ALPHA
- ALPHANUM
- PAD
- DEBLANK

In Fujitsu environments (where pure DBCS is not supported), Model 204 validates user input to ensure that all DBCS characters are within shift sequence pairs.

Sample screen definition

The following example shows some ways that you can use DBCS in User Language screens:

```
BEGIN
  SCREEN DBCSEX
     TITLE 'EXAMPLE OF DBCS SCREEN SYNTAX'
     DEFAULT INPUT MIXED DBCS
     PROMPT 'NAME:' INPUT NAME EBCDIC NOCASE LEN20
     SKIP 1 LINE
  END SCREEN
END
```

Indirectly referenced DBCS screen items

Indirectly referenced DBCS screen items are treated the same as DBCS strings when using concatenation (using WITH) and comparisons. For more information, see “Concatenating strings” on page 15.

$Functions

This section describes rules for User Language $functions and DBCS-specific $functions.

User Language $functions

The following existing functions treat all DBCS input as EBCDIC strings:

- $DEBLANK
- $INDEX
- $LEN
- $SUBSTR
- $UNBLANK

Arguments that refer to positions or lengths continue to refer to byte positions rather than character positions. To ensure that the operation of existing User Language programs does not change, these functions do not attempt to detect invalid truncation conditions (for example, dropping a trailing shift sequence).

$ONEOF supports both DBCS and MIXED DBCS strings for the first argument and MIXED DBCS strings for the second argument. The third argument (the delimiter character) must be a single EBCDIC character. If the first argument is DBCS, it is converted to MIXED DBCS before it is compared to the second argument.

DBCS $functions

The following DBCS-related $functions are provided:

$Function	Description
$DBINDEX	Returns the output position in which a string appears. $DBINDEX is similar to and has the same arguments as $INDEX.
$DBLEN	Returns the number of displayable output positions in a DBCS or MIXED DBCS string. $DBLEN is similar to and has the same arguments as $LEN.
$DBSBSTR	Returns the portion of a string starting at a specified output position that has a length of a specified number of output positions. $DBSBSTR is similar to and has the same arguments as $SUBSTR. $DBSBSTR produces a MIXED DBCS result and does not split double-byte characters or drop trailing shift sequences.

The following rules apply when counting output positions:

Environment	Rule
Fujitsu	Shifts are not counted as output positions.
IBM	Shifts are counted as output positions.

Host Language Interface IFDECL function

Use the Host Language Interface IFDECL function to declare STRING %variables as one of the following types for use in an IFFIND statement:

Data type	Meaning
STRING EBCDIC	Default type, which contains single-byte characters in the EBCDIC collating sequence.
STRING DBCS	Contains only pure DBCS characters (double-byte data only, with no shift sequences).
STRING MIXED DBCS	Contains both DBCS and EBCDIC data. All DBCS characters are contained within balanced shift sequence pairs. (The shift sequences define whether a series of bytes is interpreted as DBCS or EBCDIC.)

When you use a pure or mixed DBCS %variable in an IFFIND statement, Model 204:

1. Performs the appropriate data type conversions, following the conversion rules for assignment types.
2. Compares the pure and mixed DBCS fields.

IFDECL function syntax

The syntax for the IFDECL function is:

```
IFDECL %variable [IS] [EBCDIC | MIXED {DBCS | KANJI} |
DBCS | KANJI] [options]
```

The parameters for IFDECL are the same as for the Model 204 DECLARE statement.

Notes and tips

IFDECL declares only simple string variables. You cannot use IFDECL for arrays, ASCII, FLOAT, or BINARY strings, or for lists, labels, or subroutines.

IFDECL is allowed only on Multi-Cursor IFAM threads.

Coding example

```
WORKING-STORAGE SECTION.
01 CALL-ARGS.
77 DECLARE-MIXED  PIC X(35) VALUE '%MIXED IS STRING MIXED DBCS LEN 20;'.
77 DECLARE-PURE   PIC X(28) VALUE '%PURE IS STRING DBCS LEN 20;'.
77 FIND-DBCS      PIC X(31) VALUE 'PURE.DBCS = 'Kanji-data';END;'.
77 FIND-NAME      PIC X(08) VALUE 'FIND.DBCS;'


•
•
•
PROCEDURE DIVISION.
•
•
•
CALL IFDECL WITH DECLARE-MIXED.
CALL IFDECL WITH DECLARE-PURE.
CALL IFFIND WITH RETCODE, FIND-DBCS, FIND-NAME.
```

Retrieval conditions

When character string comparisons occur that involve different types of strings, the fields are converted following the conversion rules for assignment statements. For example, if a MIXED DBCS field is compared to a value in a DBCS %variable, the value is surrounded by shift sequences (and truncated if longer than 255 bytes) before the comparison takes place.

The LIKE operator (for pattern matching) handles DBCS and MIXED DBCS data as strings of bytes (shift sequences are not recognized as having special significance). In most cases, therefore, the LIKE operator is not useful for DBCS data. Pattern matching does work with Katakana data.

4

File Management in a DBCS Environment

In this chapter

This chapter has the following sections:

- Overview
- Defining DBCS fields
- Displaying DBCS fields
- Using the File Load utility

Overview

This chapter presents special considerations for file management in a DBCS environment.

For comprehensive information about file management, see the *Rocket Model 204 File Manager's Guide.*

Defining DBCS fields

The following field attributes are specific to DBCS Model 204 files:

- STRING DBCS (or STRING KANJI)
- STRING MIXED DBCS (or STRING KANJI)

In addition, you can use STRING EBCDIC to specify the default STRING data attribute. STRING and STRING EBCDIC are equivalent, and the EBCDIC keyword is not required. If you define a field as STRING EBCDIC, it appears as STRING in a DISPLAY FIELD request.

The STRING DBCS and STRING MIXED DBCS attributes are mutually exclusive. Furthermore, you cannot combine them with any of the following field attributes:

- BINARY
- CODED
- FLOAT
- FRV
- NUMERIC RANGE
- ORDERED NUMERIC
- STRING EBCDIC

If you do combine these attributes, the DEFINE FIELD command fails and you receive an error message.

DEFINE FIELD command syntax

The following DEFINE FIELD command syntax is specific only to the DBCS environment:

```
DEFINE FIELD name WITH STRING
       [EBCDIC | MIXED {DBCS | KANJI}
        | DBCS | KANJI] [attribs ...]
```

For other syntax relating to the DEFINE FIELD command, see the *Model 204 Command Reference Manual.*

Preallocating fields

You can preallocate STRING DBCS and STRING MIXED DBCS fields. Specify the length of preallocated fields in bytes as follows:

Environment	Rule
IBM, Fujitsu, and TEPCO	Length of a STRING DBCS preallocated field must be even. If you specify an odd length in these environments for a STRING DBCS field, then the DEFINE FIELD fails and you receive an error message.

The length of a STRING MIXED DBCS preallocated field must be sufficient to hold at least two shift sequences (one to shift into DBCS and one to shift back out) and one DBCS character:

Environment	Requires a minimum length of...
IBM, Fujitsu, and TEPCO	`(2 * (SHIFT_LEN) + 2)`

where SHIFT_LEN is the length (in bytes) of the shift sequence.

If you do not meet these length requirements, then DEFINE FIELD fails and you receive an error message.

You can define the pad character for STRING DBCS or STRING MIXED DBCS as any single-byte character. However, if you define the pad character to be a character that can be part of a DBCS character, then the field might be interpreted as null.

Displaying DBCS fields

The DISPLAY FIELD command uses the following keywords and abbreviations for these data types:

Attribute	Display symbol	Display abbreviation
STRING EBCDIC	STRING	STR
STRING DBCS	STRING DBCS	STR DBCS
STRING MIXED DBCS	STRING MIXED DBCS	STR MDBCS

Initializing files

When using the INITIALIZE command, the SORT and HASH fields can be DBCS or MIXED DBCS data types. However, STRING DBCS and STRING MIXED DBCS data types are not valid in record security (the RECSCTY option) fields.

Redefining field attributes

Once you have defined the data type for a field as STRING DBCS or MIXED DBCS, you cannot redefine the field data type. You can redefine other field attributes as long as they do not conflict with the rules described in the section “Defining DBCS fields” on page 25.

Using the File Load utility

You can use automatic file reorganization programs unchanged when you add DBCS or MIXED DBCS fields to a file. File load statements can override any defaults so that you can read any input format and turn it into a Model 204 file.

In addition, you can explicitly manipulate shift character sequences when necessary, using File Load's existing programming capabilities.

DBCS characters are valid in any portion of a File Load program in which variable length character strings are currently allowed (for example, in a CASE statement).

You cannot use the File Load Branch-on-Character-Equal statement for double-byte data. To transfer execution of the file load program from one part of the procedure to another, use the CASE statement.

DBCS characters are not valid in any File Load labels.

The following sections discuss the Read and Load a Field statement, including DBCS-specific bit settings, and the general rules for using the File Load utility in a DBCS environment.

Read and Load a Field statement

The syntax of the Read and Load a Field statement does not change in the DBCS environment:

```
(fieldname = pos, len, mode bits)
```

For pure DBCS fields, Model 204 assumes that the input field is surrounded by shift sequences and strips the appropriate number of bytes at the beginning and end of the field before storing in Table B. If the input field is not surrounded by shift sequences, you must suppress the shift stripping with the X'0020' mode bit (see the section "DBCS mode bits" on page 29).

MIXED DBCS fields are subject to the same conversions as string fields (preceding and trailing blanks are stripped), except that blanks are never stripped within a shift sequence pair. Shift sequences are not stripped when storing into a MIXED DBCS field.

The mode bits for a Read and Load a Field statement are:

Mode bit	Meaning
X'0020'	Does not strip shift sequences when storing into pure DBCS fields. Use this bit if the file was produced by some means other than PAI, and the data is already in the correct Table B format.
X'0040'	Surround input with shift sequences. Use this bit when reading pure DBCS data into a MIXED DBCS field or at any time when you must strip shift sequences.

DBCS mode bits

The following mode bits allow you to map DBCS data in string fields to DBCS and MIXED DBCS fields:

Mode bit	Meaning
X'0008'	Adds shift sequences, if needed. Use this bit to examine the value of a DBCS field and, if needed, add shift sequences. If the string starts with SO and ends with SI, it is loaded as is. Otherwise, an SO/SI pair is added surrounding the data before it is stored and indexed. This bit takes precedence over the X'0040' bit.
X'0010'	Strips shift sequences from input, if needed. Use this bit to examine the value of a DBCS field and, if needed, strip the shift sequences. If the string starts with SO and ends with SI, these shifts are removed before storing and indexing. Otherwise, the data remains unchanged. This bit takes precedence over the X'0020' bit.

You can do more complex manipulation of shift sequences with the existing string buffer and index register commands, or with FLOD Exits. For information about File Load processing and FLOD Exits, see the *Rocket Model 204 File Manager's Guide.*

Rules for using File Load with DBCS

Follow these rules when using the File Load utility in a DBCS environment:

- All translation tables are considered MIXED DBCS strings and can contain DBCS characters surrounded by shift sequence pairs. You must properly terminate all DBCS strings, and end the string with an EBCDIC equal sign (=).
- Constant fields are considered MIXED DBCS strings and follow the same translation tables.
- Strings in a CASE statement are considered MIXED DBCS strings and follow the same rules as translation tables. DBCS label names are not valid in a CASE statement.
- No changes are required for string buffers or index registers. You must be aware of the length and format of shift sequences when you manipulate them directly in string buffers, but you can do any conversion. The constant value in an SC statement is treated as a MIXED DBCS string that follows the same rules as the translation table.
- The D statement follows the same conversion rules as the normal Read a Field statement. The D statement reads PAI output and converts it back into its original form.

- The P and Q statements allow you to print the contents of the input record or string buffers and index registers. Set the X’0040’ bit (discussed in the section “Read and Load a Field statement” on page 28) to ensure that the specified contents are printed correctly.

5

System Management in a DBCS Environment

In this chapter

This chapter has the following sections:

- Overview
- DBCSENV parameter
- DBCS parameter
- Minimum LOUTPB requirements

Overview

This chapter presents special considerations for system management in a DBCS environment. Two DBCS-specific parameters and minimum requirements for LOUTPB are discussed.

For comprehensive information about system management, see the *Rocket Model 204 System Manager's Guide.*

DBCSENV parameter

A nonresettable parameter, DBCSENV, describes the DBCS environment under which a user's Model 204 system runs. Because each machine handles DBCS data in different ways, it is important to set DBCSENV correctly.

Valid settings for DBCSENV are listed in Table 5-1.

Table 5-1. Valid DBCSENV parameter settings

Setting	Terminals in use are...
0	DBCS terminals are *not* in use
1	IBM 5500 series
2	TEPCO and IBM 5500 series
3	Fujitsu

When DBCSENV=3, specify Fujitsu terminals in use support for extended attributes as listed in Table 5-2.

Table 5-2. FSOUTPUT settings in a Fujitsu environment

FSOUTPUT setting	Description
0	No extended attributes are supported. Any extended attribute assigned to a screen item is ignored.
1	All extended attributes are supported, utilizing the SFE order.
2	Only extended highlighting attributes are supported and other extended attributes are ignored. The SFE/F order is utilized.

DBCS parameter

The user-definable parameter DBCS indicates to Model 204 whether each terminal supports double-byte character data.

Once the DBCSENV parameter is set, the DBCS parameter is automatically set and might not need changing. The default settings for the DBCS parameter are:

DBCSENV setting	DBCS parameter default
0	0
1, 3, or 4	1
2	3

If DBCS is 0, you cannot reset it. In all other instances, you can reset DBCS:

- On User 0 command line
- On an IODEV line
- In a RESET command in CCAIN
- In a RESET command

Resetting the DBCS parameter

You can reset DBCS as follows:

- If the run supports DBCS but the terminal type is not one of the four listed in Table 5-1 on page 32, or does not support DBCS for some other reason, you must set DBCS to 0.

  Note: When using DBCS=0 on an IODEV=7 thread, all lowercase commands (for example, login user ID or password) fail.

- If the terminal is an IBM 5500 series or a Fujitsu, you must set DBCS to 1.
- If the terminal is a TEPCO, you must set DBCS to 3.

Non-DBCS operator terminals

The User 0 (operator) terminal in a DBCS environment can be non-DBCS. In this case, where the operator terminal is a full-screen non-DBCS device, you must reset the DBCS parameter to 0 (non-DBCS) before issuing a HALT command, or you will receive terminal I/O errors.

Minimum LOUTPB requirements

The minimum value of the LOUTPB parameter differs for various screen sizes (determined by the MODEL parameter) and for various DBCS environments. Table 5-3 shows the minimum value of LOUTPB for each combination of MODEL and DBCSENV.

Table 5-3. Minimum LOUTPB for combinations of DBCSENV and MODEL

Model	DBCSENV=1	DBCSENV=2	DBCSENV=3	DBCSENV=4
2	2134	2134	3498	6042
3	2774	2774	4594	7986
4	3654	3654	6101	10659
5	3778	3778	6231	10983

DBMS-specific messages

See the *Rocket Model 204 Messages Manual* for all DBMS-related messages.

Index

Symbols

$DBINDEX function 22
$DBLEN function 22
$DBSBSTR function 22
$DEBLANK function 22
$Functions 22
    DBCS 22
        $DBINDEX 22
        $DBLEN 22
        $DBSBSTR 22
        $ONEOF 22
    User Language 22
        $DEBLANK 22
        $INDEX 22
        $LEN 22
        $SUBSTR 22
        $UNBLANK 22
$INDEX function 22
$LEN function 22
$ONEOF function 22
$SUBSTR function 22
$UNBLANK function 22
%Variables
    rules in DBCS environment 17
    syntax 18
    truncation in Fujitsu 17
*FONT option 10

A

ALPHA option 21
ALPHANUM option 21
AT clause 9
Audience vii
Automatic file reorganization programs 27

B

BINARY field attribute 26
Branch-on-Character-Equal statement 28

C

CASE statement 28
Case translation
    NOCASE option 20
    syntax 20
    UPCASE option 20
CCAIN 32
Character sets
    DBCS 5, 6
    EBCDIC 6
    Kanji 5
    Katakana 6
CODED field attribute 26
Column alignment 9
Commands
    DEFINE FIELD 26
    DISPLAY FIELD 27
    INITIALIZE 27
    RESET 32
    RESET VTYPE 17
    USE 9
Concatenating strings 15

D

D statement
    File Load utility conversion rules 29
Data types
    DBCS 15, 18
    EBCDIC 18
    KANJI 6, 15, 18
    MIXED DBCS 15, 19
    MIXED KANJI 6, 15, 19
    STRING DBCS 6
    STRING KANJI 6
DBCS
    data types 15
    file management 25
    installing 1
    mode bits 29
    system management 31
    terminals 6
    User Language programming 13

using 5
valid device types 6
valid program elements 14
DBCS $functions 22
DBCS character set 6
DBCS data
assignment rules 15
DBCS data type 15, 18
for screen items 19
DBCS fields
defining 25
displaying 27
preallocating 26
DBCS output positioning 10
DBCS parameter 32
default settings 32
for Fujitsu 33
for IBM 33
for TEPCO 33
on IODEV=7 thread 33
resetting 33
DBCS USE positioning 9
DBCSENV parameter 31
settings 32
DBCSLEN conversion formula for Fujitsu 17
DBCSMODE parameter 14
settings 14
DBCSOPOS parameter 10
settings 10
DBCSUPOS parameter 9
settings 10
DEBLANK option 21
Decimal point specifications 18
Declaring STRING %variables
in HLI 23
Decryption utility
using 2
DEFAULT INPUT statement 20
Default settings
DBCS parameter 32
DEFINE FIELD command
syntax 26
Defining DBCS fields 25
DISPLAY FIELD command 27
abbreviations 27
keywords 27
DISPLAY FIELD request 26
Displaying DBCS fields 27
Double-byte character set (DBCS) 5

E

EBCDIC character set 6
EBCDIC data type 18
for screen items 19
EBCDIC screen input items
case translation 20

F

Field attributes
BINARY 26
CODED 26
DBCS-specific 25
FLOAT 26
FRV 26
NUMERIC RANGE 26
ORDERED NUMERIC 26
redefining 27
STRING DBCS 25
STRING EBCDIC 26
STRING KANJI 25
STRING MIXED DBCS 25
Fields
HASH 27
SORT 27
File Load labels 28
File Load program 28
File Load utility
Branch-on-Character-Equal statement 28
D statements in 29
Read and Load a Field statement 28
mode bits 28
syntax 28
rules with DBCS 29
using 27
File management
in DBCS environment 25
Files
automatic reorganization programs 27
initializing 27
FLOAT field attribute 26
Font sizes
default 11
in DBCS 10
LARGE 10
SMALL 10
FRV field attribute 26
FSOUTPUT settings
in Fujitsu environment 32
Fujitsu terminals 7

H

HASH fields 27
Host Language Interface 23

Host Language Interface IFDECL function 23

I

IBM 5500 series terminals 6
IFDECL function 23
    coding example 24
    restrictions 23
    syntax 23
IFFIND statement 23
Image items 18
    syntax 18
INCLUDE statement 16
Index register 29
INITIALIZE command 27
Initializing files 27
INPUT statement 20
Installing DBCS
    environment-specific requirements 6
Installing the DBCS software 1
IODEV line 32

J

JUSTIFY option 18

K

Kanji character set 5
KANJI data type 6, 15, 18
Katakana character set 6
Katakana screen input items
    case translation 20

L

LARGE font size 10
LIKE operator 24
Literals
    nonnumeric 16
LRECL parameter 10

M

MIXED DBCS data type 15, 19
    for screen items 19
MIXED DBCS screen items
    options 21
MIXED KANJI data type 6, 15, 19
Mode bits
    DBCS 29
    for a Read and Load a Field statement 28
Multi-Cursor IFAM threads 23

N

NOCASE option 20
Non-DBCS operator terminals 33
Nonnumeric literals 16
NUMERIC option 21
NUMERIC RANGE field attribute 26

O

Operators
    WITH 15
ORDERED NUMERIC field attribute 26
Output positions
    rules for counting
        in Fujitsu 23
        in IBM 23

P

P and Q statements 30
PAD option 18, 21
PAI output 29
Parameters
    DBCS 32
    DBCSENV 31
    DBCSMODE 14
    DBCSOPOS 10
    DBCSUPOS 9
    FSOUTPUT 32
    LRECL 10
Pattern matching
    using LIKE operator 24
    with Katakana data 24
Preallocating fields 26
    length requirements
        in Fujitsu 27
        in IBM 27
        in TEPCO 27
PRINT statement 9
    *FONT option 10
    PRINT X... form 11
    redirecting output 9
    rules for DBCS data 9
    syntax in DBCS environment 11
    translation rules 16
Printing
    in DBCS environment 9
Pure DBCS data 6
Pure DBCS screen items 21

R

Read and Load a Field statement 28
    mode bits 28
    syntax 28
RECSCTY option 27
Redefining field attributes 27
Redirecting output 9
Reference materials for installing DBCS 1
RESET command 32
RESET VTYPE command 17
Resetting the DBCS parameter 33
Retrieval conditions 24

S

Screen definition
    example 21
Screen items 18
    default data types 18
    in TEPCO environment 19
    indirectly referenced 21
    MIXED DBCS options 21
    pure DBCS options 21
    storage allocation
        for Fujitsu 20
        for IBM 20
    syntax 19
SET HEADER statement 10
SET TRAILER statement 10
Shift sequences 6
    in Fujitsu 9
    in IBM 9
SMALL font size 10
SORT fields 27
Statements
    CASE 28
    DEFAULT INPUT 20
    INCLUDE 16
    INPUT 20
    PRINT 9
    Read and Load a Field 28
    SET HEADER 10
    SET TRAILER 10
    VARIABLES ARE 17
Storage allocation
    for screen items 20
String buffers 29
STRING DBCS data type 6, 23
STRING DBCS field attribute 25
STRING EBCDIC data type 23
STRING EBCDIC field attribute 26
STRING KANJI data type 6
STRING KANJI field attribute 25
STRING MIXED DBCS data type 23
STRING MIXED DBCS field attribute 25
Strings
    concatenating 15
STRIP option 18
System management
    in DBCS environment 31

T

Table B 28
TEPCO terminals 6
Terminals
    DBCS 6
    DBCS restrictions 7
    Fujitsu 7
    IBM 5500 series 6
    non-DBCS operator 33
    TEPCO 6
TO clause 9

U

UPCASE option 20
USE command 9
User Language
    storage allocation 20
User Language $functions 22
User Language programming
    in DBCS environment 13
User zero command line 32
Using
    DBCS 5
    decryption utility 2
    File Load utility 27

V

Valid device types for DBCS 6
VARIABLES ARE statement 17
VERIFY option 21

W

WITH operator 15