

BCOCA Frequently Asked Questions

Implementation Tips for Producing
Bar Codes with the Bar Code
Object Content Architecture

Editor:

David E. Stone, IBM Corporation

Contributing writers:

Edward F. Chatcavage, IBM Corporation

David E. Stone, IBM Corporation

First Edition (November 20, 2006)

Copyright International Business Machines Corporation 2006. All rights reserved.

The content of this document was developed and approved by members of the AFP Color Consortium (www.afpcolor.org).

Overview

Implementation Tips for Producing Bar Codes with the Bar Code Object Content Architecture

This document is a collection of information about bar codes and the BCOCA architecture intended to supplement the BCOCA Reference manual, S544-3766. It is assumed that the reader is familiar with basic bar code and BCOCA concepts; refer to the BCOCA Reference for a complete description of this architecture. The information in this document is intended for people using BCOCA objects to produce bar code symbols and for people implementing the BCOCA architecture within printers, transforms, or other products.

All of the bar codes supported by the BCOCA architecture are defined by Symbology Specifications provided by industry standards groups, such as Automatic Identification Manufacturers, Inc. (AIM), or by postal organizations. A “symbology” sets the rules for a particular bar code type and defines how information is encoded into the physical attributes of bars, spaces, and two-dimensional modules.

Table of Contents

Bar Code Symbologies	4
Size of a Bar Code Symbol	7
Variations in Bar Code Symbol Width.....	8
Encrypting Bar Code Data.....	9
Aspect Ratio for PDF417 Symbols	9
FNC1 and Data Matrix Symbols	10

Bar Code Symbologies

-- contributed by David E. Stone

What types of bar codes are supported within the BCOCA architecture?

A large variety of symbol types are supported; these can be classified into the following categories:

- Linear bar codes for numeric data
- Linear bar codes for alphanumeric data
- Postal bar codes
- Two-dimensional matrix bar codes
- Two-dimensional stacked bar codes

Linear bar codes for numeric data



UPC Version A
(encoding 01234567890)



UPC A + Two-digit Supplemental
(encoding 00633895260, supplemental = 24)



UPC A + Five-digit Supplemental
(encoding 09827721123, supplemental = 21826)



UPC Version E
(encoding 078349)



MSI - no check digit
(encoding 80523)



EAN 8
(encoding 2468123)



Codabar
(encoding A34698735B)



EAN 13
(encoding 041234567890)



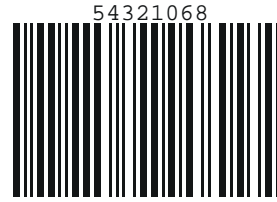
EAN + 2 Digit Supplemental
(encoding 041234567890, supplemental = 99)



EAN + 5 Digit Supplemental
(encoding 041234567890, supplemental = 54321)



Industrial 2-of-5
(encoding 54321068)

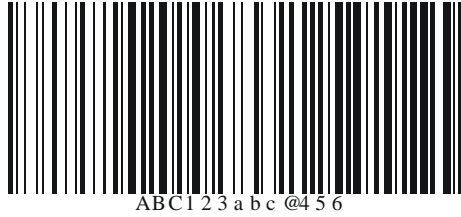


Matrix 2-of-5
(encoding 54321068)



Interleaved 2-of-5
(encoding 54321068)

Linear bar codes for alphanumeric data



Code 128
(encoding ABC123abc@456)



UCC/EAN 128
(encoding 000111111222222229)



Code 39 (3-of-9 Code)
(encoding 39OR93 with check character yielding a 2.32 inch wide symbol)



Code 93
(encoding 39OR93 yielding a 1.82 inch wide symbol)

Postal bar codes



Australia Post Bar Code
Customer Barcode 2 using Table C
(encoding 56439111ABA 9)



Japan Postal Bar Code
Modifier X'00'
(encoding 15400233-16-4)



US POSTNET
Zip+4
(encoding 12345+6789)



PLANET Code
(encoding 00123456789)



USPS Four-State Bar Code
Modifier X'03'
(encoding 01 234 567094 987654321 01234567891)

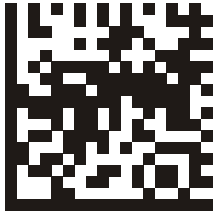


Royal Mail (RM4SCC)
UK and Singapore version
(encoding SN34RD1A)

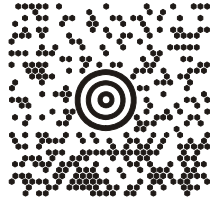


Royal Mail (RM4SCC)
Dutch KIX version
(encoding SN34RD1A)

Two-dimensional matrix bar codes



Data Matrix 2D Symbol
(encoding A1B2C3D4E5F6G7H8I9J0K1L2)

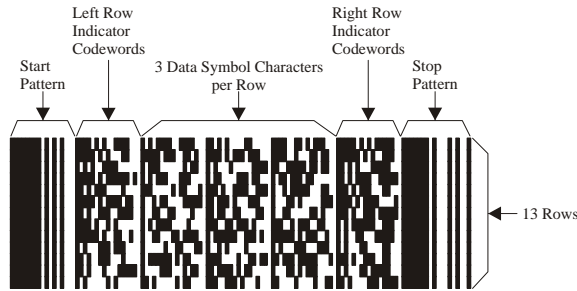


MaxiCode 2D Symbol

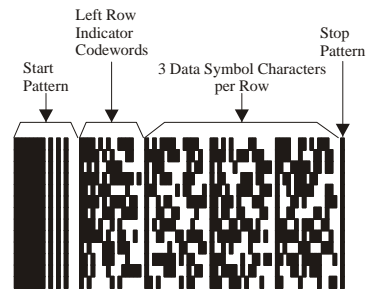


QR Code 2D Symbol

Two-dimensional stacked bar codes



PDF417



Truncated PDF417

What determines the size of a bar code symbol?

The height of a bar code symbol is controlled by the bar code symbology definition, by the amount of data to be encoded, and by various BCOCA parameters. The width of the symbol is usually dependent on the amount of data to be encoded and by choices made in various BCOCA parameters. Default values exist for most of the BCOCA parameters that can be used to produce minimal-size, scannable symbols; refer to your printer documentation for information about the specific default values used by BCOCA printers.

Linear Symbologies: The element-height and height-multiplier parameters specify the height of the symbol. For some bar code types, these parameters also include the height of the human-readable interpretation (HRI). Some bar code symbologies (Australia Post Bar Code, Japan Postal Bar Code, POSTNET, RM4SCC, and USPS Four-State) explicitly specify the bar code symbol height; in this case, the element-height and height-multiplier parameters are ignored.

Two-Dimensional Matrix Symbologies: The MaxiCode symbology specifies a fixed physical size, nominally 28.14 mm wide by 26.91 mm high; the module-width, element-height, and height-multiplier parameters are ignored for MaxiCode symbols.

Data Matrix symbols are rectangular and are made up of a pattern of light and dark squares (called modules). The size of each module is specified in the module-width parameter and the number of rows and columns of these modules is controlled by the desired-number-of-rows and desired-row-size parameters and the amount of data to be encoded. The element-height and height-multiplier parameters are ignored for Data Matrix symbols.

QR Code symbols are square and are made up of a pattern of light and dark squares (called modules). The size of each module is specified in the module-width parameter; the number of rows and columns of these modules is controlled by the version parameter, the error correction level selected, and the amount of data to be encoded. The element-height and height-multiplier parameters are ignored for QR Code symbols.

Two-Dimensional Stacked Symbologies: PDF417 symbols are rectangular and are made up of a pattern of light and dark rectangles (called modules). The size of each module is specified in the module-width, element-height, and height-multiplier parameters and the number of rows and columns of these modules is controlled by the data-symbols and rows parameters and the amount of data to be encoded. A PDF417 symbol must contain at least 3 rows.

Variations in Bar Code Symbol Width

-- contributed by David E. Stone

Why are bar code symbols printed on a 240 pel printer sometimes wider than the same symbol printed on a 300 pel printer?

The width of a BCOCA bar code symbol depends on a variety of factors, including 1) amount and type of data to be encoded, 2) choices of BCOCA parameters (such as, module width, wide-to-narrow ratio, and bar code modifier), and 3) presentation device characteristics. It is the responsibility of the presentation device to produce bars and spaces that are as close to the user-specified module width and wide-to-narrow ratio as is practically possible. A difference in symbol length between different printers (given the same BCOCA object) is usually caused by device-implementation differences (#3).

Typically, size differences are most noticeable in the width direction and this is due to how the module width parameter is handled by the implementation. Module width is specified in mils (1/1000th of an inch) and each printer must convert the specified module width to an integer number of printer pels; this conversion frequently cannot produce an exact measure and each implementer chooses an appropriate rounding or truncation method to yield acceptable results. This is made more difficult by device-specific characteristics, such as:

- 1) Size and positioning precision for each printer pel
- 2) The resolution of the printhead (a good rounding/truncation choice made for a 300 pel/inch printer might be the wrong choice for a 480 pel/inch printer)
- 3) The ability of the print-head technology to produce precise and consistent pels. A laser printer is quite different from an ink jet or ribbon-based printer, a write-black technology is different from a write-white technology, etc.

The goal for each printer-development team is to produce bar widths that are as close to what the user specified as possible and still be within the tolerance defined in the symbology specification. Once a narrow-bar width in printer pels is generated, the printer lays out the sequence of bars and spaces that make up the symbol. If there is a slight difference between the theoretical module width (specified as a BCOCA parameter) and the resulting narrow-bar width in printer pels, this difference is multiplied by the number of bars (and spaces) in the symbol; the error gets worse when there are many bars and spaces in the symbol. Since there are real differences between printers (as described in the previous paragraph), there will be differences in symbol length between vendors and between printers with different resolutions.

The goal is defined by the BCOCA architecture, but the realization is up to the printer implementation. Most bar code symbologies specify a minimum recommended size for module width (for example, 7.5 mils is the minimum for a Code 39 symbol). Size problems are particularly noticeable when the module width is close to the minimum because some implementations round up to avoid producing a bar that is narrower than the minimum (other implementations don't take this into consideration). Some printers provide options on the printer control panel to deal with this kind of boundary condition and to emulate the rounding/truncation algorithm used by other printers.

Since it uses a wide-to-narrow ratio, Code 39 symbols are also affected by the user choice for the wide-to-narrow ratio (WE:NE) parameter and by the ability of the printer to accurately generate the wide-bar that this symbology uses. The wide-to-narrow ratio specifies the module width of wide bars, using a ratio involving the narrow-bar width. If

the narrow-bar width isn't exact, the wide-bar width will be affected which can make the symbol width vary even more. Code 39 isn't the only bar code type for which this symbol-width problem exists, but it is often the most noticeable.

Encrypting Bar Code Data

-- contributed by David E. Stone

How can bar code data be protected by a key or password, so that the data can only be read by using the key or password?

It is possible to create a 2D bar code symbol whose data has been encrypted. Using this method, the customer would encrypt the data for the bar code using whatever encryption method and key the customer chooses, then the BCOCA bar code would contain the encrypted data (instead of the original data). Whenever the bar code is read by a scanning device, the output would be the encrypted data (and therefore not readable by an unauthorized person); the customer would have to decrypt the scanned output after the bar code has been read (this could be done by an application program whose input is from the scanner).

Using encrypted data is possible with BCOCA 2D bar codes because each of these bar code types (PDF417, Data Matrix, MaxiCode, and QR Code) can handle the full 256-byte ASCII and can handle binary data as input (so each input character can have any value and there is no need for the input to be readable by humans).

Aspect Ratio for PDF417 Symbols

-- contributed by David E. Stone

Why can't an aspect ratio (for example, a 2:1 ratio of columns to rows) be specified for BCOCA PDF417 symbol?

The BCOCA parameters for PDF417 are based on the PDF417 symbology definition which defines the symbol in terms of a number of data symbol characters per row and a number of rows. When building a BCOCA PDF417 bar code, you must specify the number of data symbol characters per row. The actual symbol contains 4 more symbol characters (start pattern, stop pattern, left row indicator, and right row indicator); so what is specified is effectively the number of columns minus 4. For the number of rows, you have two choices:

1. Specify a number (which can be 1/2 the number of columns ... giving you a 2:1 ratio); the printer will pad if necessary to fill all of the requested rows
2. Specify X'FF', which tells the printer to use the minimum number of rows necessary to hold the data.

Some application programs allow a user to specify the relative size of a PDF417 symbol as an aspect ratio. When the BCOCA architecture was defined for PDF417, it was decided that having the printer figure out how to arrange the data into a particular aspect ratio was too time consuming at print time (particularly for printers running at greater than 1000 pages per minute). AFP data is usually generated as final form such that the generator makes the layout decisions (for example, the aspect ratio for a bar code). Defining the final-form data is pretty easy for most bar code generators, but it can be harder for a transform product that only has an aspect ratio as input.

How are control characters, such as FNC1, used to produce the smallest Data Matrix symbol?

Background

Data Matrix is a two-dimensional matrix symbology that is made up of square modules arranged within a perimeter finder pattern. The specification for the symbology is published by AIM International, Inc. There are two types of symbols: ECC 000–140 with several available levels of convolutional error correction, and ECC 200 which uses the Reed-Solomon error correction. Since ECC 200 is the recommended type for all new applications, the BCOCA architecture uses only ECC 200.

The manner in which a user's data is converted to form an ECC 200 symbol is as follows:

1. Analyze the user's input data to identify the variety of different characters that will be encoded into the symbol. This is part of the Data Encodation step, where the data is efficiently converted into codewords using six different encodation schemes.

Note: The user's input data may be supplemented with additional information obtained from the Data Matrix Special-Function Parameters as defined in the Bar Code Object Content Architecture Reference (S544-3766).

2. Generate the Reed-Solomon error correction codewords from step 1 above. These codewords will become part of the symbol.
3. Generate the symbol itself by converting the codewords into square modules for placement within the symbol.

Encoding the Data

During the encoding step, there are several special symbology control characters that get inserted as codewords for the symbol. These special control characters are used to instruct a bar code reader on how to interpret the data contained within the symbol. As an example, there are **Latch Control Codewords** to indicate switching between the different encodation schemes. Another special control character is **FNC1** which is used as an *Alternate Data Type Identifier*.

As stated in section 6.2.4.6 of the AIM specification:

"To encode data to conform to specific industry standards as authorized by AIM International, a FNC1 character shall appear in the first or second symbol character position (or in the fifth or sixth data positions of the first symbol of Structured Append). FNC1 encoded in any other position is used as a field separator and shall be transmitted as GS control character (ASCII value 29)."

The FNC1 character is inserted into the data when either one of the two Alternate Data Type identifiers is set in the Data Matrix Special-Function Parameters.

Problem Found While Encoding the Data

A problem was reported by a customer: Some implementations of the Data Matrix bar code are not producing the smallest size symbol for the given amount of input data. Different BCOCA implementations produced slightly different sized symbols. The problem was found to be caused by how the AIM Specification was interpreted by different people.

When the user's data is encoded for the symbol, it is desired that the smallest number of symbol codewords be inserted into the symbol. The AIM Specification has an Annex S (Informative) section describing an algorithm that will usually produce the shortest codeword stream. As part of this algorithm, there is a look-ahead test that scans the data to be encoded to find the best mode. The test basically wants to examine the next four data characters before determining a best mode. In the AIM Specification, part 'Q' of the look-ahead test read as follows:

"Q. Process the B256 count:

- 1. If the character is a Function character (FNC1, Structured Append, Reader Program, or Code Page), add 4 to the B256 count.*
- 2. Otherwise add 1 to the B256 count."*

Now, a FNC1 character could only be in the data when explicitly inserted by the BCOCA-implementation code when processing the Alternate Data Type identifiers parameters. So step 'Q' above should only recognize FNC1 characters that are inserted into the user's data. It so happens, that the codeword value for FNC1 is 232. Therefore, the user's data might contain this value as normal binary data. When this is the case, the look-ahead code should not treat the value 232 as a FNC1 character.

Note: The BCOCA implementation does not specify any means of inputting a FNC1 character as part of a user's data input.