

# **Morovia QRCode Fonts & Encoder 5 Reference Manual**

# Morovia QRCode Fonts & Encoder 5 Reference Manual

Copyright © 2009, 2011, 2017, 2020 Morovia Corporation. All rights reserved.

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Morovia Corporation or its subsidiaries (Morovia). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Morovia does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH MOROVIA® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND MOROVIA, MOROVIA ASSUMES NO LIABILITY WHATSOEVER, AND MOROVIA DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF MOROVIA PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Morovia is a trademark of Morovia Corporation. Other product and company names mentioned herein may be the trademarks of their respective owners.

Publication date: March 2022

Revision: 11739

## **Technical Support**

Phone: (905) 752-0226

Fax: (905) 752-0355

Email: [support@morovia.com](mailto:support@morovia.com)

Web: <http://www.morovia.com>

For more information about Morovia products, visit <http://www.morovia.com>.

# Table of Contents

|  |    |
|--|----|
| 1. Introduction .....                                      | 1  |
| Installing Morovia QRCode Fonts & Encoder 5 .....          | 1  |
| To Install from a CD .....                                 | 1  |
| To Install from Direct Download .....                      | 2  |
| Limitations of Trial Version .....                         | 2  |
| Backward Compatibility .....                               | 2  |
| Font Encoder DLL .....                                     | 2  |
| Crystal Reports .....                                      | 2  |
| Input Format Change for Structural Append .....            | 2  |
| Unicode Encoding Support .....                             | 2  |
| 2. Overview .....  | 5  |
| What is QR Code? .....                                     | 5  |
| Encoding Character Set .....                               | 5  |
| Options .....  | 6  |
| Input Format .....   | 7  |
| Center Overlay Block .....                                 | 7  |
| Kanji Block .....  | 8  |
| Extended Channel Interpretation (ECI) .....                | 8  |
| Structural Append (SA) .....                               | 9  |
| FNC1 (First Position) .....                                | 10 |
| FNC1 (Second Position) .....                               | 10 |
| Unicode String Support .....                               | 10 |
| Working with QRCode Fonts & Encoder 5 .....                | 11 |
| Font-Based .....   | 11 |
| Image File Based .....                                     | 12 |
| ActiveX Control .....                                      | 12 |
| 3. Using Encoder GUI .....                                 | 13 |
| Export Images .....  | 14 |
| Export Options .....                                       | 15 |
| Parity Calculator .....                                    | 16 |
| 4. Word Add-In .....                                       | 17 |
| Install/Remove Add-In .....                                | 17 |
| Creating Single Barcode .....                              | 17 |
| Creating Multiple Barcodes .....                           | 18 |
| Mail Merge .....   | 19 |
| 5. Adding QR Code Symbols to Crystal Reports .....         | 23 |
| User Function DLL .....                                    | 23 |
| Working with Crystal Reports .....                         | 24 |
| Distributing UFL, Fonts with your report application ..... | 26 |
| 6. Adding QR Codes in Microsoft Office .....               | 27 |
| VBA Module .....   | 27 |
| To Import VBA Module .....                                 | 27 |
| Word Mail Merge .....                                      | 27 |
| Using QRCode ActiveX in Word and Excel .....               | 28 |
| Inserting QRCode ActiveX into Word/Excel .....             | 28 |
| Editing QRCode ActiveX Object .....                        | 28 |
| Data Binding in Excel .....                                | 29 |

|   |    |
|---|----|
| Adding QR Code in Access Reports .....                      | 30 |
| Adding QR Code to InfoPath .....                            | 30 |
| Adding QRCode Control to Custom Controls .....              | 30 |
| Inserting Barcode into Form Template .....                  | 32 |
| 7. Developing Application with QRCode Fonts & Encoder ..... | 35 |
| Common Issues .....   | 35 |
| Color Types .....   | 35 |
| Version .....   | 35 |
| Error Codes .....   | 35 |
| Windows Native DLL Interface .....                          | 36 |
| ActiveX Control Interface (COM) .....                       | 36 |
| 8. Encoder DLL API Reference .....                          | 37 |
| ImageTypeEnum .....   | 38 |
| QRCodeEncode .....  | 38 |
| QRCodeEncode2 .....   | 39 |
| QRCodeEncode2W .....  | 40 |
| MicroQRCodeEncode .....                                     | 40 |
| MicroQRCodeEncode2 .....                                    | 41 |
| HIBC_QRCodeEncode .....                                     | 42 |
| HIBC_QRCodeEncode2 .....                                    | 43 |
| QRCodeResultGetVersion .....                                | 43 |
| QRCodeResultGetBarcodeString .....                          | 44 |
| DestroyQRCodeEncodeResult .....                             | 44 |
| PaintQRCodeImageRaster .....                                | 45 |
| PaintQRCodeImageVector .....                                | 46 |
| PaintQRCodeImageRaster2 .....                               | 46 |
| PaintQRCodeImageVector2 .....                               | 47 |
| PaintQRCodeImageEMF .....                                   | 48 |
| QRCodeGetErrorMessage .....                                 | 49 |
| QRCodeResultGetBarcodeString2 .....                         | 49 |
| PaintQRCodeImageClipboard .....                             | 49 |
| QRCodeGetParity .....                                       | 50 |
| 9. QRCode ActiveX Reference .....                           | 51 |
| 9.. Specification .....                                     | 51 |
| 9.. Enumerations .....                                      | 51 |
| 9.. Properties .....  | 51 |
| 9.. ActualVersion Property .....                            | 52 |
| 9.. BackColor, ForeColor Properties .....                   | 52 |
| 9.. ECLevel Property .....                                  | 53 |
| 9.. Mask Property .....                                     | 53 |
| 9.. ModuleSize Property .....                               | 54 |
| 9.. Picture Property .....                                  | 54 |
| 9.. QRType Enumeration .....                                | 54 |
| 9.. QRType Property .....                                   | 55 |
| 9.. TargetDPI Property .....                                | 55 |
| 9.. Text Property .....                                     | 55 |
| 9.. Version Property .....                                  | 56 |
| 9.. CopyToClipboard Method .....                            | 56 |
| 9.. ExportImageRaster Method .....                          | 56 |

|   |    |
|---|----|
| 9.. ExportImageVector Method .....                              | 57 |
| 10. Adding QRCode Symbols to SQL Server Reporting Service ..... | 59 |
| Custom Assembly .....   | 59 |
| Installing Custom Assembly .....                                | 59 |
| Adding QRCode to Report .....                                   | 60 |
| Deployment .....  | 61 |
| 11. Technical Support .....                                     | 63 |
| A. Font Character Set .....                                     | 65 |
| B. How to Calculate Parity Value in Structural Append .....     | 67 |
| C. Version ID Parameter (updated in version 5.2) .....          | 69 |
| Modifier byte .....   | 69 |
| Version byte .....  | 69 |
| Example .....   | 69 |
| D. Unicode String Encode Support .....                          | 71 |
| E. Center Overlay (Feature 50) Support .....                    | 73 |
| E.. Specify bit pattern .....                                   | 73 |
| Limitations .....   | 73 |
| E.. Swiss Payment QR Code .....                                 | 73 |
| Font and Font Size .....  | 74 |
| F. Fontware License Agreement .....                             | 77 |



# List of Figures

|   |    |
|---|----|
| 3.1. QRCode Encoder GUI Options .....                                     | 13 |
| 3.2. Export Options Dialog .....  | 15 |
| 3.3. Parity Calculator Dialog .....                                       | 16 |
| 6.1. Editing QRCode Activex properties in Office programs .....           | 29 |
| 6.2. Using LinkedCell Property to Hookup with Excel Data .....            | 30 |
| 6.3. Adding QRCode control to Custom Controls in Microsoft InfoPath ..... | 31 |
| A.1. QRCode Font Character Set .....                                      | 65 |





## List of Tables

|  |    |
|--|----|
| 2.1. List of common ECI values .....                                 | 8  |
| 2.2. QRCode Font Characteristics .....                               | 11 |
| 5.1. List of Crystal Reports UFL Functions ( cruf1QRCode5.d11) ..... | 23 |
| 6.1. List of VBA Functions (Morovia.QRCodeFontDLL5.bas) .....        | 27 |
| 7.1. Error Codes .....   | 35 |
| 8.1. List of Enumerations .....                                      | 37 |
| 8.2. List of Functions .....   | 37 |
| 8.3. ImageTypeEnum Enumeration .....                                 | 38 |
| 9.1. QRCode ActiveX Specification .....                              | 51 |
| 9.2. List of QRCode ActiveX Enumerations .....                       | 51 |
| 9.3. List of QRCode ActiveX Properties .....                         | 51 |
| 9.4. List of QRCode ActiveX Methods .....                            | 52 |
| 9.5. ECLevel options .....   | 53 |
| 9.6. Mask options .....  | 53 |
| E.1. QRCode Font and Size Selection .....                            | 74 |
| E.2. Swiss Font and Size Selection .....                             | 75 |



# Chapter 1. Introduction

Morovia QRCode Fonts & Encoder 5 is the ultimate tool box to print *QR Code* and *Micro QR* symbols. A QR Code is a matrix two-dimensional code developed by Japanese corporation Denso-Wave in 1994. The “QR” is derived from “Quick Response”, as the creator intended the code to allow its contents to be decoded at high speed. QR Codes are common in Japan, where they are currently the most popular type of two dimensional codes. Moreover, most current Japanese mobile phones can read this code with their camera.

QR Code is capable of encoding large amount of data - 7,089 numeric characters, 4,296 alphanumeric, 2,953 bytes or 1,817 Kanji text. The compact version, Micro QR, is capable of encoding 35 digits, 21 alpha-numeric or 15 bytes.

HIBC QR Code is the version adopted by *HIBC* (Health Industry Barcode) to create barcodes on health items.

This software supports QR Code model 2, described in ISO/IEC 18004:2006.<sup>1</sup>

This package includes the following contents:

- One true type fonts targeting 600 dpi laser printers - `mrvqrcode.ttf`.
- The user manual, which you are reading on.
- QRCode Encoder GUI, a GUI program to create barcode strings based on data entered. The program can export barcode images in a variety of formats, such as *PNG*, *EMF*, *SVG* and *EPS*.
- A Windows native DLL that allows you to add QRCode printing to your own application.
- A Crystal Reports extension DLL that adds QR Code printing functionality to Crystal Reports.
- Microsoft Word Plug-in to create QR Codes in Word documents.
- SQL Server Reporting Service Plug-in that supports 32-bit and 64-bit SSRS.
- An ActiveX Control that can be inserted into Microsoft Office programs or integrated into your custom application.
- Examples demonstrating how to add QR Code printing functionality to your applications in a variety of programming environments, such as Access, Visual C++ and .Net.

## Installing Morovia QRCode Fonts & Encoder 5

In order to provide seamless application experience, the installer is greatly enhanced. The setup actually comprises two installers - the 32-bit version that installs 32-bit only files, and 64-bit version that installs both 32-bit and 64-bit components. Therefore, in any supported Windows system you always get the right files installed.

### **To Install from a CD**

1. Insert the program CD into your CD drive. The setup starts automatically. Or if the auto-run feature isn't enabled on your system, click the Windows Start button and choose the Run command. Type **D:\Setup.exe** in the dialog box and click the OK button (Note that D represents the letter assigned to your CD-ROM drive. If your drive is assigned to a different letter, use it instead of D).
2. Follow the on-screen instructions.
3. You will be prompted to enter the **License To and Registration Code**. The **License To and Registration Code** information are found on the back of the CD case.

---

<sup>1</sup>ISO/IEC 18004 was first published in 2000, and revised in 2006. QR Code symbols conforming to the latest standard are referred as QR Code 2005.

### **To Install from Direct Download**

1. Click the Download link to start the download.
2. When the browser prompts, do one of the following: A. To run setup immediately, click `Open or Run This Program from Its Current Location`. B. If you decide to run the setup at a later time, click `Save or Save This Program to Disk`.
3. If you chose `Save This Program to Disk` in Step 2, locate the file where you saved it, and double click on it.
4. Follow the setup instructions.
5. You will be prompted to enter the `License To/Registration Code`. The `License To` and `Registration Code` information can be found in the email we send to you after order completes.

## **Limitations of Trial Version**

A trial version is provided on our web site that can be downloaded freely. Barcodes created by the trial version have extra “DEMO” added at the end. All other functionality is identical between the two.

## **Backward Compatibility**

Version 5.0 is a major upgrade to the QR code fonts product line. Based on our release policy, minor update (reflected in the change of minor version number) will always remain backward compatible with the ones under the same major version. However, a major upgrade may break backward compatibility for the sake of feature enhancement. If you are upgrading from the previous version (QRCode Fontware 1.x), read this section carefully when planning the upgrade.

In 5.0 release, all executables have both 32-bit and 64-bit versions. Unfortunately to accommodate this feature enhancement, we have to break the backward compatibility. The following lists the changes that break backward compatibility.

### **Font Encoder DLL**

The DLL interface has changed in this version. In this version you call `encode` function to create a result object, and create image files or barcode string through this result object.

If your program calls the DLL API, you must change the code accordingly.

### **Crystal Reports**

In order to have the single source code for both 32-bit and 64-bit UFLs, the UFL interface is changed. Reports authored in previous 1.x version require update in order to use the new version.

### **Input Format Change for Structural Append**

In this version, the input format for QRCode structural append feature is enhanced to allow user to specify parity value. Parity is shared across the symbols and provides a clue for scanners to recognize the symbols within the same group. See the section called “Structural Append (SA)” for details.

### **Unicode Encoding Support**

This feature is introduced in version 5.1 release. It adds support for encoding Unicode string in a way compatible with iPhone and Android readers. If your default locale is western Latin1 and only characters in this set are encoded, there is no breaking changes. Otherwise, the QRCode produces may look different on

components that accept UTF-16 (`wchar_t`) input. More information is available at Appendix D, *Unicode String Encode Support*.



# Chapter 2. Overview

This chapter briefly reviews the symbologies (a.k.a. barcode formats) supported by QR Code Fonts & Encoder 5. QRCode Fonts & Encoder 5 supports normal QR Code, Micro QR Code and HIBC QR Code (which is based on normal QRcode).

## What is QR Code?



QR Code



Micro QR Code

QR Code is capable of handling all types of data, such as numeric and alphabetic characters, Kanji, binary bytes and control codes. Up to 7,089 digits can be encoded in one symbol.

QR Code has undergone two major revisions - Model 1 and Model 2. It is generally believed that the two models are not compatible with each other. The ISO/IEC 18004:2006 defines Model 2, often referred as QR Code 2005.

This software produces QR Code symbols compliant with Model 2 only.

QR Code supports several advanced features not present in many other barcode formats, such as Extended Channel Interpretation (ECI), FNC1 and Structural Append (SA). The advanced features require special input format. For more information, see the section called "Structural Append (SA)".

HIBC QRCode is a variant of QR Code. It uses QR Code to encode HIBC LIC. When this option is specified, our program automatically adds + sign at the beginning and appends a checksum digit at the end, then encodes them as a whole.

The Micro QR Code format (also specified in this International Standard), is a variant of QR Code 2005 with a reduced number of overhead modules and a restricted range of sizes, which enables small to moderate amounts of data to be represented in a small symbol, particularly suited to direct marking on parts and components, and to applications where the space available for the symbol is severely restricted. Micro QR is capable of encoding up to 35 digits, 21 alphanumeric characters, 15 bytes or 9 kanji characters.

### **Encoding Character Set**

QR Code is capable of encoding four types of data, as listed below. Note that the interpretation of byte characters is subject to the ECI, which is assumed to be ISO8859-1 if no ECI is defined in the symbol.

- Numeric characters: 0-9.
- Alphanumeric characters: digits 0-9, all upper case letters (A-z) and 9 punctuation symbols (space, \$, %, \*, +, -, ., : and /).
- Byte characters: single byte (ASCII value 0-255).
- Kanji characters. A Kanji character consists of two bytes and has a value either between 0x8140 and 0x9ffc, or between 0xe040 and 0xebbf. In the software, they must be enclosed in a special format block called Kanji block. Read further for more information on Kanji block.

QR Code defines several encoding modes, with the numeric set the most efficient and the Kanji set the least. Internally, the encoder switches the modes automatically based on the nature of the data, with the goal of minimizing the symbol size required. The algorithm produces the optimal results in most cases.

When space is a concern, it is desirable to adjust the character set of data encoded so that they can be more efficiently encoded. For example, the alphanumeric character set in QR Code does not include lower case letters. If the string encoded is case insensitive, it reduces space requirement by converting all letters upper case.

QR Code also defines several advanced features that can only be encoded with escaped format, referred as `tilde codes` throughout the manual. These advanced features include:

- *ECI*. ECI was proposed by AIM to give a unified interpretation to the byte data encoded in a barcode symbol, thus allowing different character set of data to be mixed in a symbol.
- *Structural Append*. Structural Append allows data to be split into multiple symbols. Beside to the data, those symbols encode additional information such as sequence number, total number of symbols, and parity value. A capable reader is able to read them randomly and put the data together.
- *FNC1*. GS1 uses this special character to indicate that a GS1-compatible data encoded in the symbol.

## Options

Three options affect the final shape of the symbol generated:

### Symbol Version

QR Code has a total of 40 different sizes. Each size is referred as *Version*, ranged from 1 to 40. Version 1 consists of 21 x 21 modules, and version 40 consists of 177 x 177 modules. Each higher version number comprises 4 additional modules per side. Each QR Code symbol version has the maximum data capacity according to the amount of data, character type and error correction level.

There are four versions available in Micro QR - M1, M2, M3 and M4. This software allows version request greater than 4 in Micro QR; however, only the maximum version allowed is realized.

By specifying 0 as the version, the software selects the minimum version that fit the data encoded. Furthermore, if the version requested is too small, the software also increases it automatically to fit the data encoded.

Note: in version 5.2, the Version parameter is enhanced to include a modifier byte to fine tune encoder behavior. See Appendix C, *Version ID Parameter (updated in version 5.2)* for more information.

### Error Correction Level

QR Code has error correction capability to restore data if the code is dirty or damaged. Four error correction levels are available for users to choose according to the operating environment. Raising this level improves error correction capability but also increases the amount of data QR Code size.

To select error correction level, various factors such as the operating environment and QR Code size need to be considered. Level Q or H may be selected for factory environment where QR Code gets dirty, whereas Level L may be selected for clean environment with the large amount of data.

The recovery capacity of the four levels are approximately 7%, 15%, 25% and 30%.

In case of Micro QR, not all error correction level is available to all versions. For example, level Q is only available on version M4. In this case, our software bumps the version automatically to accommodate the error correction level.

### Data Masking

The read reliability of a QR Code symbol can be improved through a method called *data masking*. Seven bit patterns (000 - 111) can be used to mask a symbol. Our software provides another value, -1, which selects the mask pattern that produces the best result.



## Input Format

---

**Note** The input format specified in this section applies on normal QR code and Micro QR only. HIBC QR Code only accepts the characters defined in HIBC character set.

---

QR Code supports several advanced features such as ECI and Structural Append. Those features can only be achieved using tilde codes outlined in this section.

### ~dnnn

When nnn corresponds to a numeric value between 0 and 255, the tilde code sequence represents a character with value equal to nnn. For example, ~d032 represents a space character.

~~

Represents a tilde (~) character.

~1

Represents a FNC1 character, first position. Note: this feature is not available in Micro QR.

~2

Indicates that a structural append control block follows. Must appear at the beginning of the input. Note: this feature is not available in Micro QR.

~3

Represents a FNC1 character, second position. See notes below for more information. In order for the program to find the AI, the AI must follow immediately and be placed inside a pair of square brackets. The standard implies that the AI must be a numeric value between 0 and 255. For example ~3[37]. Note: this feature is not available in Micro QR.

~4

Starts a Kanji block. The Kanji characters must be enclosed in brackets and follow immediately.

~50

Starts a center overlay block. The center overlay feature allows a pattern to be placed on the center of the QR code.

~7, ~8 and ~9

Indicates the start of an ECI block. This escape sequence must be followed by exact 6 digits, which corresponds to the ECI value. See the section called "Extended Channel Interpretation (ECI)" for more information.

~X

Represents a character value from 0 to 26. Replace the X like in the following example ~@ means character ASCII 0, ~A means character 1, ~B means character 2, ~C means character 3 ...

Note that ~dnnn sequence is allowed to appear within a Kanji block, but not in other blocks such as Structural Append. For example, ~2[1][6] is a valid SA block, but ~2[1][~d54] is not. ~dnnn should not be used in ECI block either.

### Center Overlay Block

The Center Overlay Block feature, also referred to as feature 50 in this manual, allows user to specify a bit pattern block that is overlaid on the center of the QR Code. This feature is required to implement QR Code for Swiss payment, where a Swiss logo is required to overlay on the center of the barcode.

Because QR Code has error correction capability, overlaying a small image won't affect the reading. However, care must be taken to avoid situations that the image is too big to cause the whole barcode unreadable.

The following are the two examples of center overlay block. In the first one, the overlaid bit pattern is expressed as rows, cols and pattern. In the second one, `:swiss7` indicates to overlay 7 mm x 7mm swiss logo when the overall symbol size is 46mm x 46 mm (the size required by Swiss payment standard). Both starts with tilde code 50, so this feature is also referred as Feature 50.

```
~50[7,7,007c6c446c7c]
```

```
~50[:swiss7]
```

For more information about Center Overlay Block, see Appendix E, *Center Overlay (Feature 50) Support*.

### **Kanji Block**

QRCode can encode Kanji text natively. A Kanji character consists of two bytes and has a value either between 0x8140 and 0x9ffc, or between 0xe040 and 0xebbf. Kanji characters are inputted using `kanji block`, which starts with `~4` following by the byte values of kanji characters enclosed in square brackets. Byte values can be

escaped using `~dnnn` format. For example, the following code is the input for text **89日本**

```
89~4[~d147~d250~d150~d123]
```

The Kanji characters have hex values 0x93fa and 0x967b - whose corresponding decimal values are 147, 250, 150 and 123.

### **Extended Channel Interpretation (ECI)**

ECI was introduced to allow mixed character sets in a QRCode symbol, and to enable different interpretations on the output data stream.

An ECI can be any number between 000000 and 999999. The tilde code sequence `~7nnnnnn` is used to enter an ECI value. The code sequence can appear at any places of the input, but there must be exact 6 digits following `~7`. For example to start an interpretation of 10, enter `~7000010`.

In addition to `~7` (*ECI basic*), `~8` and `~9` can be used to represent *Nesting-Begin* and *Nesting-End*. The Nesting-Begin ECI does not turn off the current ECI in force until Nesting-End is encountered at the current nesting level. This allows multiple ECIs (for example, Greek language plus encryption) effective in a data segment.

The table below lists common ECI values and their corresponding character encoding. Note that some character encodings may have multiple ECI values. For example, you can use either 1 or 3 for ISO8859-1 encoding.

**Table 2.1. List of common ECI values**

| Character encoding | ECI value |
|--------------------|-----------|
| CP437 (DOS)        | 0, 2      |
| ISO8859-1          | 1, 3      |
| ISO8859-2          | 4         |
| ISO8859-3          | 5         |
| ISO8859-4          | 6         |
| ISO8859-5          | 7         |
| ISO8859-6          | 8         |
| ISO8859-7          | 9         |

| Character encoding | ECI value |
|--------------------|-----------|
| ISO8859-8          | 10        |
| ISO8859-9          | 11        |
| ISO8859-10         | 12        |
| ISO8859-11         | 13        |
| ISO8859-12         | 14        |
| ISO8859-13         | 15        |
| ISO8859-14         | 16        |
| ISO8859-15         | 17        |
| ISO8859-16         | 18        |
| Shift_JIS          | 20        |
| Windows-1250       | 21        |
| Windows-1251       | 22        |
| Windows-1252       | 23        |
| Windows-1256       | 24        |
| UTF-16BE           | 25        |
| UTF-8              | 26        |
| US-ASCII           | 27, 170   |
| Big5               | 28        |
| GB2312             | 29        |
| EUC-KR             | 30        |

---

**Default ECI** In 2005 ISO standard, the default ECI is ISO8859-1. In the 1997 AIM standard and 2000 ISO standard, the default ECI corresponds to the JIS8/ShiftJIS character set.

---



---

**ECI and Structural Append** The 2005 standard makes it necessary to add ECI values after Structural Append for each ECI in force in individual symbols. This implies that ECI in force does not cross symbol boundary. This behavior is departed from the 1997 AIM standard.

---

### **Structural Append (SA)**

The structural append feature allows up to 16 symbols in a structure. A capable reader buffers the contents of each symbol until all symbols are read.

To encode structural append, supply the three required fields for each symbol:

- Symbol Position (SP). The symbol position is 1-based index which identifies the position of this particular symbol in the group. Can be any number between 1 and 16.

- Total Number of Symbols (TS). This value indicates the number of total symbols. Can be any number between 1 and 16. The value should be consistent among all symbols in the group.
- Parity (P). This value must remain the same across all symbols in the group. This value is derived from the full input data. The font encoder DLL and Crystal Reports UFL provide functions to calculate the parity value. See Appendix B, *How to Calculate Parity Value in Structural Append* for details. The Parity Calculator in the GUI encoder also can be used to calculate the parity value.

The tilde code sequence is expressed in the following format:

```
~2[SI][TS,P]
```

For example, tilde code sequence ~2[1][6,89] indicates that the current symbol belongs to a group that has 6 symbols in total. The parity value for input is 89.

The structural append block must appear at the *beginning* of the input.

Structural append is not supported in Micro QR code.

### **FNC1 (First Position)**

FNC1 (first position) is used when encoding GS1 structural data (GS1-128) in QR Code symbols. The leading FNC1 symbol is required to indicate the data followed is a GS1 structured data.

When encoding GS1 structured data, it is user's responsibility to ensure that the data supplied is correct (data length and checksum), and a separator is present after a variable length data item. QRCode standard requires % character to be used as separator, instead the FNC1 character as in the case of Code 128.

For example, to encode GS1 structural data (01)04912345123459(15)970331(30)128(10)ABC123, which denotes GT IN number 04912345123459, Sell By date 97/03/31, Quantity 128 and Lot number ABC123, the input should be:

```
~101049123451234591597033130128%10ABC123
```

FNC1 First Position is not supported in Micro QR code.

### **FNC1 (Second Position)**

FNC1 (second position) is an AIM extension to encode structural data similar to GS1. The encoder needs to know the AI in order to create an encoding. Therefore, it is required to put AI in brackets immediately after tilde code ~3.

FNC1 Second Position is not supported in Micro QR code.

### **Unicode String Support**

QR Code 2005 does not support unicode natively. The default character set is ISO8859-1. Theoretically the support could become available using ECI; however most scanners and applications do not have ECI support builtin.

Recently in mobile industry the trend is to use UTF8 to encode Unicode string into QR Code. Such feature requires support of the reader. We decided to add Unicode support in a way compliant with Android/iPhone barcode reader. Therefore, you need to have compliant reader in order to decode properly.

This feature is added in 5.1, released in May 2012. Enhancement is made for components that take UTF16 string input, such as QR Code ActiveX control and Word Add In. If all characters in the input string falls into ISO8859-1, the resulted barcode remains the same. Otherwise the input is converted to UTF8 with BOM first, and then encoded. In Font encoder DLL, additional function QRCodeEncode2W is added to perform such operation. More information is available at Appendix D, *Unicode String Encode Support*.

---

**Note** UTF-8 encoded QR code gains popularity as it encoded the full Unicode character set without using ECI. Many scanners can be configured to read in UTF-8

mode, and mobile phones read them universally due to open source zxing library. Started from version 5.3.3 (release in March 2022), unicode APIs always convert unicode strings to UTF-8. This change may break existing applications, see more information at Appendix D, *Unicode String Encode Support*.

## Working with QRCode Fonts & Encoder 5

This software provided three methods to integrate QRCode printing functionality to various applications - font, image and ActiveX. Depending on your working environment, you will find the best approach that fits your need.

### Font-Based



Creating barcodes using fonts involves two distinct processes: *encoding* and *rendering*. Rendering is to choose the appropriate font and font size and format the encoding results. Encoding is to convert the data into a special string, referred as *Barcode String*, which becomes a barcode after being formatted with the font.

**Note** In almost all circumstances, you cannot just type your number and format with a barcode font to create a valid barcode. You must generate the barcode string first, and format the barcode string with the font.

The font-based approach is easy to understand, and in some occasions the only way to add barcode printing functionality, such as in Crystal Reports.

The drawing below illustrates how you create a QR code using font-based approach. You first create the barcode string (which is an array of text lines). Format the barcode string with MRV QRCode font you get the barcode.

```
F8BBB8F0F9D9C0F8BBB8F
E2AAA2E06535B0E2AAA2E
A8178FA01B2E413E3DE9B
BAAA22B85CE1FEC10345E
F0EEEE0F0942E41BEB6BDC
888888808880888000888
```

The font characteristics are listed as below:

**Table 2.2. QRCode Font Characteristics**

| Filename      | Typeface   | Module With (under 6 points) |
|---------------|------------|------------------------------|
| mrvqrcode.ttf | MRV QRCode | 20 mils (0.5mm)              |

### Encoding

To create a valid barcode, you need to call an encoder to get a special string, and format this string with our QRCode font. To get this string, you need to call an encoder - you can run QRCode Encoder GUI, and obtain the result from this GUI program. Or if you need to bulk generate the results, using the programming interface exposed from the encoder DLL.

## QRCode Encoder GUI

This GUI program allows you to quickly create barcode string and transfer it to other programs. You are able to enter the data encoded, and create the barcode on the fly. For more information, see Chapter 3, *Using Encoder GUI*.

## MoroviaQRCodeFontEncoder5.dll

This DLL is a standard Windows DLL that can be called by many programming environments, including Microsoft Office, C++, FoxPro and .Net. Most programming environments support Windows DLL directly. We provide a VBA module and a C# class that wrap around the DLL functionality.

## cruf1QRCode5.dll

This DLL is a crystal reports extension DLL which adds the encoding functionality to Crystal Reports. For more information on using it in Crystal Reports, see Chapter 5, *Adding QR Code Symbols to Crystal Reports*.

---

**Note** Barcode string support is provided through MoroviaQRCodeFontEncoder5.dll and cruf1QRCode5.dll (which is a COM component). If your programming environment only support calling COM objects, such as PHP and VBScript, refer to KB10621<sup>1</sup> on how to get bar code string through COM.

---

## Image File Based

You can export barcodes as standard image files to be consumed in other word processing and imaging programs. The QRCode Encoder GUI supports exporting images in four formats via a GUI interface. The Encoder DLL and QRCode ActiveX Control also provide programming interfaces to export images in those formats:

- PNG (Portable Network Graphics)
- BMP (Windows Bitmap File)
- EMF (Enhanced Meta File)
- SVG (Scalable Vector Graphics)
- EPS (Encapsulated PostScript)

Special attention should be paid when the target printer has low resolution, such as a thermal printer or fax machine. When generating vector graphics files, the software allows you to specify target dpi which you should set to the one that matches your printer. It is generally not recommended to scale the image down or up when working with low-resolution printers.

## ActiveX Control

For environments that support ActiveX Controls (also called OLE controls), you can insert the QRCode ActiveX Control into documents, spreadsheets, or invokes the control at the background in your program. With ActiveX Control you are able to change the barcode image by editing its properties.

Using ActiveX Control in Microsoft Office programs is straightforward. Programming knowledge is required in order to integrate the control with your custom programs.

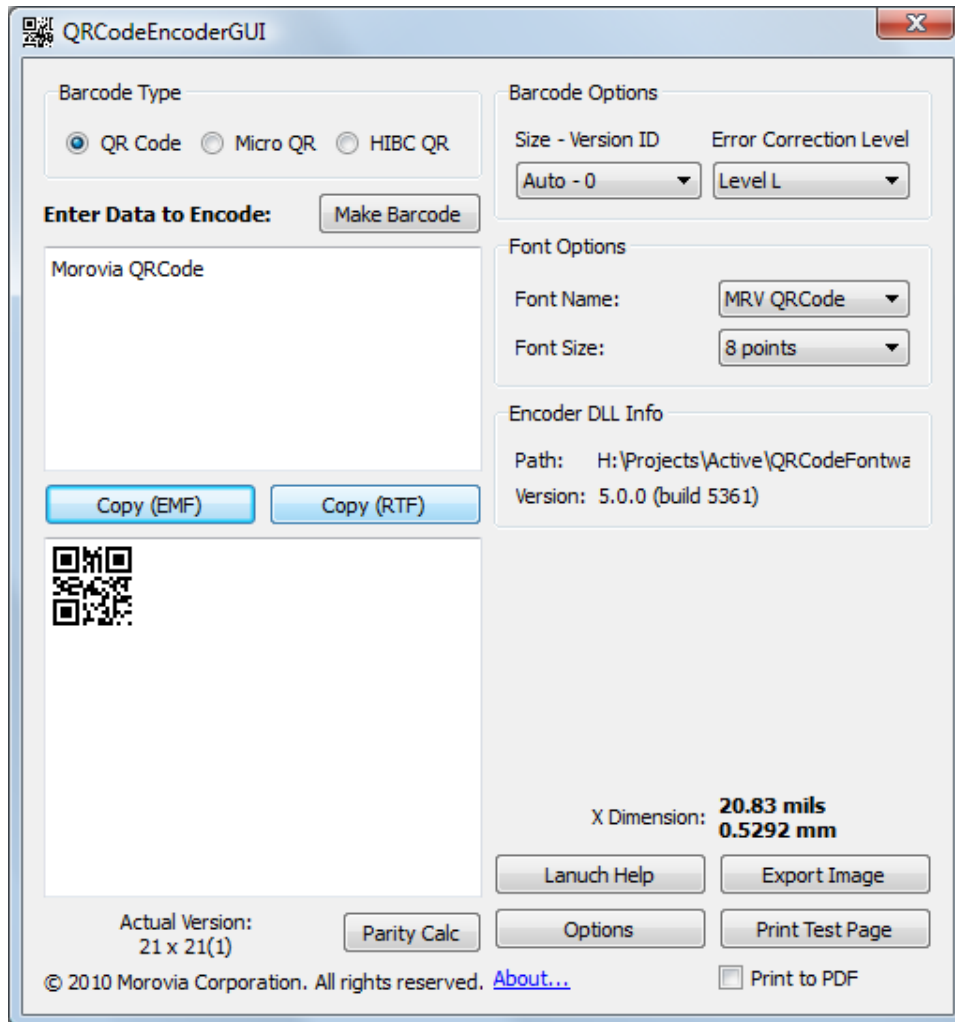
---

<sup>1</sup> <http://mdn.morovia.com/kb/10621>

# Chapter 3. Using Encoder GUI

Morovia QRCode Encoder GUI is a GUI program that runs on Windows 2000 and above. It provides a fast way to create QRCode barcodes on the fly. Note that the program requires the fonts as well as the encoder DLL to be present in the system.

Figure 3.1. QRCode Encoder GUI Options



## Barcode Type

Choose the type of the barcode to make - QR Code and HIBC QR Code.

## Data to Be Encoded

This is the place to enter the data to be encoded.

## Copy (EMF)

Place the barcode image in the clipboard, so that you can subsequently paste it into another application such as Microsoft Word. The EMF created does not use font, so that you do not need to copy the font file when you view document from another computer.

Note that X dimension in the resulted barcode is determined by the Module Size in the Options dialog, as the EMF format does not use the font.

### Copy (RTF)

Place the barcode string in the clipboard, so that you can subsequently paste them into another application such as Microsoft Word.

QRCode Encoder GUI places both text and RTF formats into the clipboard. In applications that are capable of processing *RTF*, you will see a barcode immediately after pressing the paste button. Otherwise, you will see the text string instead. If this is the case, highlight the whole string and format with “MRV QRCode” font.

### Version Info

Displays the realized version in the QR Code symbol.

### Parity Calc

Click on this button to pop up the parity calculator, which you can calculate the parity on the source data.

### Launch Help

Launch the HTML Help of this program.

### Options

Pops up the Options dialog where you can specify additional option for exporting barcode images. See the section called “Export Options” for more information.

### Export Image

Click on this button to export the barcode into standard image formats such as EMF, SVG, EPS and PNG.

### X Dimension

Displays the X-dimension of the barcode. The X dimension is displayed in both mils and mm.

### Encoder DLL Info

This section displays the absolute path and the file version of the encoder DLL. This is useful if you have multiple DLLs in the computer.

### Font Name

Select the font that applies on the barcode string. In this version, the only font selectable is “MRV QRCode”.

### Font Size

Select the font size that applies on the barcode string.

### Size - Version ID

The version of the QR code.

### Error Correction Level

The error correction level of the barcode. QRCode supports 4 levels - L, M, Q and H.

## Export Images

QRCode Encoder GUI supports four types of images. The images do not contain references to the fonts so that you can move around them without requiring the software to be installed.

- *EMF*. This is the default vector graphics format used on Windows operating system.
- *PNG*. PNG is a bitmap image format widely supported by imaging software. It is the recommended image format for web graphics.
- *BMP*. BMP is the default bitmap format used on Windows platform.

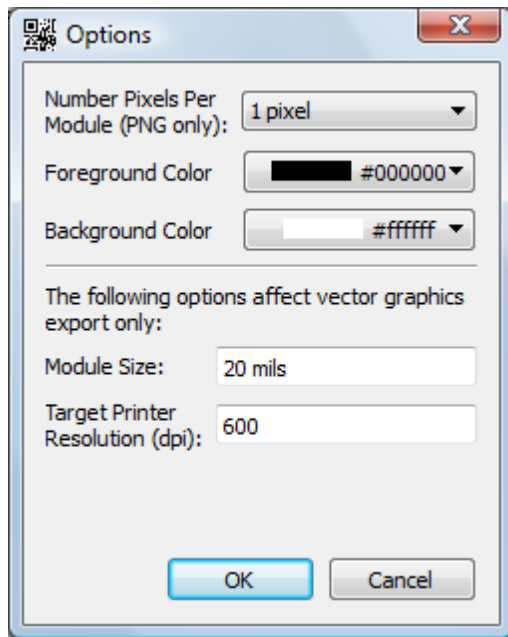


- *EPS*. This format is widely used in graphics design industry. Recommended to use in conjunction with Adobe software such as Adobe Illustrator.
- *SVG*. SVG is an emerging vector graphics format. FireFox 3 supports it out of the box, as well as many drawing programs such as Adobe Illustrator.

## Export Options

You can specify a couple of options applicable in exporting barcode images.

**Figure 3.2. Export Options Dialog**



### Number Pixels Per Module

This option applies on raster images only. Specify the number of pixels of a module (the real estate unit of a QRCode barcode).

### Foreground Color

The color of the dark modules, in RGB colorspace. To change the color, press the down arrow key to pop up the color picker.

### Background Color

The color of the white modules, in RGB colorspace. To change the color, press the down arrow key to pop up the color picker.

### Module Size

Apply on Vector image format only. You can specify a nominal size for the module length. You can enter something like "20 mils", or "0.02mm". If the unit of measure is not specified, unit of mils is assumed.

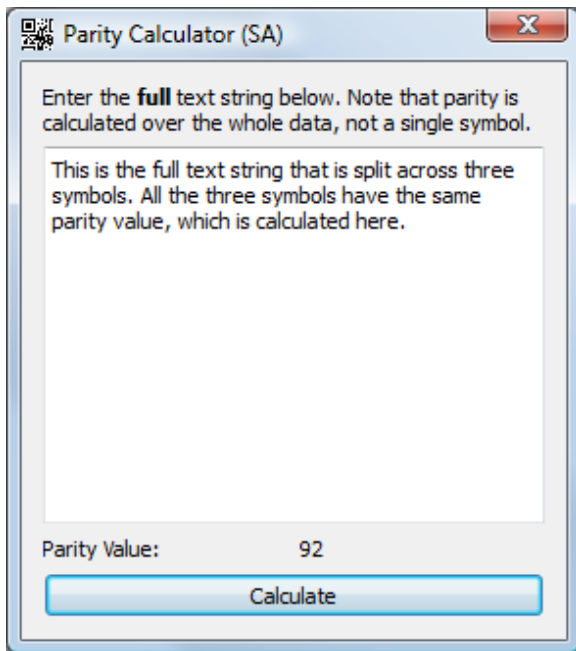
### Target Resolution

Apply on vector image format only. If you export a vector graphic images to a low resolution device such as screen or thermal printer, you should fill this field with the value of the resolution. For screen, use 96. This value makes sure that the length units are properly aligned to the edge of pixels when rasterized.

## Parity Calculator

You can use Parity Calculator to calculate the parity value of the full data encoded into a group of QRCode symbols using structural append feature. Note that the parity must remain the same across all symbols in the group. The input follows the section called “Structural Append (SA)”, except that the structural append block ~2 is not allowed.

**Figure 3.3. Parity Calculator Dialog**



# Chapter 4. Word Add-In

## Install/Remove Add-In

The Add-In is automatically installed unless you indicated not to do so.

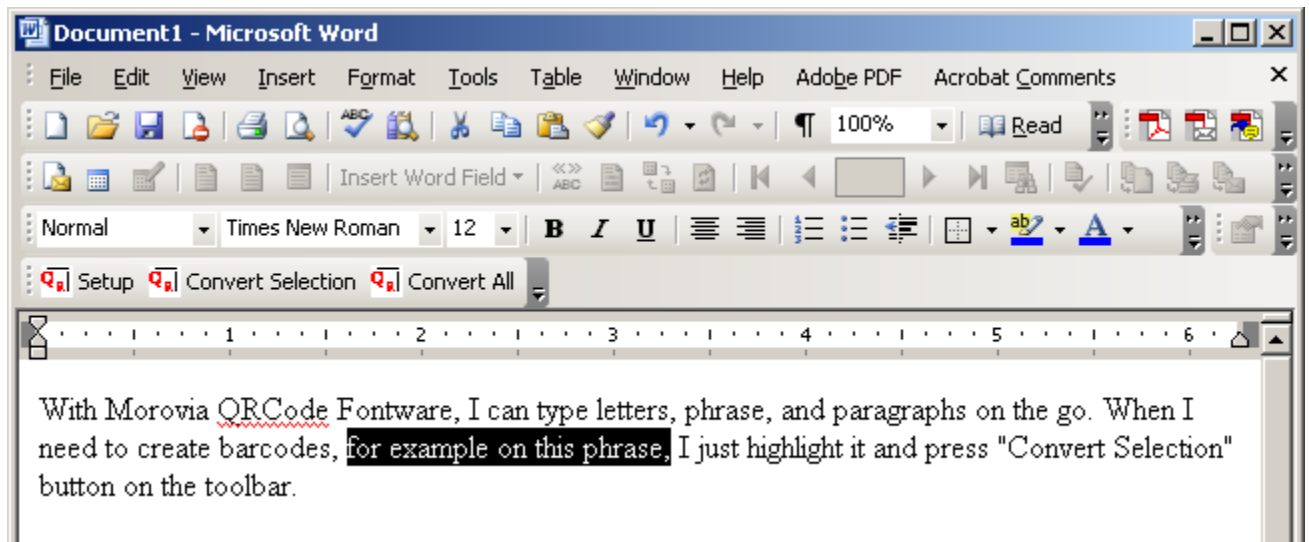
After the Add-In is installed, you should see three buttons on the toolbar. if they do not appear, right click on the tool bar area and check QRcodeAddin box.

In Word 2007, select Add-In and you will see the three buttons on the ribbon.

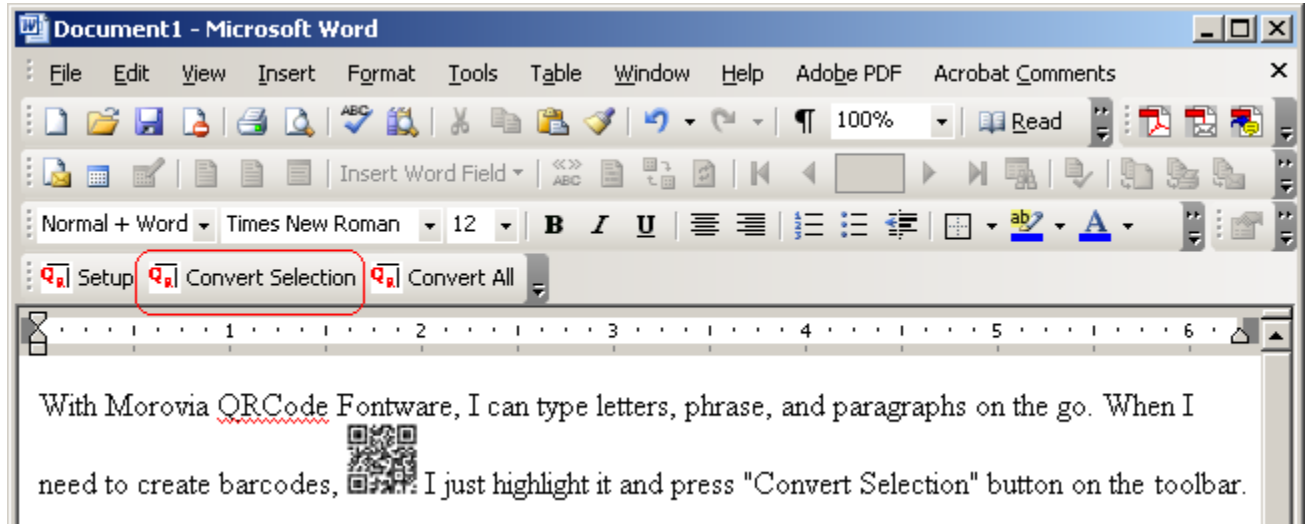
- Setup - click this button to pop up the setting dialog from which you can specify barcode type, module size and target resolution.
- Convert Selection - click this button to convert the current selected text into barcode.
- Convert All - use this button to convert all text enclosed in angle brackets (< and >).

## Creating Single Barcode

1. Enter a few strings line by line and highlight them.

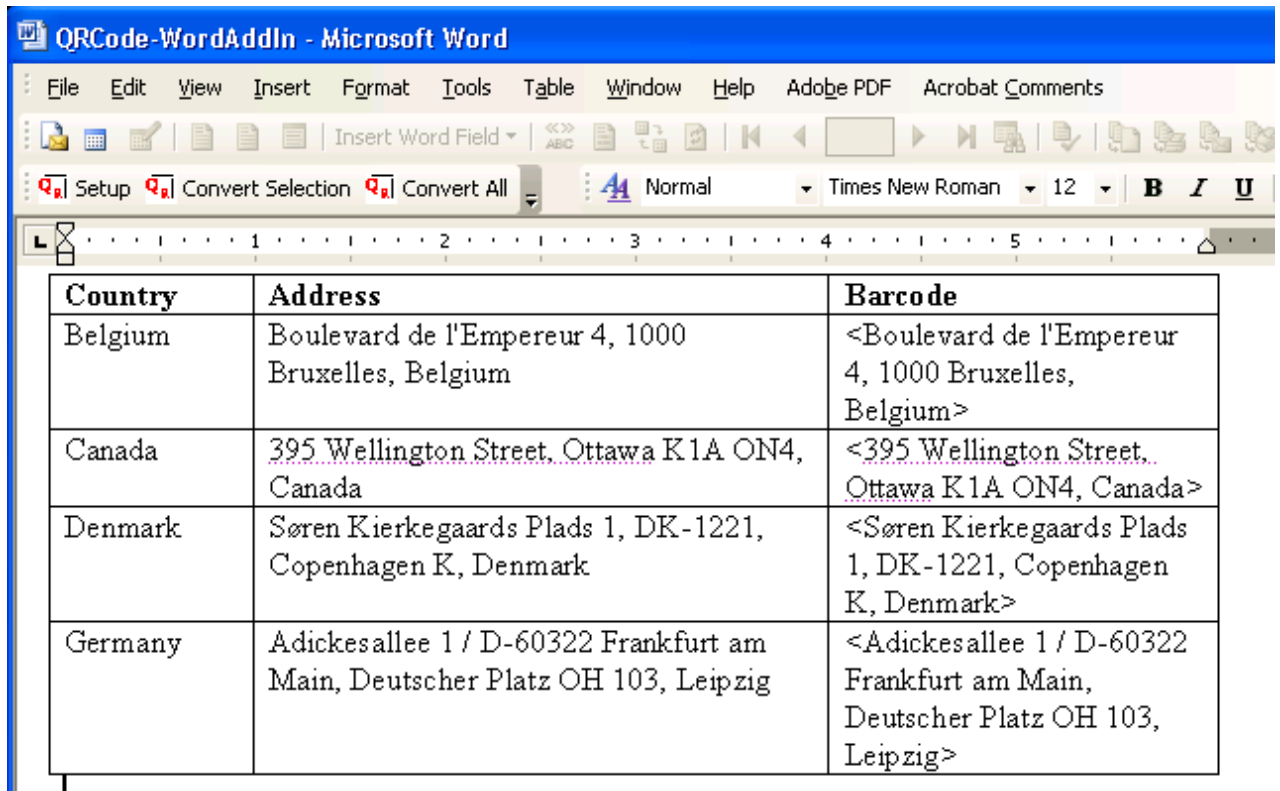


2. Click on Convert Selection to create a QRCode barcode.



## Creating Multiple Barcodes

1. Enter a few paragraphs, surround those paragraphs which will be converted to barcodes with the “<” and “>” characters.







2. Click on “Convert All” to create barcodes for paragraphs surrounded with the “<” and “>” characters.

QRCode-WordAddIn - Microsoft Word

File Edit View Insert Format Tools Table Window Help Adobe PDF Acrobat Comments

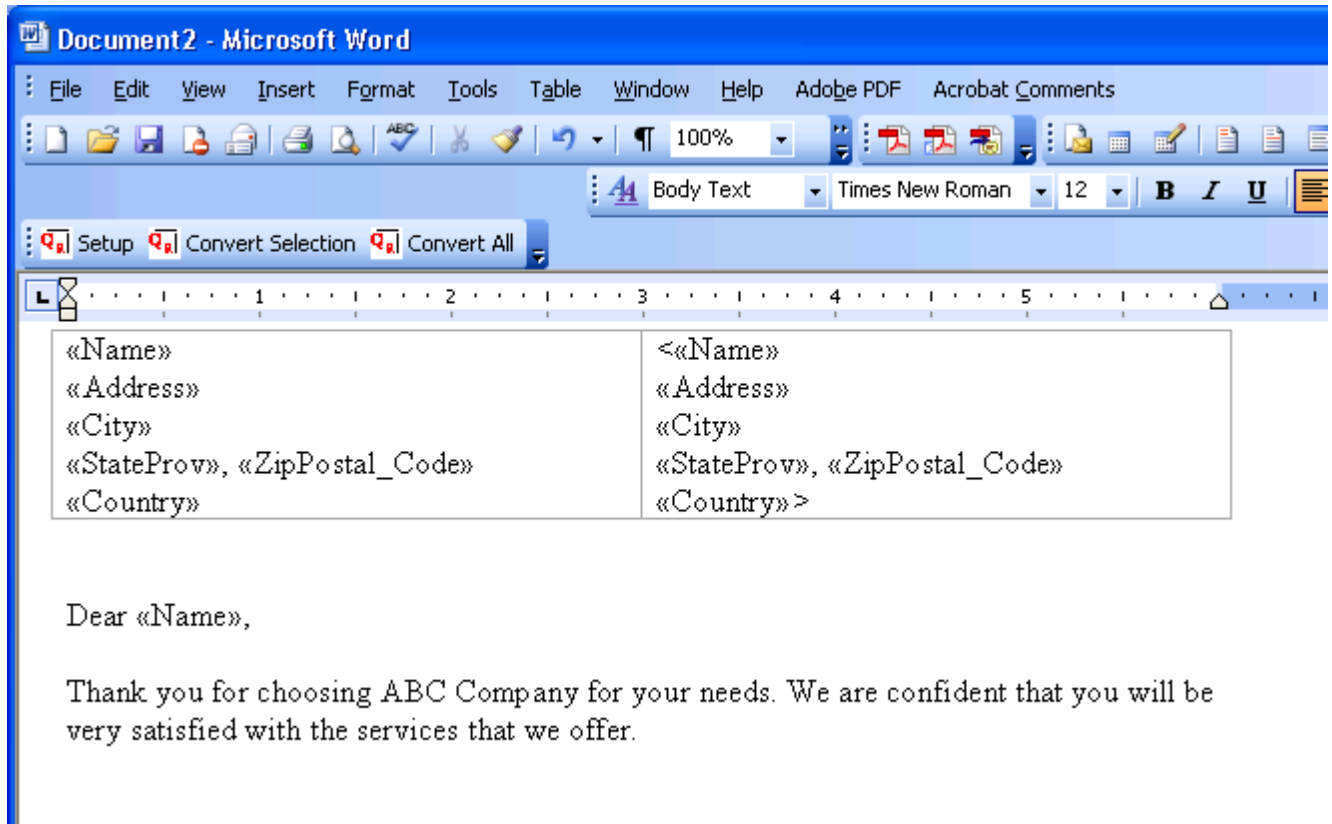
Insert Word Field

Setup Convert Selection Convert All Normal Times New Roman 12 B

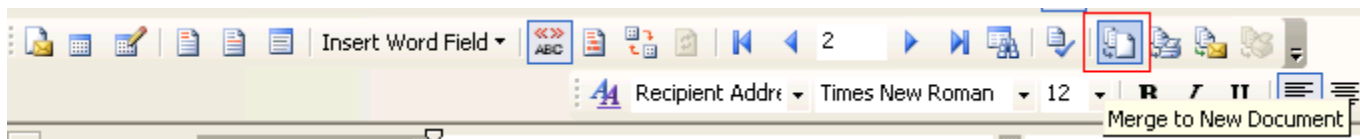
| Country | Address  | Barcode   |
|---------|--|---|
| Belgium | Boulevard de l'Empereur 4, 1000<br>Bruxelles, Belgium                          |    |
| Canada  | 395 Wellington Street, Ottawa K1A 0N4,<br>Canada                               |    |
| Denmark | Søren Kierkegaards Plads 1, DK-1221,<br>Copenhagen K, Denmark                  |   |
| Germany | Adickesallee 1 / D-60322 Frankfurt am<br>Main, Deutscher Platz OH 103, Leipzig |  |

## Mail Merge

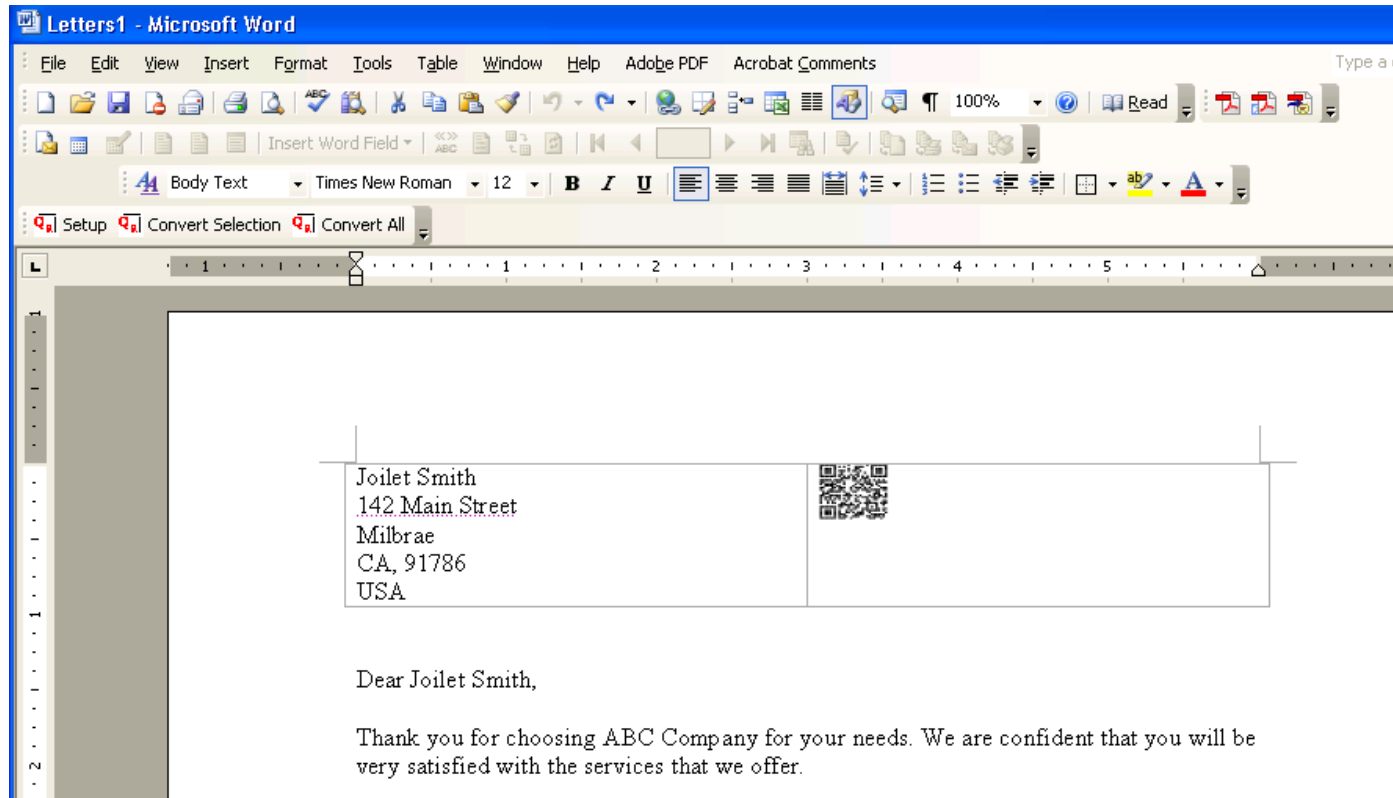
1. In Mail Merge, choose Access Example (ActiveX).mdb as the data source. Surround the paragraphs which will be converted to QRCode barcode with the ">" and "<" characters.



- Click on the Merge to New Document button to create a new document with all fields merged with database records.




- Click on "Convert All" to create QR Code barcodes for the paragraphs surrounded with the ">" and "<" characters.



The screenshot shows the Microsoft Word interface with a letter template. The title bar reads "Letters1 - Microsoft Word". The menu bar includes File, Edit, View, Insert, Format, Tools, Table, Window, Help, Adobe PDF, and Acrobat Comments. The ribbon shows the "Format" tab with options for font (Body Text, Times New Roman, 12), bold, italic, underline, and alignment. The status bar at the bottom indicates "Setup", "Convert Selection", and "Convert All".

The letter content is as follows:

|  |   |
|--|---|
| Joilet Smith<br>142 Main Street<br>Milbrae<br>CA, 91786<br>USA |  |
|--|---|

Dear Joilet Smith,

Thank you for choosing ABC Company for your needs. We are confident that you will be very satisfied with the services that we offer.





# Chapter 5. Adding QR Code Symbols to Crystal Reports

Adding barcodes to Crystal Reports is straightforward. The example included in the software was authored in Crystal Reports 9.

Note: the functions in this release are not backward compatible with old version QRCode Font & Writer SDK 1. You need to change the functions in order to use the new version API.

---

**Warning** Due to the length constraint imposed by Crystal Reports, the approach outlined in this chapter requires Crystal Reports version 9 and above, unless you are encoding fairly small amount of data (127 digits or 77 alphanumeric).

---

## User Function DLL

Crystal Reports extension DLL is included in the software (`cruf1QRcode5.d11`), which provides QR code encoding functions. By default, this file can be found under the installation directory. This is a COM DLL - so you must register it if you move the file to another computer. This is typically done by running `regsvr32 cruf1QRcode5.d11` command in a administrator console. In this release, both 32-bit and 64-bit versions are supplied. The 32-bit version is required to work with Crystal designer. The 64-bit is needed to work with 64-bit Crystal runtime. On 64-bit Windows, both DLLs are installed.<sup>1</sup>

This DLL exports three functions:

**Table 5.1. List of Crystal Reports UFL Functions ( `cruf1QRcode5.d11` )**

| Function Name  | Comment   |
|--|---|
| Number QRCodeEncodeSet (String, Number, Number)      | Encode the string specified with QRCode, and returns the number of chunks required to hold the barcode string result. The first parameter is data encoded, followed by minimum version required and error correction level.       |
| Number MicroQRCodeEncodeSet (String, Number, Number) | Encode the string specified with Micro QRCode, and returns the number of chunks required to hold the barcode string result. The first parameter is data encoded, followed by minimum version required and error correction level. |
| Number HIBCQRCodeEncodeSet (String, Number, Number)  | Encode the string specified with HIBC QRcode, and returns the number of chunks required to hold the barcode string result. The first parameter is data encoded, followed by minimum version required and error correction level.  |
| String QRCodeEncodeGet(Number)                       | Returns a chunk of barcode string. Each chunk consists of no more than 254 characters (the maximum number allowed by Crystal Reports).  |

---

<sup>1</sup>The 64-bit DLL is installed under `C:\Program Files\Morovia QRCode Fonts & Encoder 5` directory, while the 32-bit DLL can be found under `C:\Program Files (x86)\Morovia QRCode Fonts & Encoder 5`.

| Function Name                   | Comment   |
|---------------------------------|---|
| Number QRCodeGetParity (String) | Calculate the parity across all the data. This is useful when you encode a series of symbols using Structural Append feature in the report. |

Function `QRCodeEncodeGet` takes one argument which is the ordinal number of the chunk. By concatenating all the chunks together you get the full barcode string (which become a QRCode symbol after being formatted with the font).

The following script demonstrates the basic flow to get it work in Crystal Reports:

```
//Morovia example formula to display qrcode barcodes
StringVar CompleteBarcodeString:="";
StringVar DataToEncode:= "data to encode enter here.";
NumberVar i:=0;
NumberVar Segments:=
QRCodeEncodeSet(DataToEncode, 0, 0);
For i:=0 to Segments Do
(
    CompleteBarcodeString := CompleteBarcodeString +
        QRCodeEncodeGet(i);
);
CompleteBarcodeString
```

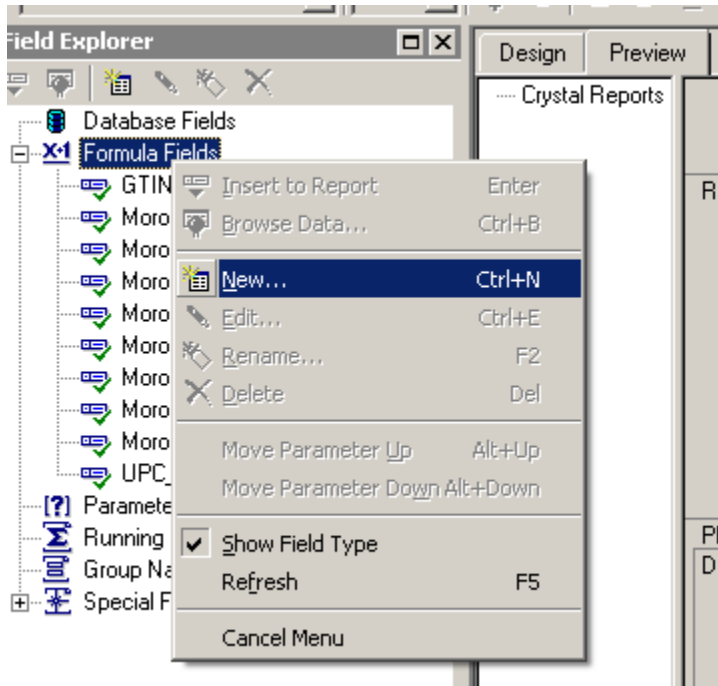
Because of the length restraint that UFL imposes on the string, it is not possible to complete in one UFL call. Instead, you prepare all parameters and call `QRCodeEncodeSet`. This function calls encoder at the backend, and returns the number of chunks required. After that the program enters a loop that calls `QRCodeEncodeGet` repeatedly until all result are retrieved and stored in `BarcodeString`.

## Working with Crystal Reports

The screen shots in the following tutorial were taken in Crystal Reports 9. Interfaces in other versions may appear differently.

Assume that you have a database table with several fields. You already put them into the report, and want to add a column to hold QRCode barcode for ID field.

1. First switch to design mode. Choose Report → Formula Workshop.
2. Right click on Formula Fields and choose New.
3. Give your formula field a name, in this example we will name it `Morovia_QRCode`. In versions 9 and above, you will be prompted to use the editor or the expert, choose Use Editor.

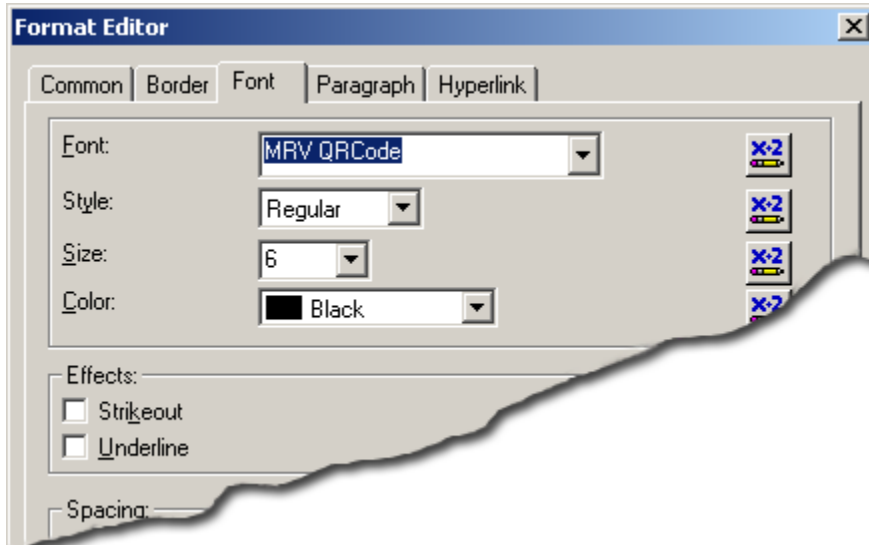


4. In the Formula Editor, choose Functions → Additional Functions → Visual Basic UFLs (u2lcom.dll). You should see QRCodeEncodeSet appears under. If not, the DLL was not installed properly.
5. In the edit box below, copy and paste entire script in Table 5.1, “List of Crystal Reports UFL Functions ( cruf1QRcode5. d11)”. Make necessary changes. For example, you are likely to change the value of DataToEncode variable to one of your database field. Version, Error Correction Level can be changed in the code accordingly.
6. Choose Save → Close to close Formula Workshop.
7. From the Field Explorer, drag Morovia\_QRCode Formula Field to the report.
8. Choose File → Print Preview. You should see a series of meaningless lower case letters and digits in the box. This is normal as the barcode string is completely difference from the data encoded in case of QRCode.

barcode string

```
F8BBB8F05F2741D4A0F8BBB8F
E2AAA2E060E0B075F0E2AAA2E
DFF9C7A9CF93AACB4A0F0A0F3
70FA62A87D59E80B39A48D82B
B2AA2AB0E03CCC8CF8A8FE8AA
F0EEE0F0F585B8BD9DC9D2847
8888888080000880080888888
```

9. Switch back to design mode. Right click on the field and choose Format Field. Click on the Font tab, and choose the MRV QRCode font. Set the font size to 6 points or to the size appropriate for your environment.



10. Drag the cursor to make the formula field big enough to contain the entire barcode. You may need to adjust both horizontally and vertically. Be sure to leave some spaces for quiet zone requirement.
11. Run the Print Preview again. The barcode should come up. Print a page and scan the barcode to make sure that the font size is appropriate and the bounding rectangle is big enough to hold the whole barcode.

---

**Note** If you add barcodes to report using the trial version and subsequently upgrade to the full version, you need to refresh the report once. Otherwise the barcode will continue to be scrambled.

---

## Distributing UFL, Fonts with your report application

Once you finish the report design, you can distribute your report application with Crystal run time files, barcode fonts and the UFL library.

---

**Note** Developer License is required to distribute font files and encoder UFL DLL outside your organization.

---

Two run time files need to be included in your distribution package:

1. **Morovia QRCode Font.** The font files are located `C:\windows\fonts\mrvqrcode.ttf`, and should be copied to the target computer's fonts folder.
2. **QRCode Encoder UFL.** This DLL is installed at `C:\Program Files\Morovia QRCode Fonts & Encoder 5` by default. In the target computer, it can be placed anywhere. However, this is a COM DLL and requires registration. If you are deploying to a 64-bit Crystal Reports runtime, you need the 64-bit version of the DLL.

# Chapter 6. Adding QR Codes in Microsoft Office

To insert a QR Code symbol into Office document, you can create the barcode in QRCode Encoder GUI and transfer it through clipboard. In Word, you can also use the Word AddIn included in the software. Either way, you can't change the barcode because the barcode image does not have the information of the data encoded. If you need to create QR code images on the fly in a Microsoft Office program, you can select either ActiveX control or font-based. It depends on the exact situation - in some cases ActiveX control based is better and may be the only solution supported; and in other cases font-based solution is better. In this chapter we'll overview several typical scenarios.

## VBA Module

Microsoft Office programs, including Word, Excel and Access, can not call DLL functions directly. Instead, you need to import the VBA module included into your document first. This module adds function to create barcode string of a QR code.

**Table 6.1. List of VBA Functions (Morovia.QRCodeFontDLL5.bas)**

| Function Prototype   | Font to Use   | Comment   |
|--|---------------|---|
| QRCodeEncode(DataToEncode As String, _<br>Optional versionID As Long=0, _<br>Optional ecLevel As Long=0) As String | MRV<br>QRCode | Returns the barcode string that<br>with DataToEncode encoded. |

Note that the version and error correction level parameters are optional. So you can call the function as below:  
`barcodestr = QRCodeEncode("this is data")`

### To Import VBA Module

To import the QR code VBA module, perform the following steps:

1. In Office 2003, Select Tools → Macro → Visual Basic Editor to launch Visual Basic Editor. In Office 2007 and above, first enable developer tab if you can't see it. Then click on Developer tab followed by Visual Basic.
2. Choose File → Import File. Navigate to QRCode Fonts & Encoder installation folder, by default `c:\program files (x86)\Morovia QRCode Fonts & Encoder 51`, and select `Morovia.QRCodeFontDLL5.bas`.
3. Close Visual Basic Editor and save the file.
4. After the VBA is imported successfully, formula `QRCodeEncode` is available to use.

## Word Mail Merge

QR Code can be crated in a mail merge word document from a data source. Note that in order for the barcode properly created, you should have the access to the data source to add an extra field that holds the barcode string. You can't just pull a field and format it with the font. This tutorial assumes an Excel data source.

1. Import the VBA module into the Excel spreadsheet, as described in the section called “To Import VBA Module”.
2. Given each column a title on row 1 if you have not done so. With this you can refer a column with its title, instead of cell reference such as \$A\$1.
3. At the right side choose a blank column and at row 1 give it title `barcodestring`.
4. Add a formula on row 2 of the `barcodestring` column that calls the function of the formula. For example, to create QR code on column A2, enter `QRCodeEncode(A2, 0, 0)`.
5. After pressing enter, you should see the cell filled with a string with hexadecimal characters. Highlight this cell and choose Edit → Copy (**Ctrl+C**).
6. Select the entire range of cells in `barcodestring` column to paste this formula and choose Edit → Paste (**Ctrl+V**).
7. Follow the procedures in Word to add this spreadsheet as the data source and pull the merge fields into Word document, including `barcodestring` field.
8. Highlight «`barcodestring`» merge field and select MRV QRCode font.
9. Click on the merge button to display merged document. The QR code should appear in the merged document. Adjust font size to adjust the barcode size.

---

**Tip** If you saw blank lines inside barcode, highlight the field and select paragraph. Set spacing After and Before to 0, and Line spacing to Single/1.0.

---

A working example is provided in the software. See `Word-MailMerge.doc` and `MailMerge-Data-Source.xls` for details.

## Using QRCode ActiveX in Word and Excel

It is straightforward to use QRCode ActiveX control in Word, and Excel. You can bind the data encoded to a cell in Excel, or change the QRcode by editing properties.

### **Inserting QRCode ActiveX into Word/Excel**

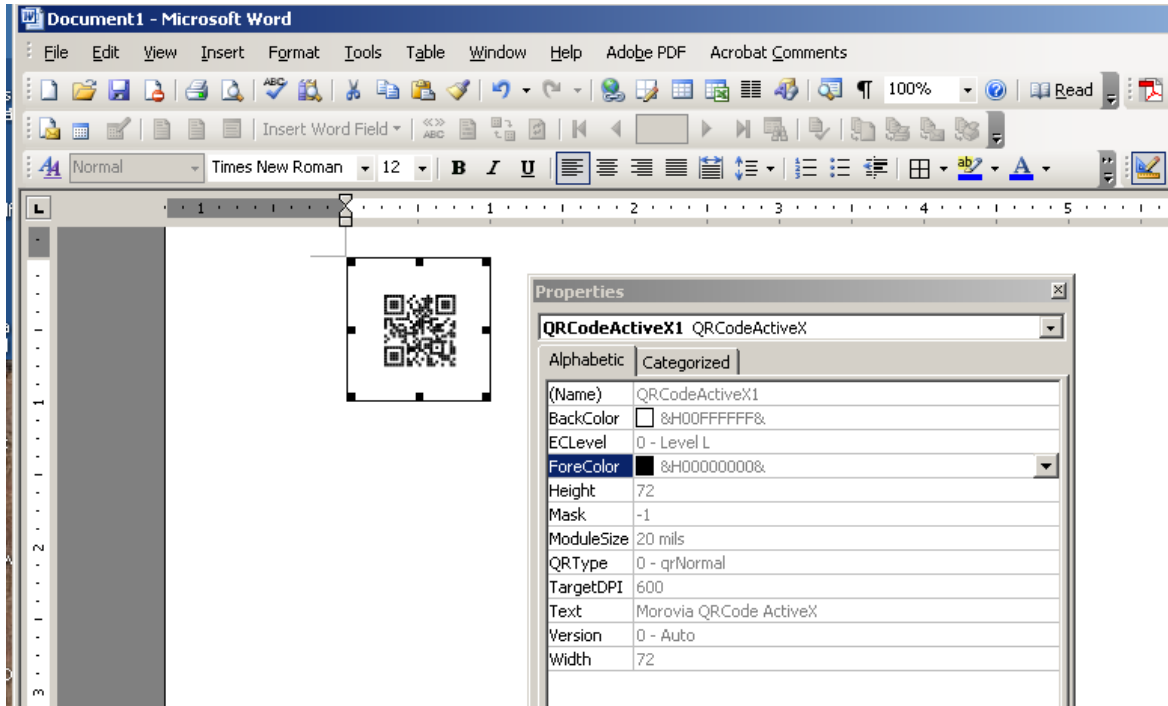
You insert QRCode ActiveX control into Word/Excel documents through Insert Control command. First, make sure the control toolbox toolbar appears somewhere over the window by check the option from View | Toolbars.

At the right bottom of the Control Toolbox, click on the More Controls icon. From the pop-up menu, scroll all the way down and select Morovia QRCodeActiveX. The cursor turns into a cross-hair icon. Click on the place you'd like to place the object and drag the mouse down to draw a rectangle. Release the mouse. The barcode control now appears on the document. The barcode object fills the rectangle you just drew.

### **Editing QRCode ActiveX Object**

The QRCode ActiveX object does not offer menus or toolbars for you to do the interactive editing. Instead, you can edit the barcode object by modifying the properties. The property window is accessible by right clicking on the barcode object and select Properties.

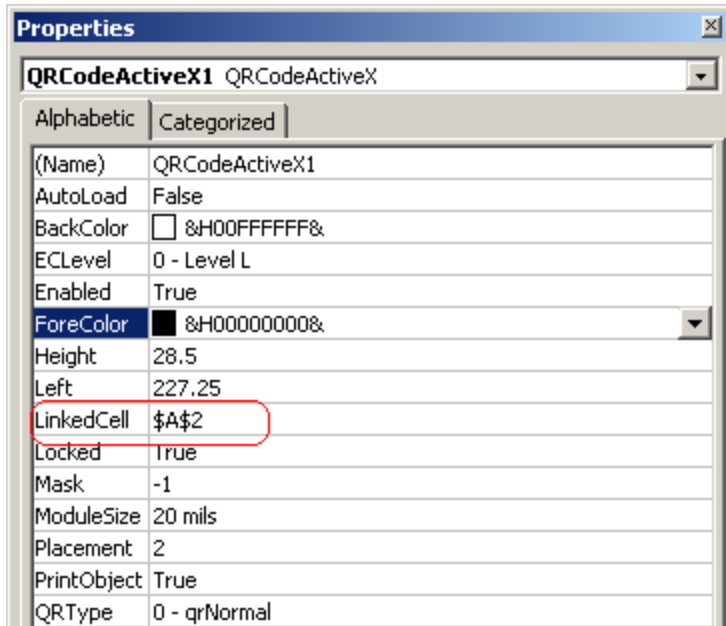
**Figure 6.1. Editing QRCode ActiveX properties in Office programs**



If you are familiar with Visual Basic, you know that it is the same property sheet appearing in the Visual Basic when placing the Barcode ActiveX into a form.

### ***Data Binding in Excel***

When creating the object in Excel, Excel adds the property `LinkedCell` in the property list. This additional property links the `Message` property to a cell. For example, to make the barcode data bound to cell G4, enter G4 in the `LinkedCell` property entry.

**Figure 6.2. Using LinkedCell Property to Hookup with Excel Data**

## Adding QR Code in Access Reports

You can use either ActiveX control or QRcode font in Access reports. You can find example files `Access Example (DLL).mdb` and `Access Example (ActiveX).mdb` under the installation directory, which demonstrates the two approaches.

## Adding QR Code to InfoPath

A major improvement in 5.1 release is InfoPath support. Now you can add QR code easily to an Infopath form and bind the control to a field.

### Adding QRCode Control to Custom Controls

Follow the steps below to add QRCode Control to Custom Controls. You only need to perform this step once.

1. Make sure that you are in Design Mode. You can activate the *Design Mode* by clicking on the triangle symbol

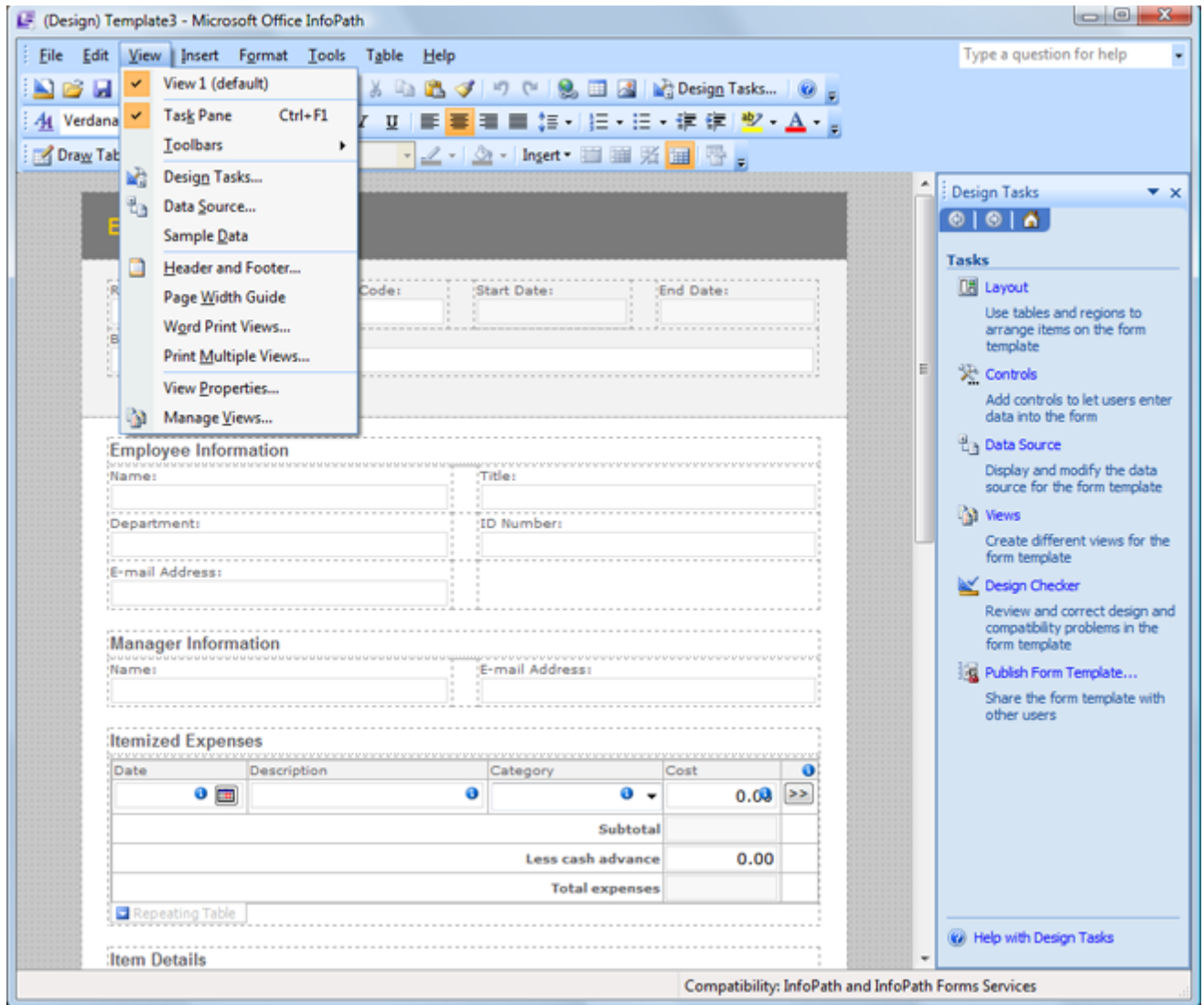


in the Standard toolbar, or select `Tools → Design this Form... (Atl-T+M)`

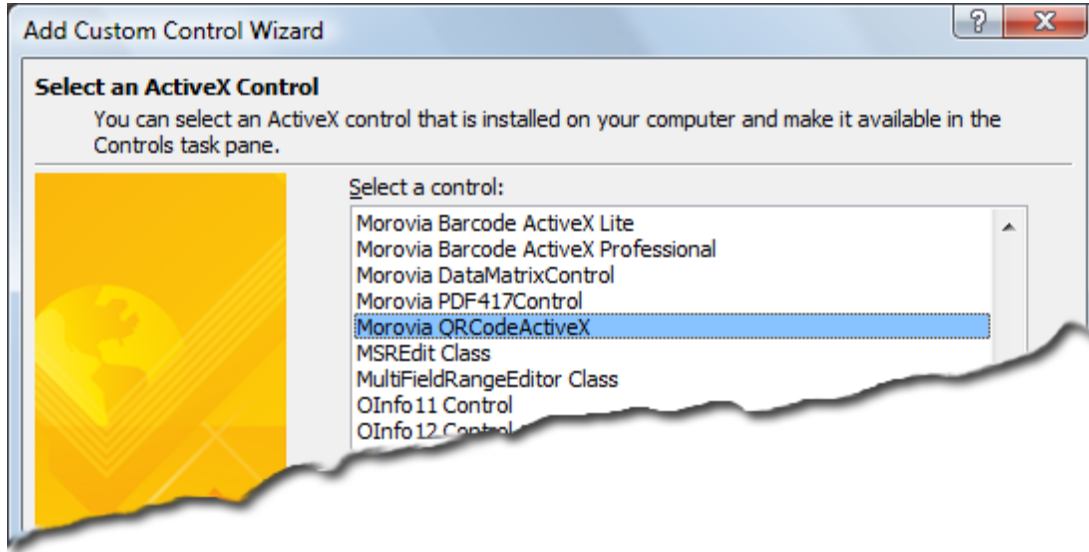
2. Open the Task Pane if it is not visible by `View → Task Pane (Ctrl+F1)`.



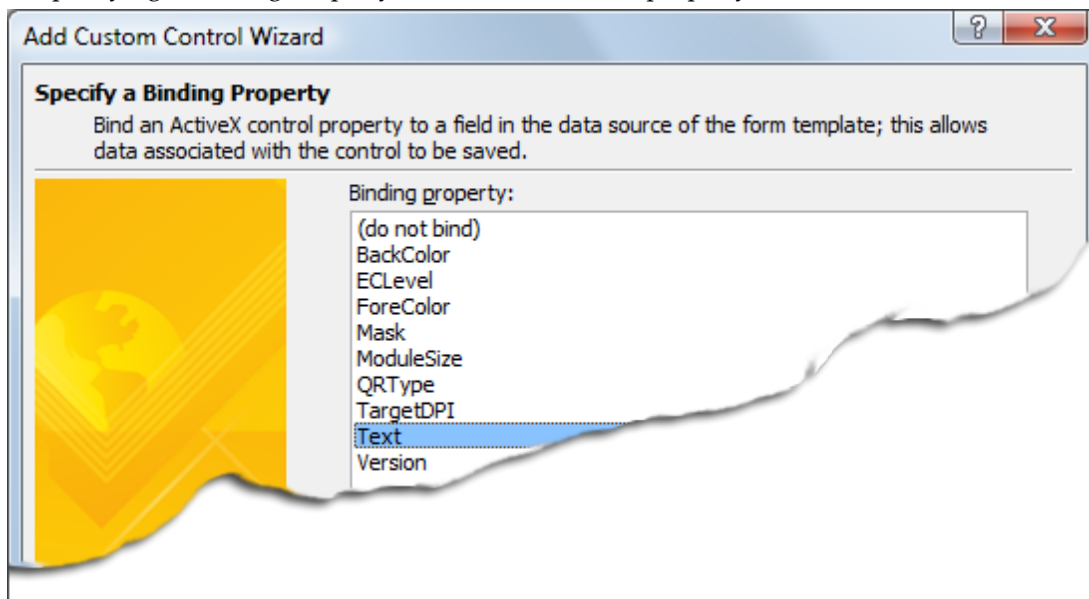
Figure 6.3. Adding QRCode control to Custom Controls in Microsoft InfoPath



3. In the Task Pane, click on Controls...
4. At the bottom of the Controls task pane, click Add or Remove Custom Controls.
5. In the Add or Remove Custom Controls dialog box, click Add. From the wizard, select ActiveX Control, followed by Morovia QRCodeActiveX.



6. On the next page, leave Don't include a .cab file selected. Click Next.
7. In Specifying a Binding Property list, click on the Text property.

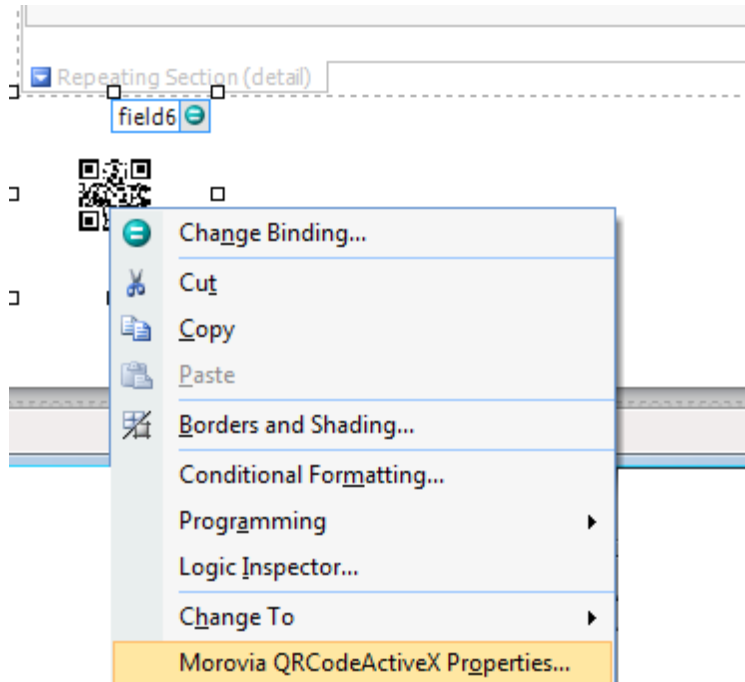


Note: Forms templates designed to be opened in a browser can't contain ActiveX controls. If you saw a warning message that ActiveX control can't be used in this template, uncheck Design a form template that can be opened in a browser or InfoPath option. This option is accessible through Tools → Form Options → Compatibility.

### ***Inserting Barcode into Form Template***

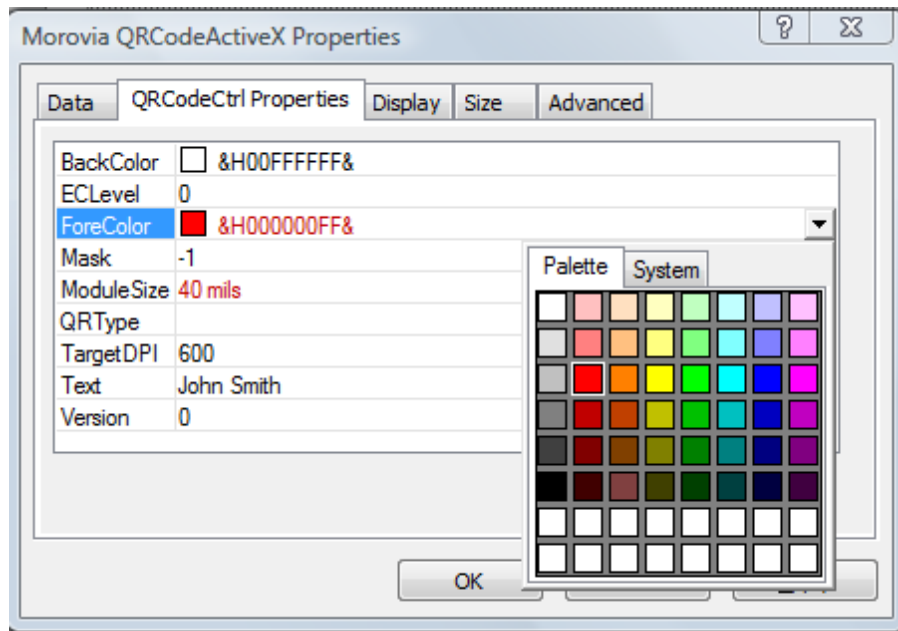
After you added QRCode control to the Custom Controls in Microsoft InfoPath, you can add a barcode to form template.

1. Adjust or resize the form to create space for the control. Drag the control and drop to the place. A small barcode will appear at the place.



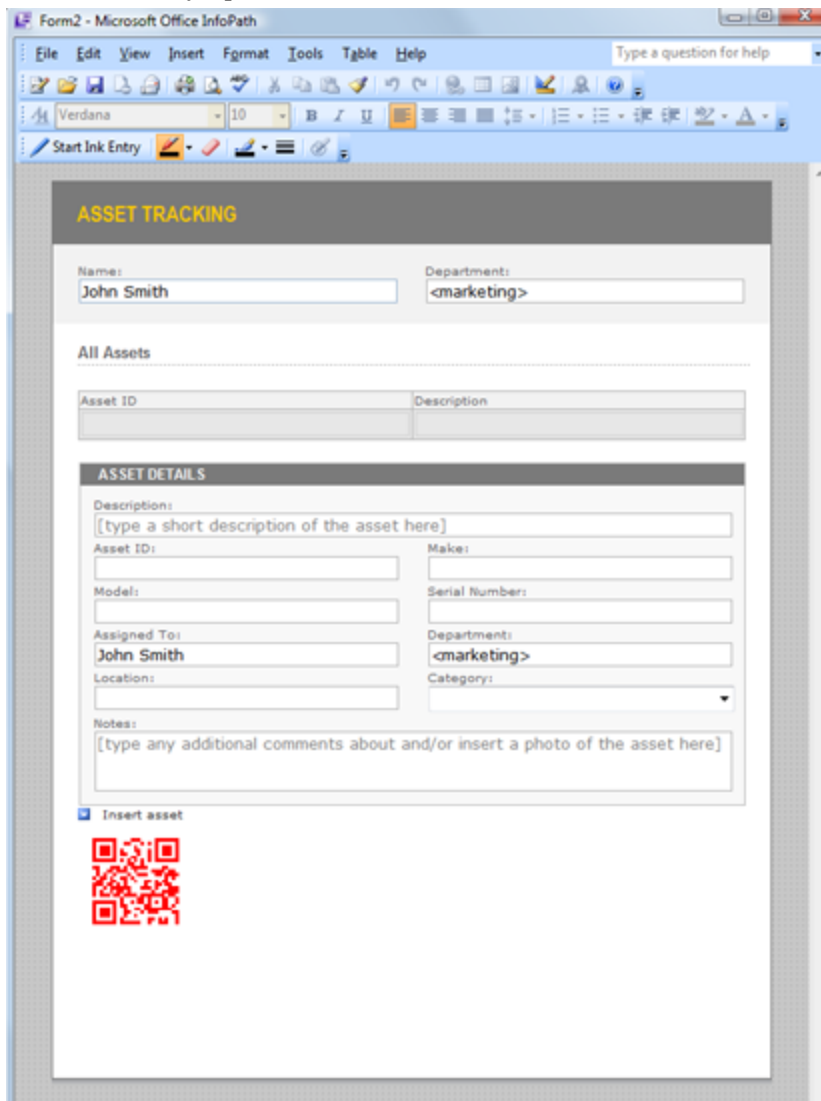
- Right click on the control. From the menu you can change the binding to another field on the form using Changing binding... To change the appearance of the barcode, select Morovia QRCodeActive Properties... Click on QRCodeCtrl Properties tab.

There are a number of properties to change. Here we click on ForeColor edit box and change the foreground color to red. Click on ModuleSize edit box and change it to 40 mils. Properties changed have their values displayed in crimson color. Click on the Apply to apply the changes. The barcode on the design form changed accordingly.





3. After saving and/or publishing the modified form, the barcode should appear. The barcode field will be automatically updated after that field is edited and the cursor is moved to another field.



# Chapter 7. Developing Application with QRCode Fonts & Encoder

Morovia QRCode Fontware & Writer SDK provides two distinct approaches when integrating with your application - a Windows encoder DLL and an image-based ActiveX. Depending on your programming environment, you may find one way easier than the other.

## Common Issues

### Color Types

The software supports RGB color space, with each component expressed in a byte. Windows programmers often deal with two types of color values: COLORREF and OLE\_COLOR. The first type uses a 32-bit integer with the leading byte always empty. The second one use the most significant bit to indicate whether it is a system color and conversion is required. The two color types use the lower three bytes for B, G and R components respectively - note that the Red component is in the last byte.

Other platforms typically place the component values in the order of Red, Green and Blue. We adopts this format in our DLL interface. Therefore, the value of 0xff0000 indicates red color in our DLL, but blue to Windows if no translation is performed.

In QRCodeActiveX control, the color is represented as OLE\_COLOR format, therefore no translation is required.

### Version

In DLL and ActiveX Control, you can specify the size desired by setting a value to Version. If the value is 0, the program chooses the value that produces the smallest barcode possible. If the value specified is too small to encode all the data, the program increases the value automatically.

### Error Codes

Error may occur during encoding process - for example, the data may exceed the maximum capacity allowed, and malformed structural append block or Kanji block may exist in the input. QRCode ActiveX control and encoder DLL return error code in COM-compliant way - with this code consisting of any information such as facility code.

**Table 7.1. Error Codes**

| Value | HRESULT    | Description                      |
|-------|------------|----------------------------------|
| 5     | 0x83290005 | Input is too long                |
| 6     | 0x83290006 | Invalid characters in input data |
| 9     | 0x83290009 | Encoding problem. Try run again  |
| 10    | 0x8329000a | File access denied               |
| 11    | 0x8329000b | Out of memory                    |
| 12    | 0x8329000c | Output buffer size is too small  |
| 30    | 0x832901e  | Invalid length string            |

| Value | HRESULT    | Description                                 |
|-------|------------|---|
| 50    | 0x83290032 | Missing field in Structural Append block    |
| 52    | 0x83290034 | Invalid Sequence in Structural Append block |
| 53    | 0x83290035 | Invalid TotalNum in Structural Append block |
| 80    | 0x83290050 | Invalid Kanji block                         |
| 81    | 0x83290051 | Invalid FNC1 second position block          |

DLL client can call exported function `QRCodeGetErrorMessage` to retrieve a text description on the error message.

COM client can retrieve `IErrorInfo` interface to the Error object, following the standard COM error protocol.

## Windows Native DLL Interface

the file name of this dll is `MoroviaQRCodeFontEncoder.dll`, installed by default in the system directory `c:\windows\system32`. The Encoder GUI and Word AddIn calls this dll at the background for encoding.

Using DLL is convenient in many occasions because you do not need to register it. Instead it, just place the dll under the same directory as you exe.

## ActiveX Control Interface (COM)

Writing a client application that creates and manipulates COM objects is quite straightforward and supported in many programming environments. We have included coding examples in C++, C# and Visual Basic 6. Investigate the examples we provided if you decide to use this approach.

# Chapter 8. Encoder DLL API Reference

Creating QR Code barcodes using DLL APIs involves three steps. The first step is to call `QRCodeEncode` or `QRCodeEncode2` with data to encode and size option. If this step succeeded, a pointer is returned from which you can query the attributes of the barcode created, “paint” the barcode into an image file, or obtain the barcode string. Finally, you should call `DestroyQRCodeEncodeResult` to release the memory resource used by the encoder result object.

**Table 8.1. List of Enumerations**

| Enumeration                | Comment                                 |
|----------------------------|---|
| <code>ImageTypeEnum</code> | Image formats supported by the encoder. |

**Table 8.2. List of Functions**

| Function                                   | Comment   |
|--|---|
| <code>QRCodeEncode</code>                  | Encodes data and returns a pointer that points to encoder result object.  |
| <code>QRCodeEncode2</code>                 | Abbreviated version of <code>QRCodeEncode</code> .  |
| <code>QRCodeEncode2W</code>                | The function <code>QRCodeEncode2W</code> is used to encode unicode strings into QR Code in a way compliant with ZXing reader. |
| <code>MicroQRCodeEncode</code>             | Encodes data in Micro QR Code and returns a pointer that points to encoder result object.                                     |
| <code>MicroQRCodeEncode2</code>            | Abbreviated version of <code>MicroQRCodeEncode</code> .   |
| <code>HIBC_QRCodeEncode</code>             | Encodes data and returns a pointer that points to encoder result object.  |
| <code>HIBC_QRCodeEncode2</code>            | Abbreviated version of <code>QRCodeEncode</code> .  |
| <code>QRCodeResultGetVersion</code>        | Retrieves the size ID of the QRCode barcode created.  |
| <code>QRCodeResultGetBarcodeString</code>  | Retrieves the barcode string that becomes a QRCode symbol after being formatted with a QRCode font.                           |
| <code>DestroyQRCodeEncodeResult</code>     | Releases all resources allocated to the encoder result object.  |
| <code>PaintQRCodeImageRaster</code>        | Writes the QRCode barcode in raster image format specified to a disk file.  |
| <code>PaintQRCodeImageVector</code>        | Writes the QRCode barcode in vector image format specified to a disk file.  |
| <code>PaintQRCodeImageRaster2</code>       | Writes the QRCode barcode in raster image format specified to IStream object.   |
| <code>PaintQRCodeImageVector2</code>       | Writes the QRCode barcode in vector image format specified to an IStream object.  |
| <code>PaintQRCodeImageEMF</code>           | Creates a Windows Enhanced Meta File object and returns the handle.   |
| <code>QRCodeGetErrorMessage</code>         | Retrieves a string that describes the error.  |
| <code>QRCodeResultGetBarcodeString2</code> | Returns barcode string without requiring user to preallocate the buffer.  |
| <code>PaintQRCodeImageClipboard</code>     | Creates a Windows Enhanced Meta File object and place it into the clipboard.  |

| Function        | Comment  |
|-----------------|--|
| QRCodeGetParity | Calculates the parity value based on the whole data encoded and split into multiple symbols. |

## ImageTypeEnum

Image formats supported by the encoder.

```
enum ImageTypeEnum {
    IMAGE_PNG== 0,
    IMAGE_SVG== 1,
    IMAGE_EMF== 2,
    IMAGE_EPS== 3,
    IMAGE_BMP== 4
};
```

### Remarks

The ImageTypeEnum enum has the following values:

**Table 8.3. ImageTypeEnum Enumeration**

| Constant  | Value | Comment                         |
|-----------|-------|---------------------------------|
| IMAGE_PNG | = 0   | PNG (Portable Network Graphics) |
| IMAGE_SVG | = 1   | SVG (Scalable Vector Graphics)  |
| IMAGE_EMF | = 2   | EMF (Windows Enhanced MetaFile) |
| IMAGE_EPS | = 3   | EPS (Encapsulated PostScript)   |
| IMAGE_BMP | = 4   | BMP (Windows Bitmap)            |

PNG and BMP are raster formats, which store color information of individual pixels. SVG, EMF and EPS are vector graphics formats and contains drawing commands instead. If you are using raster graphic format, one pixel should be mapped to one or integral times pixels on the printer. If you are using vector graphics format, the drawing units should map to integral times pixels on the printer.

## QRCodeEncode

The **QRCodeEncode** function encodes data and returns a pointer that points to encoder result object.

```
int __stdcall QRCodeEncode(
    const char * dataToEncode,
    int versionRequested,
    int ecLevel,
    void ** ppResult
);
```

### Parameters

#### dataToEncode

[in] Pointer to a null-terminated string containing the data to be encoded. You can use tilde codes to encode control characters such as NUL, as well as advanced features such as ECI and structural append. For more information, see the section called “Input Format”.



**versionRequested**

[in] An integer value that corresponds to the size ID of the QRCode barcode. QR Code has a total of 40 different sizes, ranging between 1 and 40.

<linebreak></linebreak>

If 0 is specified, the encoder will select the minimum size that can have the data encoded. Note: after version 5.2 this parameter is updated to include modifier byte, which can be used to fine tune the encoder behavior. See Appendix C, *Version ID Parameter (updated in version 5.2)*

**ecLevel**

[in] An integer that sets the error correction level in the barcode. Valid values are 0, 1, 2 and 3, which correspond to Level L, M, Q and H respectively.

**ppResult**

A pointer to a pointer that points to the encode result created. Use this pointer to discover the actual attributes of the symbol created, or create image files.

**Return Values**

If the function succeeded, the return value is 0. If the function failed, it returns the error code. You can call function `QRCodeGetErrorMessage` to obtain the description of the error.

**Remarks**

This function encodes the data according to the parameter specified, and returns a pointer from which you can retrieve the encoding result.

## QRCodeEncode2

The function `QRCodeEncode2` is an abbreviated version of `QRCodeEncode`.

```
void *__stdcall QRCodeEncode2(
    const char * dataToEncode,
    int versionRequested,
    int ecLevel
);
```

**Parameters****dataToEncode**

[in] Pointer to a null-terminated string containing the data to be encoded. You can use tilde codes to encode control characters such as NUL, as well as advanced features such as ECI. For more information, see the section called “Input Format”.

**versionRequested**

[in] An integer value that corresponds to the size ID of the QRCode barcode. QR Code has a total of 40 different sizes, ranging between 1 and 40.

<linebreak></linebreak>

If 0 is specified, the encoder will select the minimum size that can have the data encoded.

**ecLevel**

[in] An integer that sets the error correction level in the barcode. Valid values are 0, 1, 2 and 3, which correspond to Level L, M, Q and H respectively.

**ppResult**

A pointer to a pointer that points to the encode result created. Use this pointer to discover the actual attributes of the symbol created, or create image files.

### Return Values

If the function failed, the return value is 0 (NULL). If it succeeds, it returns a pointer that points to the result object. You should release the encoder result object by calling function `DestroyQRCodeEncodeResult`.

### Remarks

In this function, a pointer is returned instead of status code. A non-null return value indicates that no error occurred. The pointer associated resource must be released by calling `DestroyQRCodeEncodeResult`.

## QRCodeEncode2W

The function `QRCodeEncode2W` is used to encode unicode strings into QR Code in a way compliant with ZXing reader.

```
void *__stdcall QRCodeEncode2W(
    const wchar_t * dataToEncode,
    int versionRequested,
    int ecLevel
);
```

### Parameters

#### `dataToEncode`

[in] Pointer to a null-terminated UTF16 string containing the data to be encoded. You can use tilde codes to encode control characters such as NUL, as well as advanced features such as ECI. For more information, see the section called “Input Format”.

#### `versionRequested`

[in] An integer value that corresponds to the size ID of the QRCode barcode. QR Code has a total of 40 different sizes, ranging between 1 and 40.

<linebreak></linebreak>

If 0 is specified, the encoder will select the minimum size that can have the data encoded.

#### `ecLevel`

[in] An integer that sets the error correction level in the barcode. Valid values are 0, 1, 2 and 3, which correspond to Level L, M, Q and H respectively.

#### `ppResult`

A pointer to a pointer that points to the encode result created. Use this pointer to discover the actual attributes of the symbol created, or create image files.

### Return Values

If the function failed, the return value is 0 (NULL). If it succeeds, it returns a pointer that points to the result object. You should release the encoder result object by calling function `DestroyQRCodeEncodeResult`.

### Remarks

Read the section called “Unicode String Support” for more information. QR Code does not natively support unicode. Use of this feature require special support of a reader.

## MicroQRCodeEncode

The **MicroQRCodeEncode** function encodes data in Micro QR Code and returns a pointer that points to encoder result object.

```
int __stdcall MicroQRCodeEncode(
    const char * dataToEncode,
    int versionRequested,
    int ecLevel,
    void ** ppResult
);
```

### Parameters

#### dataToEncode

[in] Pointer to a null-terminated string containing the data to be encoded. You can use tilde codes to encode control characters such as NUL. For more information, see the section called “Input Format”.

#### versionRequested

[in] An integer value that corresponds to the size ID of the QRCode barcode. Micro QR Code has a total of 4 sizes: 1, 2, 3 and 4. If 0 is specified, the encoder selects the minimum version that can accommodate the data. If value greater than 4 is specified, the encoder uses 4.

#### ecLevel

[in] An integer that sets the error correction level in the barcode. Valid values are 0, 1, 2 and 3, which correspond to Level E, L, M and Q, respectively. Not all version/eclevel combinations are possible. If ecLevel is not possible, a lower level is used. If the error correction level is not achievable in the version requested, the symbol version is bumped up.

#### ppResult

A pointer to a pointer that points to the encode result created. Use this pointer to discover the actual attributes of the symbol created, or create image files.

### Return Values

If the function succeeded, the return value is 0. If the function failed, it returns the error code. You can call function **QRCodeGetErrorMessage** to obtain the description of the error.

### Remarks

This function encodes the data according to the parameter specified, and returns a pointer from which you can retrieve the encoding result.

## MicroQRCodeEncode2

The function **MicroQRCodeEncode2** is an abbreviated version of **MicroQRCodeEncode**.

```
void * __stdcall MicroQRCodeEncode2(
    const char * dataToEncode,
    int versionRequested,
    int ecLevel
);
```

### Parameters

#### dataToEncode

[in] Pointer to a null-terminated string containing the data to be encoded. You can use tilde codes to encode control characters such as NUL. For more information, see the section called “Input Format”.

**versionRequested**

[in] An integer value that corresponds to the size ID of the QRCode barcode. Micro QR Code has a total of 4 sizes: 1, 2, 3 and 4. If 0 is specified, the encoder selects the minimum version that can accommodate the data. If value greater than 4 is specified, the encoder uses 4.

**ecLevel**

[in] An integer that sets the error correction level in the barcode. Valid values are 0, 1, 2 and 3, which correspond to Level E, L, M and Q, respectively. Not all version/eclevel combinations are possible. If ecLevel is not possible, a lower level is used. If the error correction level is not achievable in the version requested, the symbol version is bumped up.

**ppResult**

A pointer to a pointer that points to the encode result created. Use this pointer to discover the actual attributes of the symbol created, or create image files.

**Return Values**

If the function failed, the return value is 0 (NULL). If it succeeds, it returns a pointer that points to the result object. You should release the encoder result object by calling function DestroyQRCodeEncodeResult.

**Remarks**

In this function, a pointer is returned instead of status code. A non-null return value indicates that no error occurred. The pointer associated resource must be released by calling DestroyQRCodeEncodeResult.

## HIBC\_QRCodeEncode

The **HIBC\_QRCodeEncode** function encodes data and returns a pointer that points to encoder result object.

```
int __stdcall HIBC_QRCodeEncode(
    const char * dataToEncode,
    int versionRequested,
    int ecLevel,
    void ** ppResult
);
```

**Parameters****dataToEncode**

[in] Pointer to a null-terminated string containing the data to be encoded. You can use tilde codes to encode control characters such as NUL, as well as advanced features such as ECI and structural append. For more information, see the section called "Input Format".

**versionRequested**

[in] An integer value that corresponds to the size ID of the QRCode barcode. QR Code has a total of 40 different sizes, ranging between 1 and 40.

<linebreak></linebreak>

If 0 is specified, the encoder will select the minimum size that can have the data encoded.

**ecLevel**

[in] An integer that sets the error correction level in the barcode. Valid values are 0, 1, 2 and 3, which correspond to Level L, M, Q and H respectively.

**ppResult**

A pointer to a pointer that points to the encode result created. Use this pointer to discover the actual attributes of the symbol created, or create image files.

## Return Values

If the function succeeded, the return value is 0. If the function failed, it returns the error code. You can call function `QRCodeGetErrorMessage` to obtain the description of the error.

## Remarks

This function encodes the data according to the parameter specified, and returns a pointer from which you can retrieve the encoding result.

# HIBC\_QRCodeEncode2

The function `HIBC_QRCodeEncode2` is an abbreviated version of `QRCodeEncode`.

```
void *__stdcall HIBC_QRCodeEncode2(
    const char * dataToEncode,
    int versionRequested,
    int ecLevel
);
```

## Parameters

### dataToEncode

[in] Pointer to a null-terminated string containing the data to be encoded. You can use tilde codes to encode control characters such as NUL, as well as advanced features such as ECI. For more information, see the section called “Input Format”.

### versionRequested

[in] An integer value that corresponds to the size ID of the QRCode barcode. QR Code has a total of 40 different sizes, ranging between 1 and 40.

<linebreak></linebreak>

If 0 is specified, the encoder will select the minimum size that can have the data encoded.

### ecLevel

[in] An integer that sets the error correction level in the barcode. Valid values are 0, 1, 2 and 3, which correspond to Level L, M, Q and H respectively.

### ppResult

A pointer to a pointer that points to the encode result created. Use this pointer to discover the actual attributes of the symbol created, or create image files.

## Return Values

If the function failed, the return value is 0 (NULL). If it succeeds, it returns a pointer that points to the result object. You should release the encoder result object by calling function `DestroyQRCodeEncodeResult`.

## Remarks

In this function, a pointer is returned instead of status code. A non-null return value indicates that no error occurred. The pointer associated resource must be released by calling `DestroyQRCodeEncodeResult`.

# QRCodeResultGetVersion

The `QRCodeResultGetSizeID` function retrieves the size ID of the QRCode barcode created.

```
int __stdcall QRCodeResultGetVersion(
```

```

    void * pEncodeResult
);

```

### Parameters

#### **pEncodeResult**

[in] The pointer that points to the encoding result object, returned from `QRCodeEncode` or `QRCodeEncode2` functions.

### Return Values

The actual version of the barcode.

## QRCodeResultGetBarcodeString

The `QRCodeResultGetBarcodeString` function retrieves the barcode string that becomes a QRCode symbol after being formatted with a QRCode font.

```

int __stdcall QRCodeResultGetBarcodeString(
    void * pEncodeResult,
    char * buffer,
    unsigned int * maxSize,
    const char * eol
);

```

### Parameters

#### **pEncodeResult**

[in] The pointer that points to the encoding result object, returned from `QRCodeEncode` or `QRCodeEncode2` functions.

#### **buffer**

[in] The pointer that points to a byte array that receives the string.

#### **maxSize**

[in,out] Pointer to a variable that specifies the size of the buffer pointed to by the `buffer` parameter, in bytes. When the function returns, this variable contains the size of the data copied to buffer.

#### **eol**

[in] Pointer to a NUL terminated string that will be appended to each row in the barcode string.

### Return Values

If the function succeeded, it returns the length of the string (excluding terminating NUL character). If the function fails due to insufficient storage, it returns 0.

### Remarks

When creating barcodes using font-based solution, you first call encoder function `QRCodeEncode` or `QRCodeEncode2` to obtain a pointer that points to the result object. Then `QRCodeResultGetBarcodeString` is called to obtain a string that becomes QRCode symbol after the font is applied. If memory is not an issue, allocate a large buffer with 8096 bytes to hold the largest QRCode barcode. This size is sufficient for the largest symbol with carriage return and line feed as line ending.

## DestroyQRCodeEncodeResult

The **DestroyQRCodeEncodeResult** function releases all resources allocated to the encoder result object.

```
void __stdcall DestroyQRCodeEncodeResult(
    void * pResult
);
```

### Parameters

#### pResult

[in] Pointer to the encoder result object.

### Remarks

After the object is destroyed, the specific pointer pResult is no longer valid.

## PaintQRCodeImageRaster

The **PaintQRCodeImageRaster** function writes the QRCode barcode in raster image format specified to a disk file.

```
int __stdcall PaintQRCodeImageRaster(
    void * pEncodeResult,
    const wchar_t * wszFilename,
    int pixelsPerModule,
    int forecolor,
    int backcolor,
    int imageType
);
```

### Parameters

#### pEncodeResult

[in] Pointer that points to the encoder result object.

#### wszFilename

[in] Pointer to the file name for the image file to be created.

#### pixelsPerModule

[in] Number pixels per module with.

#### forecolor

[in] Value of the foreground color, in RGB color space.

#### backcolor

[in] Value of the background color, in RGB color space.

#### imageType

[in] An integer that indicates the image file format. The current version supports two formats: PNG(0) and BMP(4).

### Return Values

If the function failed, it returns the error code. You can call function **QRCodeGetErrorMessage** to obtain the description of the error.

### Remarks

Note: the order of the component bytes are R, G and B, which is opposite of **COLORREF** type on Windows.

## PaintQRCodeImageVector

The **PaintQRCodeImageVector** function writes the QRCode barcode in vector image format specified to a disk file.

```
int __stdcall PaintQRCodeImageVector(
    void * pEncodeResult,
    const wchar_t * wszFilename,
    int module_width_hm,
    int target_dpi,
    int forecolor,
    int backcolor,
    int imageType
);
```

### Parameters

#### pEncodeResult

[in] Pointer that points to the encoder result object.

#### wszFilename

[in] Pointer to the file name for the image file to be created.

#### module\_width\_hm

[in] Module width (X dimension), in the unit of high metric. 1 unit high metric = 1/1000 cm.

#### target\_dpi

[in] The resolution of the target printer.

#### forecolor

[in] Value of the foreground color, in RGB color space.

#### backcolor

[in] Value of the background color, in RGB color space.

#### imageType

[in] An integer that indicates the image file format. The current version supports three formats: SVG(1), EMF(2) and EPS(3).

### Return Values

If the function failed, it returns the error code. You can call function **QRCodeGetErrorMessage** to obtain the description of the error.

### Remarks

Note: the order of the component bytes are R, G and B, which is opposite of **COLORREF** type on Windows.

## PaintQRCodeImageRaster2

The **PaintQRCodeImageRaster** function writes the QRCode barcode in raster image format specified to IStream object.

```
int __stdcall PaintQRCodeImageRaster2(
    void * pEncodeResult,
    IStream * ostream,
    int pixelsPerModule,
    int forecolor,
    int backcolor,
    int imageType
);
```



```
);
```

### Parameters

#### **pEncodeResult**

[in] Pointer that points to the encoder result object.

#### **ostream**

[in] Pointer to an IStream object.

#### **pixelsPerModule**

[in] Number pixels per module with.

#### **forecolor**

[in] Value of the foreground color, in RGB color space.

#### **backcolor**

[in] Value of the background color, in RGB color space.

#### **imageType**

[in] An integer that indicates the image file format. The current version supports two formats: PNG(0) and BMP(4).

### Return Values

If the function failed, it returns the error code. You can call function `QRCodeGetErrorMessage` to obtain the description of the error.

### Remarks

Note: the order of the component bytes are R, G and B, which is opposite of `COLORREF` type on Windows.

## PaintQRCodeImageVector2

The `PaintQRCodeImageVector` function writes the QRCode barcode in vector image format specified to an IStream object.

```
int __stdcall PaintQRCodeImageVector2(
    void * pEncodeResult,
    IStream * ostream,
    int module_width_hm,
    int target_dpi,
    int forecolor,
    int backcolor,
    int imageType
);
```

### Parameters

#### **pEncodeResult**

[in] Pointer that points to the encoder result object.

#### **ostream**

[in] Pointer to an IStream object.

#### **module\_width\_hm**

[in] Module width (X dimension), in the unit of high metric. 1 unit high metric = 1/1000 cm.

#### **target\_dpi**

[in] The resolution of the target printer.

**forecolor**

[in] Value of the foreground color, in RGB color space.

**backcolor**

[in] Value of the background color, in RGB color space.

**imageType**

[in] An integer that indicates the image file format. The current version supports three formats: SVG(1), EMF(2) and EPS(3).

**Return Values**

If the function failed, it returns the error code. You can call function `QRCodeGetErrorMessage` to obtain the description of the error.

**Remarks**

Note: the order of the component bytes are R, G and B, which is opposite of `COLORREF` type on Windows.

## PaintQRCodeImageEMF

The `PaintQRCodeImageClipboard` function Creates a Windows Enhanced Meta File object and returns the handle.

```
HENHMETAFILE __stdcall PaintQRCodeImageEMF(
    void * pEncodeResult,
    int module_len_hm,
    int target_dpi,
    int forecolor,
    int backcolor,
    int imageType
);
```

**Parameters****pEncodeResult**

[in] Pointer that points to the encoder result object.

**module\_width\_hm**

[in] Module width (X dimension), in the unit of high metric. 1 unit high metric = 1/1000 cm.

**target\_dpi**

[in] The resolution of the target printer.

**forecolor**

[in] Value of the foreground color, in RGB color space.

**backcolor**

[in] Value of the background color, in RGB color space.

**Return Values**

If the function failed, it returns the error code. You can call function `QRCodeGetErrorMessage` to obtain the description of the error.

**Remarks**

The caller should destroy this handle after using.

## QRCodeGetErrorMessage

The `QRCodeGetErrorMessage` function retrieves a string that describes the error.

```
const char *__stdcall QRCodeGetErrorMessage(
    int errorno
);
```

### Parameters

#### `errorno`

[in] The error number.

### Return Values

a read only string that describes the error.

### Remarks

The string returned is a read only string. User should not modify the string directly.

## QRCodeResultGetBarcodeString2

The `QRCodeResultGetBarcodeString2` function returns barcode string without requiring user to preallocate the buffer.

```
const char *__stdcall QRCodeResultGetBarcodeString2(
    void * pEncodeResult,
    const char * eol
);
```

### Parameters

#### `pEncodeResult`

[in] The pointer that points to the encoding result object, returned from `QRCodeEncode` or `QRCodeEncode2` functions.

#### `eol`

[in] Pointer to a NUL terminated string that will be appended to each row in the barcode string.

### Return Values

Pointer to a NUL terminated string that represents the barcode (barcode string). If the function failed due to insufficient memory, it returns `NULL`.

### Remarks

The encoder result manages the buffer by itself. Caller should cache the string returned, but not the pointer, as the contents may change after another call of `QRCodeResultBarcodeString2` is made.

## PaintQRCodeImageClipboard

The `PaintQRCodeImageClipboard` function Creates a Windows Enhanced Meta File object and place it into the clipboard.

```
int __stdcall PaintQRCodeImageClipboard(
    void * pEncodeResult,
```

```

    int module_width_hm,
    int target_dpi,
    int forecolor,
    int backcolor
);

```

### Parameters

#### pEncodeResult

[in] Pointer that points to the encoder result object.

#### module\_width\_hm

[in] Module width (X dimension), in the unit of high metric. 1 unit high metric = 1/1000 cm.

#### target\_dpi

[in] The resolution of the target printer.

#### forecolor

[in] Value of the foreground color, in RGB color space.

#### backcolor

[in] Value of the background color, in RGB color space.

### Return Values

If the function failed, it returns the error code. You can call function `QRCodeGetErrorMessage` to obtain the description of the error.

## QRCodeGetParity

The `QRCodeGetParity` function calculates the parity value based on the whole data encoded and split into multiple symbols.

```

int __stdcall QRCodeGetParity(
    const char * dataToEncode
);

```

### Parameters

#### dataToEncode

[in] Pointer to a null-terminated string containing the data to be encoded. You must pass the full data string to this parameter. Subsequent calls should use `~2[seqno][symbol, parity]` plus data that each symbol encodes.

### Return Values

The parity value that groups the symbols. It should be between 0 and 255.

### Remarks

The QR Code structural append features allows data to be split across multiple symbols (< 16). A SA capable scanner reads the symbols one and one, and emits the data after all the symbols are read. Symbols belong to the same cluster share the same parity value, which is calculate based on the full data.

# Chapter 9. QRCode ActiveX Reference

Many programming environments support the use of ActiveX objects. We have tested the Morovia QRCode ActiveX with Visual Basic, Visual C++, Internet Explorer, Microsoft Word, Excel, Access and IIS programs. The object can be used as a control embedded in a VB form or a dialog, or be used at background for image creation and printing purposes. The QR Code ActiveX object can also be inserted into Microsoft Office documents and many other ActiveX-aware programs.

## Specification

**Table 9.1. QRCode ActiveX Specification**

|              |  |
|--------------|--|
| Prog ID      | Morovia.QRCodeActiveX                  |
| ClassID      | {13C6D4E9-81BF-4BC6-B90E-B405CC3C0DC2} |
| Licensed     | no                                     |
| File name    | QRCodeCtrl.dll                         |
| Interface    | IQRCodeActiveX                         |
| Interface ID | {5AF858A3-0D3E-4D8E-B22F-F8191C60C161} |

## Enumerations

**Table 9.2. List of QRCode ActiveX Enumerations**

| Name   | Description        |
|--------|--------------------|
| QRType | QRType Enumeration |

## Properties

**Table 9.3. List of QRCode ActiveX Properties**

| Name          | Description   |
|---------------|---|
| ActualVersion | Returns the actual version in the symbol created.                         |
| BackColor     | Specifies the background color for the control.                           |
| ECLevel       | Specifies the error correction level of the barcode generated.            |
| ForeColor     | Specifies the foreground color for the control.                           |
| Mask          | Specifies the mask used to create final image.                            |
| ModuleSize    | Specifies the length of a module (the smallest unit in a QRCode barcode). |

| Name      | Description  |
|-----------|--|
| Picture   | Returns a snapshot of the drawing in Windows Enhanced Metafile Format (EMF). |
| QRType    | Specifies the type of QR code to create.                                     |
| TargetDPI | Specifies the value of DPI when exporting the images to vector graphics.     |
| Text      | Specified the data to encode.  |
| Version   | Specifies the desired version (size) of the barcode.                         |

**Table 9.4. List of QR Code ActiveX Methods**

| Name              | Description   |
|-------------------|---|
| CopyToClipboard   | Place the drawing to clipboard in EMF format.                                 |
| ExportImageVector | Export image to a file in vector graphics format specified (EMF, EPS or SVG). |
| ExportImageRaster | Export image to a file in raster graphics format specified (PNG or BMP).      |

## ActualVersion Property

### Description

Returns the actual version of the symbol created.

### Syntax

```
object.ActualVersion
```

### Remarks

When Version specified is too small to encode the data, the program automatically select one that creates smallest barcode that fits. To find out the value of version realized, use this property.

## BackColor, ForeColor Properties

### Description

BackColor - returns or sets the background color of the control.

ForeColor - returns or sets the foreground color of the control.

### Syntax

```
object.BackColor[= Color]
object.ForeColor[= Color]
```

### Remarks

For open systems we strongly recommend to set the background color to solid white (0xFFFFFFFF) and foreground color to black (0x000000). Note: barcode requires decent contrast between the foreground color and the background color in order to be readable. Always test the readability thoroughly when you select a color pair different from black and white.

## ECLevel Property

Specifies the error correction level.

### Syntax

```
object.ECLevel = Number
```

### Description

QR Code employs Reed-Solomon error control coding to detect and repair errors. There are four user-selectable levels of error correction, as shown in the table below.

**Table 9.5. ECLevel options**

| Value | Description | Recovery Capacity (approx.) |
|-------|-------------|-----------------------------|
| 0     | Level L     | 7%                          |
| 1     | Level M     | 15%                         |
| 2     | Level Q     | 25%                         |
| 3     | Level H     | 30%                         |

## Mask Property

Specifies the pattern to be used for data masking.

### Syntax

```
object.Mask[= Number]
```

### Description

For reliable QR Code reading, it is desirable for dark and light modules to be arranged in a well-balanced manner in the symbol. QR standard allows 7 patterns (000 - 111) to be used for data masking. If the value is -1, this program evaluates all seven patterns and select the one that produces the best result.

**Table 9.6. Mask options**

| Value | Description     |
|-------|-----------------|
| -1    | Auto            |
| 0     | pattern 0 (000) |
| 1     | pattern 1 (001) |
| 2     | pattern 2 (010) |
| 3     | pattern 3 (011) |
| 4     | pattern 4 (100) |
| 5     | pattern 5 (101) |
| 6     | pattern 6 (110) |

| Value | Description     |
|-------|-----------------|
| 7     | pattern 7 (111) |

## ModuleSize Property

### Description

Returns or sets a value that determines the width and height of a single cell in the QRcode symbols generated.

### Syntax

```
object.ModuleSize[= String]
```

### Remarks

The “real estate” unit of a QR Code symbol, the *module*, is always square. This property sets both the width and the height of the square. It affects the overall symbol size.

The default value for **ModuleSize** is 20 mils. The property can be any numbers between 1 and 100.

This property is a string, and accepts the following units: mil, himetric, mm, cm, pt and inch. For example, “0.01cm” and “1mm” are all valid. If no measure unit is specified, unit of mil (which is 1/1000th inch) is assumed.

## Picture Property

### Description

Readonly property. Returns a snapshot of the drawing in Windows Enhanced Metafile Format (EMF).

### Syntax

```
object.Picture
```

### Remarks

The *Picture* property provides a convenient method to retrieve the drawing without first saving it to disk. The picture object contains an enhanced metafile handle which can be passed to *clipboard* or played on a device.

The included VC++ and C# examples use this property to retrieve an EMF handle and play it on the target printer to print the barcode.

## QRType Enumeration

Specifies the type of the QRCode.

### Syntax

```
typedef enum QRType {
    qrNormal = 0,
    qrHIBC   = 1,
```



```
} QRType;
```

### **Constants**

#### **qrNormal**

The normal QR Code.

#### **qrHIBC**

The data to encode is a HIBC LIC. The program will add + and mod 43 checksum automatically and encode the whole into a QRCode.

#### **qrMicro**

The Micro QR Code.

## **QRType Property**

Specifies the type of QR code. Currently it supports two types of QR Code symbols - normal QR and HIBC QR.

### **Syntax**

```
object.QRType = Number
```

### **Remarks**

The default value of this property is QRNormal.

## **TargetDPI Property**

Specifies the resolution of the target printer, applicable during exporting vector graphics.

### **Syntax**

```
object.TargetDPI = Number
```

### **Remarks**

When producing small sizes of barcodes, especially on low resolution printers, lengths must be aligned to the boundary of pixels in order to retain high quality. The ActiveX control adjusts the drawing units automatically based on this property when exporting vector graphics images.

## **Text Property**

Specifies the Data to be encoded into the QR Code symbol.

### **Description**

Returns or sets a string for the message to be encoded.

### **Syntax**

```
object.Text[= String]
```

### **Remarks**

## Version Property

Specifies the version of the symbol. The value ranges from 0 to 40. If the value is 0, the program selects one that encodes the data and with minimum size.

### Syntax

```
object.Version = Number
```

### Description

There are forty sizes of QR Code symbol referred to as version 1, Version 2 ... Version 40. Version 1 measures 21 modules x 21 modules, Version 2 measures 25 modules x 25 modules and so on increasing in steps of 4 modules per side up to version 40 which measures 177 modules x 177 modules.

When 0 is specified, the program selects a size that produces the most compact barcode that encodes the data.

---

**Warning** If `Version` specified is too small to encode the data, the program automatically increases it to a value that is big enough. It means that the value of property will change every time the data is changed, if it is set to 0. Therefore, you must explicitly set it back to 0 if you intend to use this value for creating the next QR code symbol.

---

## CopyToClipboard Method

Copies the QR Code image into the system clipboard, in EMF format.

### Syntax

```
obj.CopyToClipboard
```

### Description

Use this method to place a copy of barcode image to the clipboard, so that you can either retrieve it immediately in your application, or other applications such as Microsoft Word.

## ExportImageRaster Method

### Description

Exports the current drawing to a disk file, in raster image format (PNG or BMP).

### Syntax

```
obj.ExportImageRaster(filename, Number, Number)
```

### Parameters

**pixelsPerModule**

[in] Number of pixels per module.

**imageType**

[in] Image type. The current version supports two raster image formats:

- 0 - PNG
- 4 - BMP

## ExportImageVector Method

**Description**

Exports the current drawing to a disk file, in raster image format (EMF, SVG and EPS).

**Syntax**

```
obj.ExportImageVector(filename, imageType)
```

**Parameters****imageType**

[in] Image type. The current version supports two three image formats:

- 1 - SVG
- 2 - EMF
- 3 - EPS



# Chapter 10. Adding QRCode Symbols to SQL Server Reporting Service

Adding QRCode symbols to SQL Reporting Service report is straightforward with QRCode Font & Encoder 5. This chapter explains how you can achieve the objective. Our report service plugin supports SQL Reporting Service 2000, 2005 and 2008. This chapter uses Visual Studio 2008/SQL Reporting Service 2008 for demonstration. To follow the steps, you also need to have AdventureWorks sample database installed.

## Custom Assembly

SSRS can't use the native encoder DLL directly. A ready-to-use custom assemblies built under Visual Studio 2005 (.net 2.0) is included in the software.

---

**Note** In case that you need to build the assembly by yourself, The source code is also included in the distribution. The project files are packed in ReportServicePlugin\_src.zip, which is located under the program (x86) folder. The build script is written in NAnt.

---

This assembly exposes single function: QRCodeEncode. The prototype for this function is as below:

```
string QRCodeEncode(string strDataToEncode,  
int versionRequested, int ecLevel);
```

The function is quite simple: the first parameter is the data encoded, followed by the version ID requested and error correction level. The function returns the barcode string if successful, otherwise ERROR is returned.

The screen shots in this tutorial are taken from Visual Studio 2008. If you are using VS 2005, remember to change the directory to [Program Files]Microsoft Visual Studio 9.0. Other parts are identical between the two.

### Installing Custom Assembly

For Reporting Service 2000, copy the DLL into the following two directories:

- Report Designer Folder: [Program Files]Microsoft SQL Server\80\Tools\Report Designer.
- Deploy folder [Program Files]Microsoft SQL Server\MSSQL\Reporting Services \ReportServer\bin.

For Reporting Service 2005 or 2008, copy the DLL into the following directories:

- Report Designer Folder: [Program Files]Microsoft Visual Studio 9.0\Common7\IDE \PrivateAssemblies.
- Deploy folder [Program Files]Microsoft SQL Server\MSSQL\Reporting Services \ReportServer\bin.

If you have multiple SQL Server instances, you will have multiple MSSQL.n directories. Locate the one that the Reporting Service is installed under.

If you are running VisualStudio 2008, or BI Development Studio 2008, you need to perform an additional task. You need to grant additional permission in order to print preview the report. Refer to the section called “Deployment” to modify policy file `RSPreviewPolicy.config`.

## Adding QR Code to Report

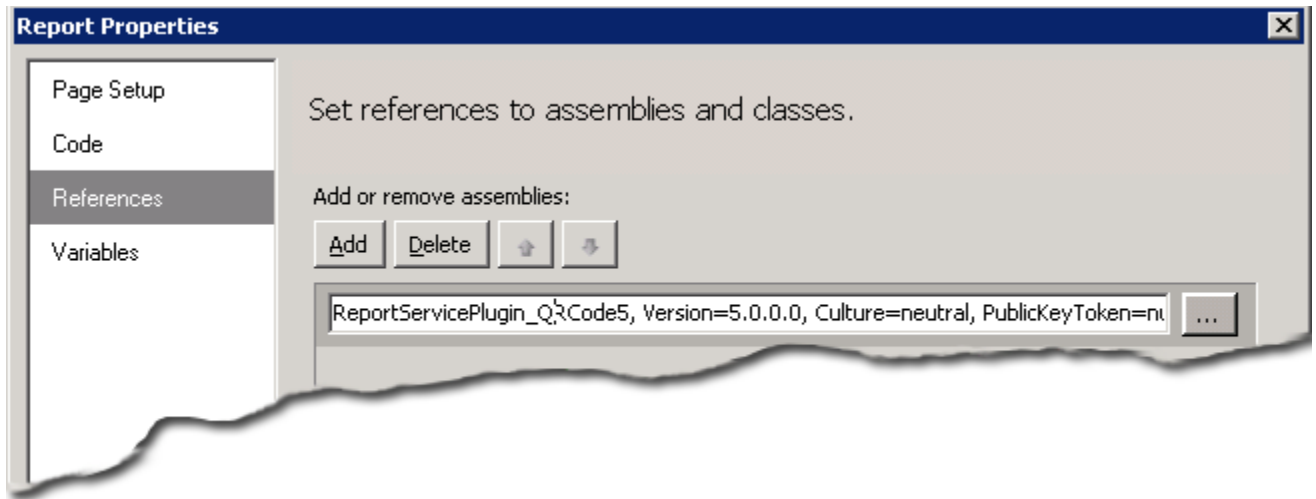
In this tutorial, we demonstrate how you can add QR Code symbols into a report. The data table is `Person.Contact`, available in AdventureWorks sample database. We'd like to present email address in QR Code barcode.

1. Open Visual Studio or SQL Server Business Intelligence Development Studio and create a new Report Server Project.
2. In Query builder, enter SQL statement `SELECT * FROM Person.Contact`.
3. Layout the report as desired. Make changes to the format of each field. Add a new column to the right that will hold the barcodes. The result looks like below:

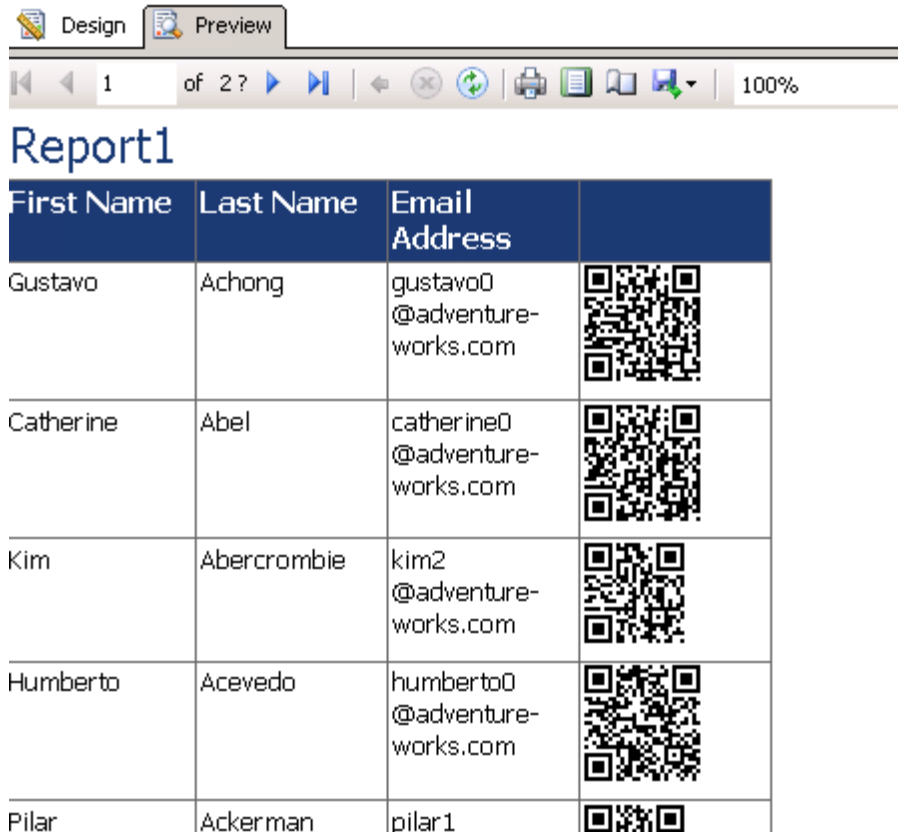


| First Name  | Last Name  | Email Address  |        |
|-------------|------------|----------------|--------|
| [FirstName] | [LastName] | [EmailAddress] | «Expr» |





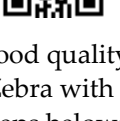
4. Select Report → Report Properties to bring up Report Properties dialog.
5. click on References tab. Navigate to the location of the customer assembly and add it.



6. Click on the new column created, and select expressions, change its value to the formula below:  
`=Morovia.ReportService.QRCodeV5.QRCodeEncode(Fields!EmailAddress.Value, 0, 0)`  
The formula calls the `QRCodeEncode` function with automatic size selection and error correction level L.
7. Now preview the report. You should see lines of hexadecimal characters at the place of the barcode.
8. Format the text box with `MRV QRCode5, 6 points`. Preview the report again, you should see the barcodes.



The screenshot shows a report preview window titled "Report1". The window has a "Design" and "Preview" tab. The report content is a table with four columns: "First Name", "Last Name", "Email Address", and an empty column. The table contains five rows of data, each with a QR code in the empty column. The data is as follows:

| First Name | Last Name   | Email Address                  |   |
|------------|-------------|--------------------------------|---|
| Gustavo    | Achong      | gustavo0@adventure-works.com   |    |
| Catherine  | Abel        | catherine0@adventure-works.com |    |
| Kim        | Abercrombie | kim2@adventure-works.com       |    |
| Humberto   | Acevedo     | humberto0@adventure-works.com  |    |
| Pilar      | Ackerman    | pilar1                         |  |

If you are printing to a laser printer, you should get a good quality barcode label printed. Now if you are printing to a low resolution thermal printer, such as a Zebra with 203 dpi in resolution, you should examine if the font size produces the optimal results. Follow the steps below:

1. According to Table 2.2, "QRCode Font Characteristics", MRV QRCode produces barcodes with X dimension at 20 mils at 6 points. Now convert it into inches and multiply the result by the printer resolution:  $20 \times 0.001 \times 203 = 4$ .
2. 4 pixels is optimized value. Therefore we can use 6 points on 203 thermal printer.

## Deployment

The Reporting Service require any custom assembly defined in the security policy otherwise a run-time error will be thrown and all you get is #Error without any explanation. Follow the steps below to change security policy. Two security policy files are required to change:

- `RSPreviewPolicy.config`. This policy file is used for `DebugLocal` preview in Visual Studio. This file is located in the Report Designer folder which is `[Program Files]Microsoft SQL Server\80\Tools\Report Designer for RS2000` and `[Program Files]Microsoft Visual Studio 9.0\Common7\IDE\PrivateAssemblies for RS2005`.
- `rssrvpolicy.config`, the policy file used for running Report Server. The file is located under `[Program Files]Microsoft SQL Server\MSSQL\Reporting Services\ReportServer\bin` directory.

Perform the following steps to add security policy required to run custom assembly:

1. Open `RSPreviewPolicy.config`, and add the following content at the end (just before two ending `CodeGroup` tags):<sup>1</sup>

```
<CodeGroup class="FirstMatchCodeGroup"
  version="1"
  PermissionSetName="FullTrust"
  Name="ReportServicePlugin_QRCode5.dll" Description="ReportServicePlugin_QRCode5.dll">
  <IMembershipCondition class="UrlMembershipCondition" version="1"
  Url="C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\ " \
  "PrivateAssemblies\ReportServicePlugin_QRCode5.dll" />
</CodeGroup>
```

Note that on RS2000 the custom assembly is located in a different directory.

2. Save the file. In Visual Studio, change the active configuration to `DebugLocal` and run the report. You should see the barcodes on the report. Examine the contents in the Output window.

If you see message A first chance exception of type 'System.Security.SecurityException' occurred in `mscorlib.dll`, the security is not configured properly.

3. After you have successfully run the report under `DebugLocal` configuration, publish the report to the Report Server. Open `rssrvpolicy.config` and add similar lines.

```
<CodeGroup class="FirstMatchCodeGroup"
  version="1"
  PermissionSetName="FullTrust"
  Name="ReportServicePlugin_QRCode5.dll" Description="ReportServicePlugin_QRCode5.dll">
  <IMembershipCondition class="UrlMembershipCondition" version="1"
  Url="C:\Program Files\Microsoft SQL Server\MSSQL.2\Reporting Services\ " \
  "ReportServer\bin\ReportServicePlugin_QRCode5.dll" />
</CodeGroup>
```

Note that you may change the file path if it is located in a different location.

4. Restart SQL Server Reporting Service and browse the report. You should see the barcodes on the report this time.



# Chapter 11. Technical Support

Morovia offers a wide variety of support services. To help you save time and money when you encounter a problem, we suggest you try to resolve the problem by following the options below in the order shown.

- Consult the documentation. The quickest answer to many questions can be found in the Morovia product documentation.
- Review the tutorial and sample applications. The tutorial steps you through the development process for a typical barcode application. The sample applications provide working code examples in several programming languages. All sample applications are extensively commented.
- Access Morovia Online. Morovia Online provides a knowledge base which documents the frequently asked questions and a web forum.

The web address for knowledge base is <http://support.morovia.com>. You can ask question at support forum at <http://forum.morovia.com>.

- Contact Morovia Technical Support Service. The Technical Support service is provided for free up to 180 days after the purchase. Email Morovia support engineers at [support@morovia.com](mailto:support@morovia.com).

---

**Note** If you purchased your software from our reseller, check to see if they provide support services before contacting Morovia.

---

Support services and policies are subject to change without notice.

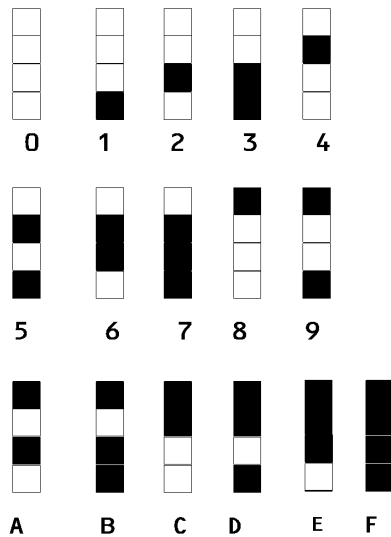


# Appendix A. Font Character Set

In some circumstances, it is desirable to substitute the font with the custom drawing routines. This section explains the mapping of each character in the font.

The basic element in a QR code is a square cell called *module*, in either black or white. If “1” represents a black module and “0” represents a white module, we can use an array of strings to represent a QR barcode. The result is further compressed by combining adjacent cells into a character. In QRCode font, four vertical modules are merged into one character, as illustrated below:

**Figure A.1. QRCode Font Character Set**



The QRCode font uses 16 characters (0-9, A-F) to represent 16 possible scenarios. To draw the QRCode barcode from the encoding results, you have two options:

- Convert the encoding results to an array of string consisting of 1 and 0 only, and draw cells from the left to the right, and from the top to the bottom.
- Create a drawing routine to draw these 16 glyphs. Scan the encoding results and call the routine one by one.

When designing drawing routines, you may also consider joining adjacent dark cells into polygons to reduce the number of drawing commands.



# Appendix B. How to Calculate Parity Value in Structural Append

The structural append feature in QR Code allows data encoded in multiple symbols (up to 16). In addition to the split data, each symbol prepends three additional fields: the sequence number, total number of symbols and the parity. The parity is calculated against the full data, and the scanner checks the parity to see if the symbols are encoded correctly after all symbols in the group are read.

---

**Warning** If you did not specify the parity value, the encoder uses a fixed value. Some scanners do not perform parity check while some do. If parity value is not specified, you should disable the parity check in the scanner.

---

You can call Font Encoder DLL function `QRCodeGetParity` or Crystal UFL function `QRCodeGetParity` to calculate the parity value. Note that the input must be the full text string encoded, not the one for a particular symbol. You can also use Parity Calculation Dialog to calculate the parity value.

Section 7.3 of ISO/IEC 18004 explains how to calculate the parity data:

The Parity Data shall be an 8-bit byte following the Symbol Sequence Indicator. The parity data is a value obtained by XORing byte by byte the byte values of all the original input data before division into symbol blocks. Mode Indicators, Character Count Identifiers, padding bits, Terminator and Pad Characters shall be excluded from the calculation. Input data is represented for this calculation by 2-byte Shift JIS values for Kanji (each byte being treated separately in the XOR calculation, most significant first) and 8-bit values as shown in Table 6 for other characters. In ECI mode the byte values obtained after any encryption or compression of the data shall be used for the calculation.



# Appendix C. Version ID Parameter (updated in version 5.2)

The QRCode encoder is enhanced with the release of version 5.2. The change enables user to specify encoding modes and specify if the encoder increases size automatically if it determines that the data cannot be encoded with the size specified. No new API is introduced. Instead, the new features are retrofitted into the version parameter.

Prior to version 5.2, the version parameter allowed is an integer between 1 and 40 which corresponds to 40 datamatrix sizes, or 0 for automatic size selection. In version 5.2, we added option to allow caller application to set additional bits to tune encoder behavior.

An integer on Windows platform is at least 32 bits. The bit numbering starts at zero for the least significant bit (LSB 0). Only 16 bits are used (LSB 15 to LSB 0). Other bits should be set to 0 for the future use.

Secondly, the 16 bits are divided into 2 bytes. The first byte, containing LSB 15 to 8, is called “modifier byte”. The modifier byte specifies additional behaviors of the encoder. The second byte, LSB 7 to 0, denotes the QRCode version number.

## Modifier byte

The 8 bits in the modifier byte are referred bit 7 to bit 0. The below lists the meaning of each bit:

Bit 7: if this bit is set, the encoder should report an error instead of increasing the internal version if it finds that the size specified cannot encode the data.

Bit 6-4: reserved for the future use and must be set to 0.

Bit 3-0: the value of the four bits determines the encoding mode selection. Some applications require the entire data to be encoded under the same mode. For those applications, choose a value that fits the requirement.

- Value 0 indicates that encoder selects the encodation mode automatically, with the goal to minimize the size required. This the behavior defined in the ISO standard, and si compatible with previous releases.
- Value 1 indicates that encoder must use the same mode throughtout; however, the mode is selected based on data entered.
- Value 2 indicates that encoder must use numeric mode throughout the symbol.
- Value 3 indicates that encoder must use alphanumeric mode throughout the symbol.
- Value 4 indicates that encoder must use byte mode throughout the symbol.

## Version byte

The value of this byte corresponds to QRCode version. A qrcode symbol may have 40 different sizes, numbered from 1 to 40.

## Example

The QRcode writer in zxing library (as of v3.3) uses a single mode for the whole symbol. In order to achieve the same result, the modifier byte should have a value of 1 (single mode, auto mode selection). If the version byte is 17 (0x11), then the value to Version parameter should be  $0x0100 \mid 0x11 = 0x0111 = 273$ .



# Appendix D. Unicode String Encode Support

The most recent QR code standard, ISO IEC 18004 2015, does not support Unicode natively. The standard states that the default character set is ISO8859-1, and ECI is required to switch to a different character set. Unfortunately, most barcode readers do not support ECI.

Because there is a demand on encoding characters outside ISO8859-1, several methods have been developed. The common approach is to encode characters in native character set, and the reader is configured to read based on the default locale. This approach produces the smallest barcode as possible with one major caveat. The same QR code is decoded into different text when read by readers with different locale configured. In many use cases this is not an issue, as a QR code with Chinese text encoded is intended to be used in China only.

The open source library zxing creates all QR codes by converting the Unicode string into UTF-8 first then encode as is. As this method is capable of encoding all Unicode characters without using ECI, and the free nature of this library, this so-called "UTF-8 encoded QR code" has become mainstream. Many 2D barcode scanners provide configuration to interpret this kind of QR codes. Some industry standards start to mandate this QR code type. For example, Swiss Payment Standards version 2.2, published in February 2021 states "the Swiss QR Code must also be UTF-8 encoded".

The QR Code encoder handles unicode string conversion when a Unicode string is passed. For example, when you call `QRCodeEncode2W` function, or use a Unicode capable components (such as Crystal Reports UFL, QR Code ActiveX etc.).

In versions after 5.1 and before 5.3.3, the input Unicode string is examined to see if there are any characters outside ISO8859-1. If none of characters are outside ISO8859-1, no conversion is performed and the string is encoded as the way that is compliant to the ISO standard. Otherwise, it converts the string to UTF-8 with BOM, then encodes the UTF-8 string in the same way as an ISO8859-1 string. This method produces standard-compliant QR codes when all characters in the string are either Latin or ASCII.

In version 5.3.3 and above, the input string is always converted to UTF8, thus creating unambiguous "UTF-8 encoded QR code". The decoded bytes should be treated as UTF-8 and processed accordingly. User can set environment variable `morovia.qrencoder5.mode` to `ISO2015` to keep the previous behavior.

Users should evaluate this change when upgrading to version 5.3.3 and above.

- If all characters in the input string are ASCII, the QR code does not change. Any readers should read the QR code correctly.
- If the string always contain characters that are neither ISO8859-1 nor ASCII, the QR code does not change. Phone readers will pick up the QR code correctly, and 2D barcode readers must be configured to set UTF-8 encode mode.
- For Latin strings (containing ISO8859-1 characters and ASCII), the QR code will change. The version prior to 5.3.3 produces standard compliant QR codes, and 2D barcode readers should read them correctly under ISO8859-1 mode or the standard-compliant mode. Phone readers generally read correctly by recognizing latin characters. In version 5.3.3 and above, UTF-8 encoded QR codes are produced.



# Appendix E. Center Overlay (Feature 50)

## Support

The feature 50 (center overlay) is introduced in version 5.3. This feature allows users to specify a pattern string in the input to construct a pattern in the center of the QR code. It supports two syntaxes:

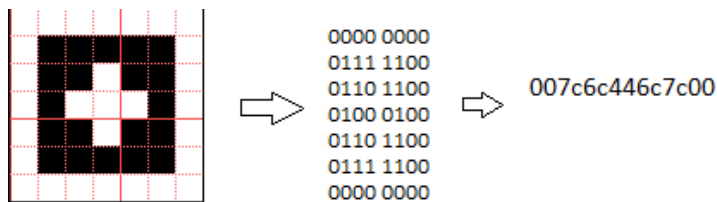
- `~50[rows,cols,pattern]`. For example `~50[7,7,007c6c446c7c]`
- `~50[:keyword]`. Currently the encoder supports only one keyword: `swiss7`. Using `:swiss7` allows user to constructs a compliant Swiss payment QR Code.

### Specify bit pattern

The first method to use the feature is to specify the exact bit pattern of overlaid image. The string format is `rows, cols, pattern`.

To get the bit string, first convert all black pixels to 1 and white to 0. Each row must end at byte boundary with 0s appended. Convert 4 bits into a hexadecimal character (0-9, a-f). Trailing zeros can be omitted.

The image below has 7 pixels per row so add 0 to make it 8 bits per row. Convert the bits to hexadecimal value we get the string `007c6c446c7c00`. Because the encoder knows the number of rows and columns from preceding parameter, the trailing zeros can be omitted. Therefore, both `007c6c446c7c00` and `007c6c446c7c` are valid.



### Limitations

In the center overlay pattern each unit has identical width and height, which are equal to the module width of the QR code (X dimension). Therefore, not all bitmaps can be implemented in this way while maintaining its size. Generally speaking, you can only implement simple bitmaps in this way, unless the QR code has a large size.

### Swiss Payment QR Code

The current Swiss payment standard requires the QR code to have a fixed size at 46 mm with a 7 x 7 mm swiss logo overlaid at the center. To implement such a QR code, you will need to come up different pattern strings for each QR code size. The keyword syntax greatly reduces the work; as all you need is to insert `~50[:swiss7]` in the data encoded, and choose the appropriate font and font size. Internally the encoder engine selects an overlay string automatically based on the final size of the QR Code.

---

**Note** The minimum Version to overlay a swiss logo is 6. If the version requested is less than 6, the engine will increase it to 6. If the version requested is less than 6 and the modifier byte forbids the increase, an error is returned.

---



### Font and Font Size

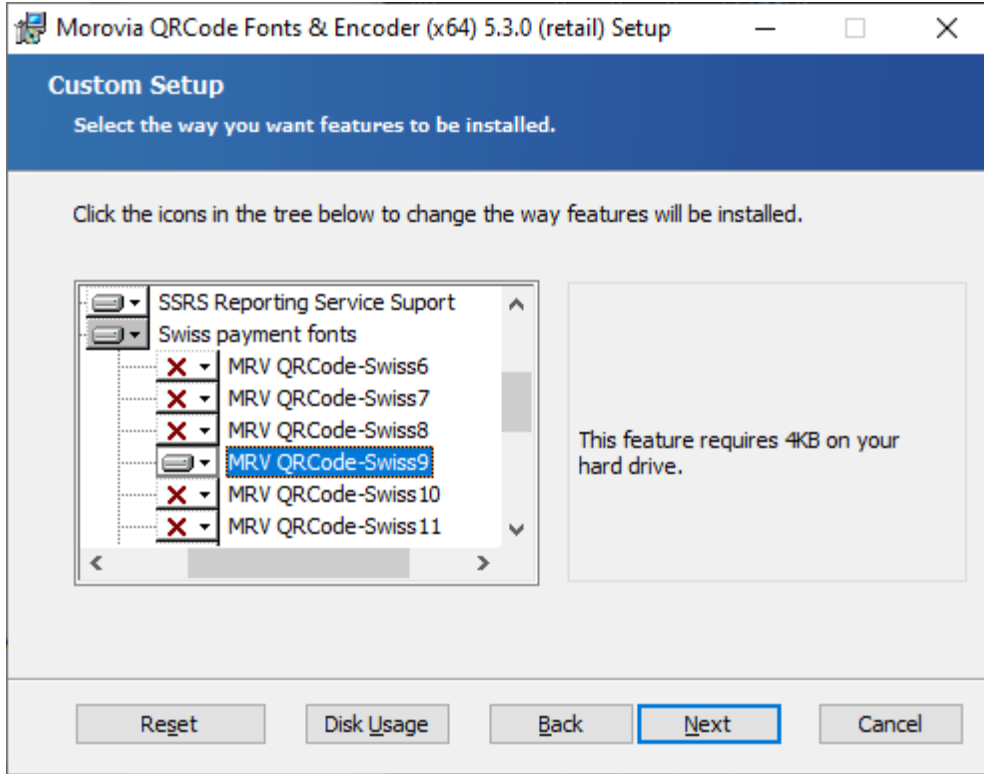
You can use the MRV QRCode font to produce the Swiss payment barcode. However you need to set the font size accordingly based on the Version of the symbol in order to achieve the required 46 mm x 46 mm size. The table below calculate the font size.

**Table E.1. QRCode Font and Size Selection**

| Version | No. of Modules | Module width (mm) | Font       | Font Size |
|---------|----------------|-------------------|------------|-----------|
| 6       | 41             | 1.12              | MRV QRCode | 12.72     |
| 7       | 45             | 1.02              | MRV QRCode | 11.59     |
| 8       | 49             | 0.94              | MRV QRCode | 10.64     |
| 9       | 53             | 0.87              | MRV QRCode | 9.84      |
| 10      | 57             | 0.80              | MRV QRCode | 9.15      |
| 11      | 61             | 0.75              | MRV QRCode | 8.55      |
| 12      | 65             | 0.70              | MRV QRCode | 8.02      |
| 13      | 69             | 0.67              | MRV QRCode | 7.56      |
| 14      | 73             | 0.63              | MRV QRCode | 7.14      |
| 15      | 77             | 0.60              | MRV QRCode | 6.77      |
| 16      | 81             | 0.57              | MRV QRCode | 6.44      |
| 17      | 85             | 0.54              | MRV QRCode | 6.14      |
| 18      | 89             | 0.52              | MRV QRCode | 5.86      |
| 19      | 93             | 0.49              | MRV QRCode | 5.61      |
| 20      | 97             | 0.47              | MRV QRCode | 5.38      |
| 21      | 101            | 0.45              | MRV QRCode | 5.16      |
| 22      | 105            | 0.44              | MRV QRCode | 4.97      |

Many software do not allow fractional font sizes. To work around the issue, version 5.3 release includes a series of true type fonts that produce the same results at 12 points. The font name is MRV QRCode-SwissN, where N is the version number. For example, font MRV QRCode-Swiss20 is designed to format a Version 20 QR Code, and the ended symbol has a size of 46mm x 46mm.

Note that those accuracy fonts are not installed by default. In order to install the fonts into the system, run the installer and select feature "Fonts for Swiss Payment". You can install each font individually.



**Table E.2. Swiss Font and Size Selection**

| Version | No. of Modules | Module width (mm) | Font               | Font Size |
|---------|----------------|-------------------|--------------------|-----------|
| 6       | 41             | 1.12              | MRV QRCode-Swiss6  | 12        |
| 7       | 45             | 1.02              | MRV QRCode-Swiss7  | 12        |
| 8       | 49             | 0.94              | MRV QRCode-Swiss8  | 12        |
| 9       | 53             | 0.87              | MRV QRCode-Swiss9  | 12        |
| 10      | 57             | 0.80              | MRV QRCode-Swiss10 | 12        |
| 11      | 61             | 0.75              | MRV QRCode-Swiss11 | 12        |
| 12      | 65             | 0.70              | MRV QRCode-Swiss12 | 12        |
| 13      | 69             | 0.67              | MRV QRCode-Swiss13 | 12        |
| 14      | 73             | 0.63              | MRV QRCode-Swiss14 | 12        |
| 15      | 77             | 0.60              | MRV QRCode-Swiss15 | 12        |
| 16      | 81             | 0.57              | MRV QRCode-Swiss16 | 12        |

| <b>Version</b> | <b>No. of Modules</b> | <b>Module width (mm)</b> | <b>Font</b>        | <b>Font Size</b> |
|----------------|-----------------------|--------------------------|--------------------|------------------|
| 17             | 85                    | 0.54                     | MRV QRCode-Swiss17 | 12               |
| 18             | 89                    | 0.52                     | MRV QRCode-Swiss18 | 12               |
| 19             | 93                    | 0.49                     | MRV QRCode-Swiss19 | 12               |
| 20             | 97                    | 0.47                     | MRV QRCode-Swiss20 | 12               |
| 21             | 101                   | 0.45                     | MRV QRCode-Swiss21 | 12               |
| 22             | 105                   | 0.44                     | MRV QRCode-Swiss22 | 12               |

# Appendix F. Fontware License Agreement

By using or installing font software (referred as "Fontware" and "SOFTWARE" in this agreement, including fonts, components, source code, install program etc.) created by Morovia Corporation (referred as "Morovia" below), you (or you on behalf of your employer) are agreeing to be bound by the terms and conditions of this License Agreement. This License Agreement constitutes the complete agreement between you and Morovia. If you do not agree to the terms and condition of the agreement, discontinue use of the Fontware immediately.

## **License Grant**

Number of Installation or Users: In consideration for the license fee paid, Morovia grants to you only, the License, the non-exclusive, non-transferable right to use the font in accordance with the license you purchase. If you are using this product for your employer, this agreement also includes your employer. You may only use the font on computers (CPUs) for which the Fontware is licensed.

The Single User License allows an individual to use the license Fontware on 1 CPU in your organization connected to any number of printers or other image producing devices. If you install or use the fonts on more than one CPU in your organization, or multiple individuals access the barcode printing functionality (e.g. printing from a printer server), multiple single user licenses must be purchased.

The Corporate License allows unlimited use of the licensed fonts in the organization that purchases it. After the Corporate License is purchased, the fonts may be installed and used on multiple CPUs in that organization without any additional fees to be paid to Morovia.

The Developer License allows 1 developer to install the fonts within his organization (Corporate License) as well as rent, lease or distribute the licensed fonts bundled with an application up to 10,000 users (formerly referred as Distribution License). The developer may not resell, rent, lease or distribute the fonts alone, they must be bundled with an application or with the application installation files. The developer may not resell, rent, lease or distribute the fonts in any way that would directly compete with Morovia. If use exceeds 10,000 concurrent users or installations, additional Developer License is required.

## **Copyright**

All title and intellectual property rights in and to the Software Product (including but not limited to any images, photographs, animations, video, audio, music, text, and "applets" incorporated into the Software Product), the accompanying printed materials, and any copies of the Software Product are owned by Morovia. All title and intellectual property rights in and to the content that is not contained in the Software Product, but may be accessed through use of the Software Product, is the property of the respective content owners and may be protected by applicable copyright or other intellectual property laws and treaties. This Agreement grants you no rights to use such content. If this Software Product contains documentation that is provided only in electronic form, you may print one copy of such electronic documentation. You may not copy the printed materials accompanying the Software Product.

## **Distribution Limits**

You must own a Developer License to distribute fonts outside your organization. You are allowed to distribute the software inside or outside your organization for up to 10,000 copies. When you distribute the software, you adhere to the following terms: (a) You may not resell, rent, lease or distribute the Software alone. The Software must be distributed as a component of an application and bundled with an application or with the application's installation files. The Software may only be used as part of, and in connection with, the bundled application. (b) You may not resell, rent, lease or distribute Software in any way that would compete with Morovia. (c) You must include the following MOROVIA copyright notice in your Developed Software documentation and/or in the "About Box" of your Developed Software, and wherever the copyright rights notice is located in the Developed Software ("Portions Copyright (c) Morovia Corporation 2004. All Rights

Reserved."). (d) You agree to indemnify, hold harmless, and defend MOROVIA, its suppliers and resellers, from and against any claims or lawsuits, including attorney's fees that may arise from the use or distribution of your Developed Software. (e) you may use the SOFTWARE only to create Developed Software that is significantly different than the SOFTWARE. (f) You must use font embedding technology when you use the SOFTWARE in PDF and WORD documents; (g) You must specify the exact domain name when creating embedded fonts for web pages. (h) All GUI programs coming with the font package are non-distributable.

### **Termination**

This Agreement is effective until terminated. This Agreement will terminate automatically without notice from Morovia if you fail to comply with any provision contained here. Upon termination, you must destroy the written materials, the Morovia product, and all copies of them, in part and in whole, including modified copies, if any.

### **Warranty & Risks**

The fonts provided by Morovia are licensed to you as is and without warranties as to performance of merchantability, fitness for a particular purpose or any other warranties whether expressed or implied. You, your organization and all users of the font, assume all risks when using it. Morovia shall not be liable for any consequential, incidental, or special damages arising out of the use of or inability to use the font or the provision of or failure to provide support services, even if we have been advised of the possibility of such damages. In any case, the entire liability under any provision of this agreement shall be limited to the greater of the amount actually paid by you for the font or US \$5.00.