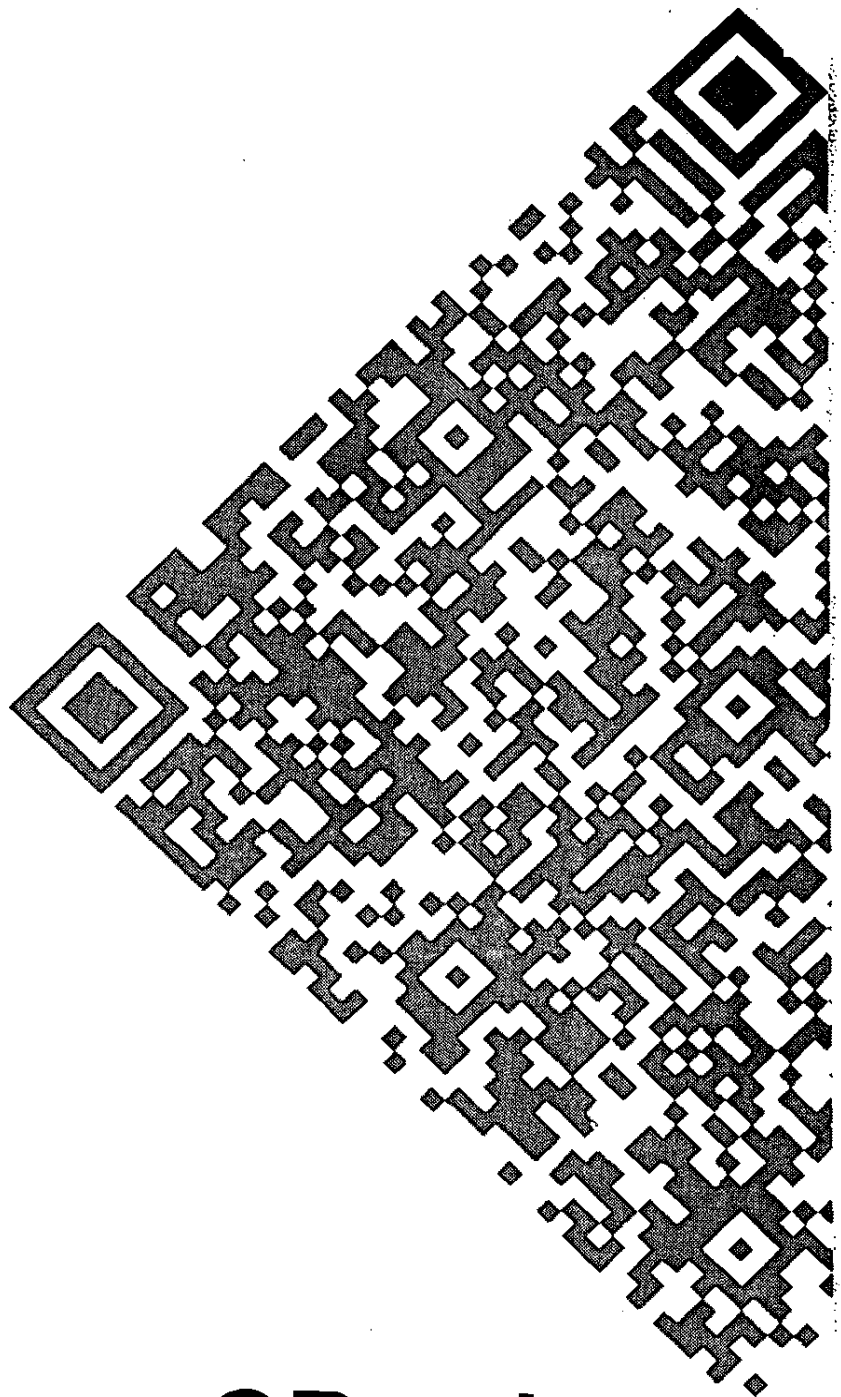


**DENSO**



# **QRmaker**

---

## **User's Manual**

Copyright © DENSO, 1998

All rights reserved. No part of this publication may be reproduced in any form or by any means without permission in writing from the publisher.

Specifications are subject to change without prior notice.

All products and company names mentioned are trademarks or registered trademarks of their respective holders.

# Preface

This manual describes how to use QRmaker which creates QR code images. QRmaker is an ActiveX control.

This manual is intended for persons who have basic knowledge of Windows®. For the details about Windows®, refer to the Microsoft Windows documentations.

## ■ Precautions on use

### - Printer resolution

You may set the printer resolution (dpi) at the CellUnit property. The default is the resolution of the default printer that has been set in the initializing sequence of the ActiveX control (OCX program). Usually use the default as is.

Mismatch between the specified value and the resolution of the printer you use actually will result in low-quality codes which are difficult to read by code readers.

### - Cell pitch

You may set the cell pitch at the CellPitch property, so you can print various sizes of codes. The default cell pitch is a nearest possible value to 0.5 mm, expressed in dots.

Even if the cell pitch is larger than the minimum resolution of your code reader, the printed codes may be difficult to read depending on the printer. To avoid this, set the cell pitch at 5 dots or more.

### - Cell correction

You may control the print size of black cells without changing the cell pitch (center-to-center distance of adjacent cells) by setting the desired cell correction value at the CellAdjust property.

For ink jet printers that will print black fat, for example, set a negative correction value. This way, you may make the black-white ratio of printed codes as close to 1:1 as possible, optimizing printed codes.

### - Creating easy-to-read QR Codes

For instructions on how to create easy-to-read QR Codes, refer to Chapter 9, Section 9.4.

#### NOTE

#### Verifying printed codes

Print quality of codes will be influenced by printer conditions and quality of paper used. Before applying created codes in practical operations, be sure to verify printed codes with verifier machines or the like.

## ■ Related Publications

QHT-1000

QHT-1000 User's Manual

BHT-BASIC3.0 Programmer's Manual

BHT-BASIC3.0 QHT Programmer's Manual

QRdraw User's Manual

## ■ Screen Indication

The lettering in the screens in this manual is a little different from that in the actual screens. File names used are only for description purpose, so they will not appear if you have not set files having those names.

# Content Overviews

<b>Preface</b> .....	<b>i</b>
<b>1. Outline</b> .....	<b>1</b>
<b>2. Features</b> .....	<b>1</b>
<b>3. PC Requirements</b> .....	<b>1</b>
<b>4. Files Required</b> .....	<b>2</b>
<b>5. Installing QRmaker</b> .....	<b>2</b>
<b>6. Properties</b> .....	<b>3</b>
6.1 Custom Properties .....	3
6.1.1 List of custom properties .....	3
6.1.2 Detailed explanations of custom properties .....	4
6.2 Stock Properties .....	10
6.2.1 List of stock properties .....	10
6.2.2 Detailed explanations of stock properties .....	10
6.3 Default Properties Provided by Visual Basic .....	10
6.4 Trapping the Property Setting Errors .....	10
6.5 QR Coding Status (by MakeStatus Property) .....	11
<b>7. Methods</b> .....	<b>12</b>
7.1 List of Methods .....	12
7.2 Detailed Explanations of Methods .....	12
<b>8. Images Returned from QRmaker</b> .....	<b>14</b>
<b>9. QR Codes</b> .....	<b>16</b>
9.1 Specifications .....	16
9.2 Code Splitter (Divide1 and Divide2 Properties) .....	18
9.3 Code Size .....	19
9.4 Creating Easy-to-read QR Codes .....	20
<b>10. Using QRmaker in Visual C++ Programming</b> .....	<b>22</b>
10.1 Setting a Data String at the InputData Property .....	22
10.2 Drawing and Saving an Image by the Picture Property .....	23
10.3 Drawing an Image by DrawQrImage Method .....	24
10.4 Drawing an Image by Render () Property of Picture Holder Type .....	24
<b>11. Using QRmaker with Microsoft Access</b> .....	<b>25</b>
11.1 Source Codes for Creating a Form .....	26
11.2 Source Codes for Creating a Report .....	26

# 1. Outline

QRmaker is an ActiveX control or OCX control which is sometimes called an OLE control according to a previous naming convention. QRmaker has the filename QRmaker.ocx and creates QR code images.

You can use QRmaker in Visual Basic 4.0 (VB), Visual C++ 4.0 (VC++), Microsoft Access95, or later versions. This manual describes how to use QRmaker mainly in VB and partially in VC++ and Access95.



**TIP** Before using QRmaker, be sure to read Readme.text to get supplemental explanations and a version history of QRmaker.

## 2. Features

If you specify a data string to be QR-coded, and parameters such as cell size by using the properties and methods, then QRmaker creates an image (picture in metafile format) and returns it to the container application in any of the following five formats you specify. You may enter a data string in the Unicode or binary format.

- (1) Picture property data
- (2) Picture directly drawn on a device specified by calling the desired method
- (3) Image copy on the Clipboard by calling the desired method
- (4) Picture drawn on a control window of the container application
- (5) Windows metafile as a storage media file

If you set the Auto Redraw mode on in developing an application program, you may visually confirm how an image will change in real time as you change the parameters, on the control window of your program.

You may define a set of persistent parameters (properties) at the start of program development by default, saving your programming time and effort.

## 3. PC Requirements

OS:	Windows 95/98 or Windows NT 4.0
CPU:	486Dx4 or higher, Pentium or later recommended
Main memory:	16 MB or more for Windows 95/98 32 MB or more for Windows NT 4.0
Hard disk:	2 MB or more space

## 4. Files Required

Filename	Description	Redispatch
QRmaker.ocx	ActiveX control	Yes
QRmaker.lic	License control file (needed for program development)	No
QRmaker.tlb	Type library TLB file (needed for program development)	No
Mfc42.dll	MFC runtime module (needed for execution of ActiveX control)	Yes
Msvcrtdll	VC runtime module (needed for execution of ActiveX control)	Yes
Regsvr32.exe	MS-DOS program to register OCX programs	Yes

The QRmaker.lic and QRmaker.tlb files are required only for developing an application program using ActiveX control. Neither of them is required at the runtime of the application.

## 5. Installing QRmaker

Unlike regular ActiveX controls, QRmaker does not include any setup program. To install QRmaker to your PC, you need to create a folder and copy QRmaker.ocx, QRmaker.lic, QRmaker.tlb, and Regsvr32.exe into the folder. You also need to copy Mfc42.dll and Msvcrtdll into the system folder in Windows 95/98 or into the system32 folder in Windows NT.

After copying those files, open the MS-DOS Prompt and switch to the directory where the folder created above is located. Next, run the Regsvr32.exe to register the QRmaker.ocx into the Windows Registry as an ActiveX control.

```
foldername>Regsvr32 QRmaker.ocx
```



## 6. Properties








### 6.1 Custom Properties

#### 6.1.1 List of custom properties

The custom properties of QRmaker are listed below.

Properties	Type	Default	Read/Write	Entry range	Definition
AutoRedraw	short	0	R/W	0 or 1	Turns the Auto Redraw mode on (if not 0) or off (if 0).
Bend	short	-1	R	0 or greater	Returns the character position following the end position of the current data string to be QR-coded.
Bstart	short	0	R	0 or greater	Returns the start position of the current data string to be QR-coded.
CellAdjust	short	0	R/W		Sets a cell correction value in dpi.
CellPitch	short	(*)	R/W	1 or greater	Sets the cell pitch in dpi.
CellUnit	short	(*)	R/W		Sets the printer resolution in dpi.
Divide1	short	1	R/W		Sets the ordinal ID number assigned to the nth split code. (Numerator)
Divide2	short	1	R/W		Sets the number of splits. (Denominator)
EccLevel	short	1	R/W	0 to 3	Sets the error correction level.
InputData	VARIANT	"QRCODE"	R/W		Sets a data string (Unicode set) to be QR-coded.
InputDataB	VARIANT	-	W		Sets a data string (Binary data) to be QR-coded.
MakeStatus	short	0	R		Returns the QR coding status.
ModelNo	short	2	R/W	1, 2, or 3	Sets the QR code model number.
NumCell	short	-	R		Returns the number of cells per side in the created QR code image.
Picture	LPPICTURES	-	R		Returns the picture image of the created QR code.
QuietZone	short	5	R/W		Sets the margin around a QR code image.
Rotate	short	0	R/W	0, 1, 2, or 3	Sets the rotation angle of a QR code image. (0°, 90°, 180°, or 270°)
TextOrBinary	short	0	R	0 or 1	Returns the current QR coding mode (Text or binary mode).
ThrowError	short	0	R/W		Defines how to handle errors if caused in other property settings.

(\*) The resolution (dpi) of the default printer that has been set in the initializing sequence of the ActiveX control will be set to the CellUnit property by default. The cell pitch (expressed in dots) that is a nearest possible value to 0.5 mm will be set to the CellPitch property by default.

Properties	Functions
MakeStatus	<p>Returns the QR coding status. If QR coding is normally processed, this property returns 0. QRMaker updates the MakeStatus value when a new QR code is created by calling any method or with the AutoRedraw property set to any non-zero value.</p> <p>You can refer to the returned value, but cannot modify or set the parameters. For details, refer to Section 6.5.</p>
ModelNo	<p>Sets the model number—1 for Model 1, 2 for Model 2, or 3 for MicroQR.</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  <p>Model 1</p> </div> <div style="text-align: center;">  <p>Model 2</p> </div> <div style="text-align: center;">  <p>MicroQR</p> </div> </div> <p>NOTE: If you set a value other than 1, 2, and 3 at this property, it will be ignored and the previous value remains effective.</p>
NumCell	<p>Returns the number of cells per side in the created QR code image.</p> <p>You can refer to the returned value, but cannot modify or set the parameters.</p>
Picture	<p>Returns the picture image of the created QR code.</p> <p>You can refer to the returned value, but cannot modify or set the parameters.</p>
QuietZone	<p>Sets the print-prohibited area surrounding a QR code image in cell units.</p> <p>Four cells or more are recommended for Model 1 and Model 2, two cells or more for MicroQR, according to the code specifications.</p> <p>NOTE: If you set any negative value, QRMaker will interpret it as 0.</p>
Rotate	<p>Sets the rotation angle of a QR code image—0 for 0°, 1 for 90°, 2 for 180°, or 3 for 270°.</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  <p>0°</p> </div> <div style="text-align: center;">  <p>90°</p> </div> <div style="text-align: center;">  <p>180°</p> </div> <div style="text-align: center;">  <p>270°</p> </div> </div> <p>NOTE: If you set a value other than 0, 1, 2, and 3, QRMaker will interpret it as 0 (0°).</p>

Properties	Functions
TextOrBinary	<p>Returns the current QR coding mode—0 for the text mode or 1 for the binary mode.</p> <p>If you set or modify the InputData property, this property returns 0; if you set or modify the InputDataB property, it returns 1.</p> <p>You can refer to the returned value, but cannot modify or set the parameters.</p>
ThrowError	<p>Defines how to handle any errors caused in other property settings.</p> <p>When any non-zero value is set at this property: At any of the properties with restricted entry ranges, if you attempt to set a value out of range, this ThrowError property will cause an error without modifying the value.</p> <p>When 0 is set at this property: Even if you set a value out of range, this ThrowError property will ignore the new setting or reset it to the default value without causing an error.</p> <p>For details, refer to Section 6.4</p>

## **Entering a data string to be QR-coded in text mode or binary mode**

You may enter a data string to be QR-coded by setting Unicode data at the InputData property (in text mode) or binary data at the InputDataB property (in binary mode).

At the InputData property, you may set ordinary printable data strings except codes assigned to 0x00, 0x81 to 0x9F, and 0xE0 to 0xFC. In text mode, QRmaker will automatically switch the coding system between numerics, alphanumerics, binary codes, and double-byte characters, thus efficiently creating QR code images as small as possible.

At the InputDataB property, you may set a binary data string only. QR coding efficiency in binary mode is lower than that in text mode. The number of encoded characters per image in the binary mode is about 30% or 60% less than that in text mode for alphanumeric or numeric characters only, respectively.

If you do not need to enter codes assigned to 0x00, 0x81 to 0x9F, and 0xE0 to 0xFC, use the text mode.

### **(Example 1) Entering text data**

```
QRmaker1.InputData="ABC123abc漢字" '13 bytes QR-coded
QRmaker1.Refresh() 'Updates the current image if the
AutoRedraw value is 0
```

### **(Example 2) Entering binary data**

```
Dim strBin as String
ReDim Bdata(9) as Byte 'Reserves 10-byte area required for
                        binary data entry
Bdata(0)=&h81 'Allows you to handle &h81=129 as
data
Bdata(1)=240
Bdata(2)=0 'Allows you to handle 0 as data
Bdata(3)=49
.....
Bdata(9)=100
strBin=Bdata() 'Assigns the binary data to the data
string area (parameter area) for
setting the property
QRmaker1.InputDataB=strBin '10 bytes QR-coded
QRmaker1.Refresh() 'Updates the current image if the
AutoRedraw value is 0
```

### **Using the Picture property**

To show you how to use the Picture property, two program examples are given below. Example 3 is to output a picture image directly to your printer and Example 4 is to assign a picture image to the picture box control.

(Example 3) Outputting a picture image by using the PaintPicture

```
QRmaker1.AutoRedraw=1           'Sets the Auto Redraw mode on
QRmaker1.InputData="Prints a QR code"
Printer.ScaleMode=6             'in mm
                                'Prints from the upper left at the
                                location x=2 and y=4 cm
Printer.PaintPicture QRmaker1.Picture,20,40
Printer.EndDoc
```

(Example 4) Assigning a picture image to the picture box control

```
QRmaker1.InputData="Prints a QR code"
QRmaker1.Refresh()              'Updates the current image if the
                                AutoRedraw value is 0
Picture1.Picture=QRmaker1.Picture 'Assigns
```

### **Note for the setting order of properties**

If you do not set the following properties in the proper order, be aware that the resulting images created by QRmaker may be different from the images you want.

(1) Setting the CellUnit property first and then the CellAdjust or CellPitch property

Set the CellUnit property first. If you set or modify the CellAdjust or CellPitch property first and then CellUnit property, QRmaker will automatically modify the preset CellAdjust or CellPitch value to preserve the original print image size.

(2) Divide1 and Divide2 properties

When both the Divide1 and Divide2 properties are set to 1 by default (code splitter deactivated), setting 2 at the Divide1 property to encode the second half of the entered data string will automatically change the Divide2 value to 2 (the same value as the Divide1 value).

This is because QRmaker automatically modifies the Divide1 and Divide2 values in order to always keep the Divide1 value (numerator) even or less the Divide2 value (denominator).

## 6.2 Stock Properties

### 6.2.1 List of stock properties

The stock properties of QRmaker are listed below.

Properties	Type	Default	Read/Write	Entry range	Definition
BackColor	long	-	R/W		Sets the background color.
hWnd	long	-	R		Returns the handle of QRmaker.

### 6.2.2 Detailed explanations of stock properties

Properties	Functions
BackColor	Sets the background color for the code image area in the QRmaker control window.
hWnd	Returns the handle to identify what ActiveX or OLE control is active in your applications.

## 6.3 Default Properties Provided by Visual Basic

Microsoft Visual Basic provides the following default properties: DragIcon, DragMode, Height, HelpContextID, Index, Left, Name, TabIndex, TabStop, Tag, Top, Visible, WhatsThisHelpID, and Width. For details, refer to the Visual Basic manuals.

## 6.4 Trapping the Property Setting Errors

Setting any non-zero value at the ThrowError property will enable error trapping in developing your applications. At any of the CellAdjust, CellPitch, Divide1, Divide2, EccLevel, ModelNo, QuietZone, and Rotate properties with restricted entry ranges, if you attempt to set a value out of range, the ThrowError property returns an error.

If an error occurs in programming, an error dialog box will appear. In runtime, an error event occurs. To trap errors in runtime, you need to write error handling routines using the ON ERROR GOTO statement in VB or the TRY or CATCH statement in VC++.

If you set 0 at the ThrowError property to disable error trapping, setting a value out of range will be ignored or will reset the property value to its default. In this case, no property will issue errors, but QRmaker may create code images according to property values different from those you have set. To avoid this, you need to recall the property values after setting them and check whether the current settings match yours.

## 6.5 QR Coding Status (by MakeStatus Property)

The MakeStatus property returns the current QR coding status. The property value will be automatically updated when a new QR code is created by calling any method (CreateQrMetaFile, DrawQrImage, QrImageCopy or Refresh) or with the AutoRedraw property set to any non-zero value.

MakeStatus returned value	Meaning
0	Normal end
-1	Too many characters defined
-2	Too much switching between character modes*
-3	Invalid code splitter specification
-4	Invalid QR code model number
-5	Miscellaneous parameter setting errors

\*The QRmaker coding system may switch between the four character modes—numerics, alphanumerics, binary, and double-byte character modes. Up to 200 mode switchings are allowed in a data string to be encoded into a single QR code or a set of split codes.

## 7. Methods

### 7.1 List of Methods

Listed below are methods available in QRmaker ActiveX control.

Methods	Type	Arguments	Type	Definition
CreateQrMetaFile	short	- Handle to the device context - Filename - Enhanced Metaswitch	HANDLE VARIANT short	Creates a Metafile.
DrawQrImage	short	- Handle to the device context - Horizontal coordinate (x) - Vertical coordinate (y)	HANDLE long long	Draws a QR code image on the selected device.
QrImageCopy	short	- Enhanced Metaswitch	short	Copies an image onto the Clipboard.
Refresh	void	- None	-	Refreshes the current image.

### 7.2 Detailed Explanations of Methods

#### [ 1 ] CreateQrMetaFile

**Syntax** short CreateQrMetaFile (HANDLE hDC, VARIANT fileName, short enhSw)

ent:	hDC	Device handle
	fileName	Filename of output metafile
	enhSw=0	Windows metafile (WMF)
	=1	Enhanced metafile (EMF)
	=2	Bitmap file (BMP)
rtn:	>0	Number of cells per side in a normally created QR code

#### Description

This method saves a created QR code image with a filename specified by fileName on the selected disk drive in the Windows metafile (WMF), enhanced metafile (EMF), or bitmap file (BMP) format specified by enhSw.

A WMF includes Aldus header information based on the Aldus Placeable Metafile, so you may insert the WMF as a picture into Microsoft Word95 documents.

An EMF cannot be inserted into Word95 documents as a picture, but can be inserted into any document created by Microsoft Office 97 applications. When saving a created code image as an EMF, this method refers to the parameters of the default printer preset in the initializing sequence of QRmaker, as reference device parameters.

For BMP files, QRmaker does not support cell correction or image rotation. The size of a created image saved as a BMP file is dependent on the number of dots set in the CellPitch property regardless of the CellUnit value.

(Example) QRmaker1.CreateQrMetaFile hDC, "c:\temp\test.wmf", 0



## [ 2 ] DrawQrImage

**Syntax**      short DrawQrImage (HANDLE hDC, long x, long y)

ent:	hDC	Device handle
	x	Print position (Horizontal distance from the left margin)
	y	Print position (Vertical distance from the top margin)
rtn:	>0	Number of cells per side in a normally created QR code

### Description

This method draws a QR code image directly on the device specified by HANDLE. The HANDLE can be any of form (Form), picture box control (Picture), and printer (Printer) of Visual Basic. The unit of the coordinates (x, y) is device pixel.

```
(Example)      Dim s As Integer, x As Integer, y As Integer
               With QRmaker1
                 S=.NumCell*.CellPitch*2.54/.CellUnit
                 Printer.CurrentX=0
                 Printer.CurrentY=0
                 Printer.Print "Draw QR code directly x=5 Cm, y=3 Cm size=";
                 Printer.Print s;" Cm"
                 x=5000/2540*.CellUnit
                 y=3000/2540*.CellUnit
                 .DrawQrImage Printer.hDC, x, y
                 Printer.EndDoc
               End With
```

## [ 3 ] QrImageCopy

**Syntax**      short QrImageCopy (short enhSw)

ent:	enhSw=0	Windows metafile (WMF)
	<>0	Enhanced metafile (EMF)
rtn:	>0	Number of cells per side in a normally created QR code

### Description

This method creates a QR code image in the WMF or EMF format and copies it onto the Clipboard.

When saving a created code image as an EMF, this method refers to the parameters of the default printer preset in the initializing sequence of QRmaker, as reference device parameters.

```
(Example)      numCell=QRmaker1.QrImageCopy(0)      '=0 WMF
```

## [ 4 ] Refresh

**Syntax**      void Refresh()

### Description

This method updates the current code image when the Auto Redraw mode is off (the AutoRedraw property is set to 0). This method automatically updates the Picture property regardless of whether the control window displays or hides.

(Example)    QRmaker1.Refresh()

## 8. Images Returned from QRmaker

QRmaker may return a QR code image to the following five objects:

	Returns to:	By:	WMF	EMF	BMP
(1)	Picture property	Using the property.	√		
(2)	Selected device	Calling a method to draw an image directly.	√		
(3)	Clipboard	Calling a method.	√	√	
(4)	Control window	Acting as .ocx control to draw an image directly.	√		
(5)	Disk file	Calling a method.	√	√	√

For displaying a QR code image on the screen of your applications, you may use the above (1) through (4), but for printing it, you may use only (1) through (3).

An image returned in (1) is drawn on the logical coordinate so that you may assign it into the picture control box or use it as an argument to the PaintPicture property that draws an image directly.

You may directly draw an image returned in (2) by specifying the hDC (device context handle) of the form (Form) or picture box control (Picture) of Visual Basic. You may print it by specifying the hDC of the printer (Printer).

An image returned in (4) may be displayed in the control window on the form (Form), so you can print it from your applications, for example, in a report processing by Microsoft Access.

When printing an image returned as the Picture property or printing an image by transferring the printer handle, if a value set at the CellPitch property is different from the resolution of the actual print device, then QRmaker will internally compensate the inconsistency by proportional interpolation or extrapolation. This calculation may involve some rounding errors so that the cell count in individual QR codes will not be constant. Even if the image looks normal, it will become difficult to read by some code readers.

This issue will prove troublesome when you print small QR codes by using low-resolution printers. Practically, if you want to set 10 or less at the CellPitch property, do not forget this issue.

To avoid the problem, you should set the same value as your printer resolution at the CellUnit property and print out a created image as is without any modification such as reduction or enlargement.

Although the cell size is not constant due to mismatch between the CellPitch value and actual printer resolution as stated above, the size of the whole code image displayed or printed is almost "correct." The "correct" means that the displayed or printed code size is just as specified by the CellUnit, CellPitch, and CellAdjust properties.

The print size (side) of a QR code image is given by the following equation:

$$\frac{(\text{CellPitch} \times \text{NumCell} + \text{CellAdjust}) \times 25.4}{\text{CellUnit}} \quad (\text{mm})$$

The print size with the quiet zone is given by the following equation:

$$\frac{(\text{CellPitch} \times (\text{NumCell} + 2 \times \text{QuietZone}) + \text{CellAdjust}) \times 25.4}{\text{CellUnit}} \quad (\text{mm})$$

# 9. QR Codes

## 9.1 Specifications

QR code is a two-dimensional matrix symbol which consists of square cells arranged in a square pattern. It is available in three models—Model 1, Model 2, and MicroQR. Model 1 and Model 2 have a position detection pattern (PDP) each in three corners, and MicroQR has it in one corner. The PDP allows code readers to quickly obtain the symbol size, position and tilt. QRmaker system supports four-level error correction and a wide range of symbol sizes.

You can specify the cell size. Model 1 has the original specifications, and Model 2 is an enhanced symbol featuring additional functions. QR code readers can automatically identify Model 1 and Model 2.

---

### Model

- Model 1: Original specifications
- Model 2: Enhanced specifications with improved position correction and large volume of data capacity
- MicroQR: Small space specifications suitable for small amount of data

---

### Encodable character sets

- (1) Numeric data (0 to 9)
- (2) Alphanumeric data (0 to 9, Uppercase A to Z, and nine special characters—space, \$, %, \*, +, -, ., /, and :)
- (3) 8-bit byte data (JIS 8-bit character set (Latin and Kana) in accordance with JIS X0201)
- (4) Kanji characters (Shift JIS values 8140h to 9FFCh and E040h to EAA4h shifted from JIS X0208)

---

### Data and cell

A black cell is binary 1 and a white cell is binary 0.

---

### Symbol size (excluding quiet zone)

- Model 1: 21 x 21 cells to 73 x 73 cells (Versions 1 to 14)
- Model 2: 21 x 21 cells to 177 x 177 cells (Versions 1 to 40)
- MicroQR: 11 x 11 cells to 17 x 17 cells (Versions M1 to M4)

For Model 1 and Model 2, each time the version increases by one, every side increases by four cells. For MicroQR, each time the version increases by one, every side increases by two cells.

---

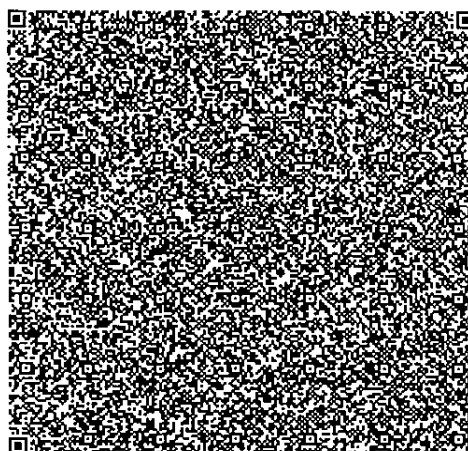
---

**Number of characters per symbol in the maximum size**

Model	Max. version	Cells/side	Numerics	Alphanumerics	Binary	Kanji
Model 1	14	73	1167	707	486	299
Model 2	40	177	7089	4296	2953	1817
MicroQR	M4	17	35	21	15	9



Model 1 in max. size  
(Ver. 14)



Model 2 in max. size  
(Ver. 40)



MicroQR in max. size  
(Ver. M4)

---

**Error correction levels selectable**

You may select the Reed-Solomon (RS) error correction level from the following four. The table below lists the maximum recoverable rate in the total codewords. (Note that these rates are approximate values.)

In some cases, QR code may not be recovered depending upon the location of dirtied or damaged area even if that area is less than the maximum recoverable rate.

Error correction level	L	M	Q	H
Max. recoverable rate	7%	15%	25%	30%

**NOTE**

MicroQR does not support error correction level H. For MicroQR version M1 with error correction level L, error detection only is possible.

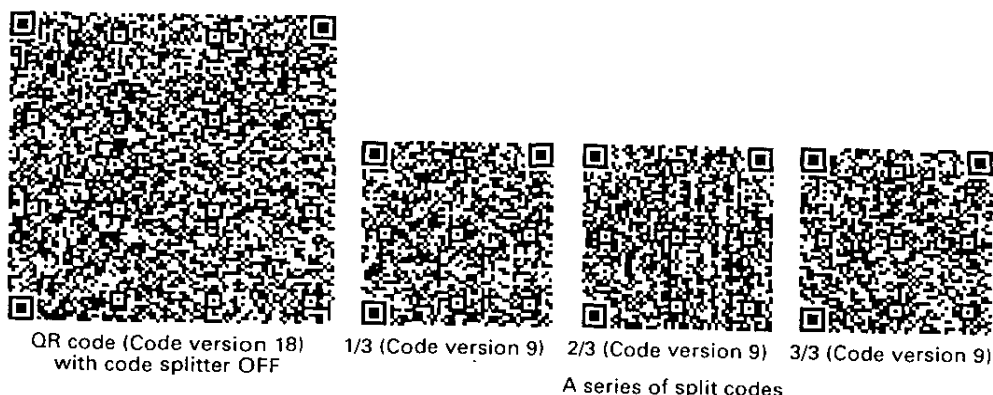
---

## 9.2 Code Splitter (Divide1 and Divide2 Properties)

QRmaker may split data into a maximum of 16 blocks and encode each of them into a split QR code to make the size of each code smaller. This feature is useful when the volume of data is too great to be encoded into a single code, when only a narrow rectangular print space is allowed, or when the code reader is limited in the reading ability of code versions. The default is to deactivate the code splitter.

Code readers supporting the code splitter can read split QR codes and restore them into the original data string.

Shown below are code image samples—a complete QR code and a series of three split codes, both of which are QR-coded from the above paragraphs (524 bytes, starting from "QRmaker may split ..... original data string."). Each of these three split codes has splitting information (1/3, 2/3, or 3/3) and a check digit produced from the original data string entered at the InputData or InputDataB property, as code internal information.



**NOTE** When creating n-split codes, do not change entered data string (InputData or InputDataB) until encoding of 1/n to n/n split codes has completed.

**NOTE** If the number of characters in text is less than the number of splits, QRmaker cannot split the text data normally. Also, in some cases, QRmaker cannot split data normally if the number of characters in text containing double-byte characters (e.g., Japanese Kanji) is less than twice the number of splits, since a double-byte character cannot be split into two.

If the number of characters in text is not exactly divisible by the number of splits, the remainder characters will be included in the last split code, so the code version of the last split code will become larger than the other split codes.

When splitting text data, QRmaker attempts to make the code version of split codes as close to the same as possible. If data volume in a split code borders on a larger version, however, the split code versions may not all be the same.

Generally, QRmaker will roughly estimate how to split data, so it may not carry out an ideal splitting in some cases where text contains alphanumerics and double-byte characters (e.g., Japanese Kanji) requiring character mode switching.

**NOTE** MicroQR cannot be split according to its specifications.

## 9.3 Code Size

### CellPitch property

At the CellPitch property, you may set the desired center-to-center distance between adjacent cells. If the CellAdjust property is set to 0, the cell pitch becomes equal to the cell size. In the initializing sequence of the ActiveX control, the cell pitch that is a nearest possible value to 0.5 mm will be set at the CellPitch property by default, expressed in dots.

**NOTE** The cell pitch of 5 dots or greater is recommended. Setting 4 dots or less to the CellPitch property may cause scanning problems depending upon the printer you use even if the cell size is higher than the minimum resolution of code readers to be used.

### CellUnit property

At the CellUnit property, you may specify the printer resolution (dpi). The resolution (dpi) of the default printer that has been set in the initializing sequence of the ActiveX control will be set at the CellUnit property by default. Changing the CellUnit value will automatically modify the cell pitch so that the same size of printout will result.

**NOTE** Use the default unit as is; do not change it. Mismatch between the value in the CellUnit property and the resolution of the printer you use will result in low-quality codes which are difficult to read by code readers. When saving a created code image as an enhanced metafile (EMF), it is essential to make the CellUnit value match the default printer resolution since QRmaker refers to the parameters preset to the default printer as reference device parameters.



### QuietZone property

At the QuietZone property, you may specify the print-prohibited area surrounding a QR code in cell units. According to the QR code specifications, the quiet zone should be at least 4 cells for Models 1 and 2, and at least 2 cells for MicroQR. The default is 5 cells.

### CellAdjust property

At the CellAdjust property, you may define a cell correction value as the number of dots calculated from the count specified at the CellUnit property. Setting a positive value will make black cells large; setting a negative value will make them small. When a negative value is set, QRmaker will create code images while keeping adjacent black cells seamless, making the print speed higher and allowing code readers to read those codes more easily.

Cell correction will bring almost no change to the overall code size.

	No correction	Correction
CellAdjust property	0	-6
Created code image (CellUnit = 600 dpi, CellPitch=30 dots)		

#### **NOTE**

Cell correction (print correction) refers to controlling the black dot size without changing the cell pitch. Use this feature, for example, when you use an ink jet printer that may print black fat so that the print size of a cell (which is a minimum drawing element) will deviate from the ideal size. To avoid this issue, set a negative correction value, and the printing results can be close to a 1:1 white and black dot size ratio.

## 9.4 Creating Easy-to-read QR Codes

- Increase the cell pitch (CellPitch property) as large as possible. For efficient operation of code readers, however, you need to take consideration so that a created QR code including the quiet zone will lie within the reader's maximum readable area with a sufficient margin.
- To decrease the code version as much as possible,
  - Use the text mode.
  - Delete meaningless characters such as appended spaces from your data string (InputData property).
  - Use uppercase letters only for alphabet characters.
  - Decrease the frequency of character mode switching by avoiding mixing numeric, alphanumeric, binary and double-byte characters. Gather the same type of characters in one area.
  - Lower the error correction level (EccLevel property) if you will use the QR code system in circumstances where there is little chance of the labels getting dirty.
- At the CellUnit property, set the same value as the resolution of the printer you use. Mismatch between the CellUnit value and the printer resolution will result in low-quality codes which are difficult to read by code readers. When saving a created code image as an enhanced metafile (EMF), it is essential to make the CellUnit value match the default printer resolution since QRmaker refers to the parameters preset to the default printer as reference device parameters.
- Adjust the cell correction value (CellAdjust property) to match the printer characteristics so that the balance of black and white in a printed image comes as close as possible a ratio of 1:1.



- If you are copying and pasting a QR code image via the Clipboard, output the image as is without any modification such as enlargement or reduction. Any modification may result in hard-to-read images.

Some applications will automatically enlarge or reduce pasted images, so be careful with the above issue about mismatches between the CellUnit value and printer resolution.

(Example) PowerPoint95 will automatically reduce pasted images to 98.3% by default, so matching the printer resolution will become meaningless. To make printed codes easier to read, you need to change the image size to 100% by using the Scale command in the Draw menu.

- Images pasted via the Clipboard into Word95/Word97 documents may be edited by double-clicking. However, do not edit the picture but output it as is. Entering the picture edit mode will greatly change the cell size of the image, making the printed code hard to read by code readers.

## 10. Using QRmaker in Visual C++ Programming

To use QRmaker as an ActiveX control in your Visual C++ program, do either of the following two:

- (1) Insert the ActiveX control into a project dialog to create CQRmaker and CPicture classes concurrently.
- (2) Insert the ActiveX control following the class wizard to create the member function `m_QRmaker` that recognizes QRmaker as an available ActiveX control in your Visual C++ program.

Given below are samples on how to use the properties and methods with the member function `m_QRmaker`.

### 10.1 Setting a Data String at the InputData Property

A sample coding given below produces a function that will be executed the moment a data string is entered into the Edit control box or the data string in the control box is modified. The function converts the data string to a data string composed by the Unicode character set, forms Variant type data, and sets the converted string into the `InputData` property.

```
void CTestOCX2View::OnChangeEditInputdata()
{
    CString buf;
    CEdit* edit=(CEdit*)GetDlgItem(IDC_EDIT_INPUTDATA);
    edit->GetWindowText(buf);
    int _convert;
    CString tmp=A2W(buf);           // Converts the current data string into Unicodes
    m_QRmaker.SetInputData(COLEVariant(tmp).Detach());
}
```

To use the function `A2W()` that converts a double-byte coded data string to a Unicode coded string, you need to add `#include<afxpriv.h>` line to the `stdafx.h` line.

## 10.2 Drawing and Saving an Image by the Picture Property

The description below gets the picture handle returned by .ocx and draws the image.

```
CPicture pict = m_QRmaker.GetPicture();
HMETAFILE hmf = NULL;
int type = pict.GetType(); //Gets the handle type
int height = pict.GetHeight(); //HIMETRIC unit
int width = pict.GetWidth(); //HIMETRIC unit
//Determines the metafile handle from the CPicture type information
if (type == PICTYPE_METAFILE) { //Asks if the picture is a metafile
    hmf = (HMETAFILE)pict.GetHandle();
    CDC *pDC = GetDC();
    int xLongPixPerInch = pDC->GetDeviceCaps(LOGPIXELSX);
    int yLongPixPerInch = pDC->GetDeviceCaps(LOGPIXELSY);
    pDC->SetMapMode(MM_ANISOTROPIC);
    pDC->SetWindowExt(width, -height);
    pDC->SetViewportExt(width * xLongPixPerInch / 2540,
                       height * yLongPixPerInch / 2540);
    pDC->PlayMetaFile(hmf); //Draws picture
    ReleaseDC(pDC); //Never forget to add this line
}
```

After saving the metafile handle hmf into the OnButtonXX() or similar property as a class variable obtained by the way stated above, if you attempt to use this hmf in the OnDraw() property, the hmf may be used normally in the front window but may not be used normally in reactivated windows hidden once. This is because any metafile handle hmf which has come into an idle loop will be discarded. To avoid this, write all program lines within an OnDraw() property.

To get a metafile handle that will not be discarded even in an idle loop, that is, to save the Picture property for future reuse, use the CopyMetaFile() property to copy the metafile handle into the m\_hmf variable obtained as a class variable.

```
hmf=(HMETAFILE)pict.GetHandle();
if(m_hmf != NULL)DeleteMetaFile(m_hmf); //Deletes the metafile
m_hmf=CopyMetaFile(hmf, NULL); //Copies the metafile into a main memory area
```

Run this m\_hmf on the OnDraw() property by using PlayMetaFile() property, then you can draw an image normally anytime.

## 10.3 Drawing an Image by DrawQrImage Method

The description below allows you to draw an image by using the DrawQrImage method.

```
m_QRmaker.DrawQrImage((long)(pDC->m_hDC), 0, 0);
```

## 10.4 Drawing an Image by Render() Property of Picture Holder Type

The description below gets a picture dispatch of .ocx.

```
CPictureHolder m_pictHolder;  
CPicture pict = m_QRmaker.GetPicture();  
LPDISPATCH lpdisp = pict.m_lpDispatch;  
m_pictHolder.SetPictureDispatch((LPPICTUREDISP)lpdisp);
```

To get a set of parameters which defines the image size, describe as follows:

```
int m_unit = m_QRmaker.GetCellUnit();  
int m_ncell = m_QRmaker.GetNumCell();  
int m_pitch = m_QRmaker.GetCellPitch();  
int m_height = pict.GetHeight();           //HIMETRIC unit  
int m_width = pict.GetWidth();             //HIMETRIC unit
```

Now, you can draw any image using the picture folder type variable obtained above. For the rect, define an image drawing area beforehand.

```
CRect rect(offsetX, offsetY, newSize+offsetX, newSize+offsetY);  
m_pictHolder.Render(pDC, rect, rect);
```

To use the CPictureHolder property, you need to add a #include<afxctl.h> line to the stdafx.h file.

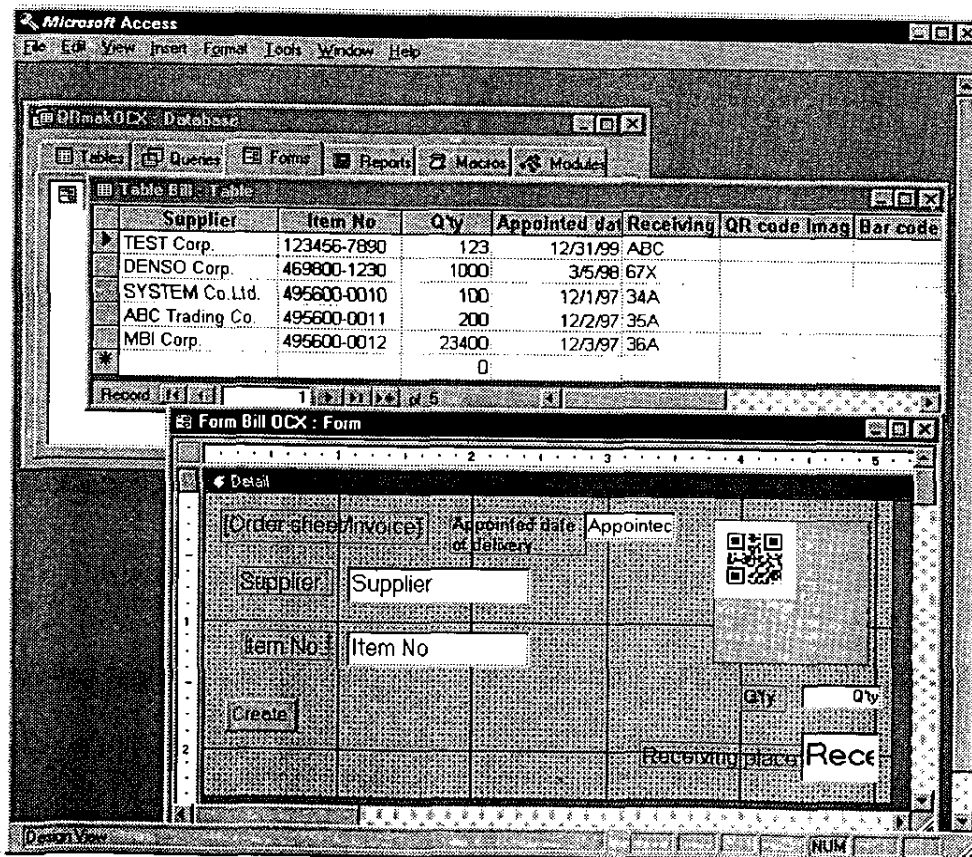
## 11. Using QRmaker with Microsoft Access

This chapter gives examples using QRmaker with Microsoft Access95.

When designing a form or report sheet with Access95, you may use QRmaker control by choosing the Custom Control command from the Insert menu and pointing QRmaker control. Access95 will locate the QRmaker window on the active form sheet as shown below.

Display the properties of the control where you click the Others or All tab, and you may refer to the properties of QRmaker or modify their default values. Values requiring no change at runtime should be fixed here.

Shown below is a sample screen where you create or edit an ordering record and bill individual orders.



# **QRmaker**

---

## **User's Manual**

11T500©

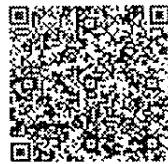
First Edition, October 1998

DENSO CORPORATION  
Electronics Applied Products Division

---

The purpose of this manual is to provide accurate information in the handling and operating of QRmaker. Please feel free to send your comments regarding any errors or omissions you may have found, or any suggestions you may have for generally improving the manual.

In no event will DENSO be liable for any direct or indirect damages resulting from the application of the information in this manual.



## **DENSO CORPORATION**

Electronics Applied Products Division

1-1 Showa-cho, Kariya-city, Aichi, Japan 448-8661

496995-0080