

Woosim Windows SDK Programmer Reference

Version 4.0.1

July 2016



Contents

1. OVERVIEW	3
1.1. PURPOSE.....	3
1.2. GET STARTED.....	3
1.3. DEVELOPMENT ENVIRONMENT	4
1.4. DEFINITIONS AND ABBREVIATIONS	4
2. SUMMARY	5
2.1. INFORMATION	5
2.2. MESSAGE HANDLING.....	5
2.3. DEFINITION VALUES.....	5
2.4. WOOSIM WINDOWS LIBRARY APIS	6
2.4.1. <i>Printable Data Handling APIs</i>	7
2.4.2. <i>Printer Mode & Setting APIs</i>	9
2.4.3. <i>Miscellaneous Device Handling APIs</i>	10
3. WOOSIM WINDOWS APIS.....	12
3.1. PRINTABLE DATA HANDLING APIS	12
3.2. PRINTER MODE & SETTING APIS	20
3.3. MISCELLANEOUS DEVICE HANDLING APIS	28
4. SAMPLE CODES	32
4.1. SAMPLE TEST APPLICATION	32
APPENDIX	33
APPENDIX A. BACKWARD COMPATIBILITY	33
APPENDIX B. CONVERT DATA TYPE BETWEEN C/C++ AND .NET LANGUAGE	33
APPENDIX C. TWO-DIMENSIONAL BARCODE TABLE.....	34

1. Overview

This release of the Woosim printer Windows Software Development Kit (SDK) documentation provides information about Windows OS based application development.

Copyright © 2015-2016 Woosim System Inc.

1.1. Purpose

This document introduces the APIs that is not used the hexadecimal command directly. It was created to more convenient and easy to use than the function provided by *Woosim Command manual*. Also, this document explains the sample project created by using the APIs.

1.2. Get Started

The SDK consists of the library files, sample projects, and the document.

A user definitely should choose the library according to the language and the OS, but it may be limited by the build environment such as a CPU type.

Woosim printers provide two types of the printer modes: Standard mode and Page mode. If a user wants to print the data immediately, standard mode is right answer. Or, if a user wants to print the result of the designed data, Page mode should be selected. The Woosim Windows SDK provides the printer mode API, and controls the data printing by the internal buffer in the library.

The Woosim Windows SDK supports the languages to support MFC DLL and static library such as Visual Basic, C++ and C#.

1. Visual C++

As the purpose, two types of libraries are provided: Dynamic link library and static library. For the examples to make, please refer to the sample project inside.

1) Dynamic linking library

In the Woosim Windows SDK, DLL files for user library, LIB files for linker input, Header files including APIs and the related information. In the Woosim Windows SDK, DLL files consisted of the library, LIB files for linker input, Header files including APIs and the related information. A user should create the project by adding "header file" and ".lib file" containing the information related to the library.

2) Static library

The library file is used when the user creates an application including the Woosim Windows library. User should be created the project by adding the "header file" and "library file".

2. .NET language (Visual C# and Visual Basic)

The conversion of the data type is required: Because it does not provide the data type of the common C / C ++. Before use this APIs, the declared information in the SDK's header file to the appropriate data type in .net languages must be changed. Please refer to *Appendix B* to see the matching table associated with the data type converting.

Please refer to the sample project included in the SDK for the example of changing the data type declaration and detailed API.

1.3. Development Environment

Target platform	Windows XP x86 / Vista x86, x64 / 7 x86, x64 / 8 x86, x64
Target Language	Visual C++, Visual C#, and Visual Basic

1.4. Definitions and Abbreviations

API	Application Interface Unit
Bpp	Bit per pixel
DBCS	Double Byte Character System
DLL	Dynamic Link Library
ECC	Error Correction Code
HRI	Human Readable Interpretation
MCU	Main Control Unit
MFC	Microsoft Foundation Class
Misc.	Miscellaneous
MSR	Magnetic Stripe Reader
SCR	Smart Card Reader
SDK	Software Development Kit
TTF	True Type Font

2. Summary

2.1. Information

Woosim Windows SDK presents the sample project including the source code as well as the executable files. The results of the function usage by the sample executable file with the functions in the library can be checked. As your environment to use the executable file included in the SDK to confirm the result, "Visual Studio C++ 2008 SP1 Redistributable Package" may be required.

The SDK supports M16C, ARM, and RX MCU's printer. However, the old MCU, i.e. M37702 provides limitedly. Most of APIs are based on the RX MCU printer. Your printer's MCU can be identified through the self-test.

2.2. Message Handling

All Woosim printers are organized to send some data to user terminal. The library receives the message in the following format to be provided clearly between the various messages on the inside of Windows.

```
MSG_RECEIVE_CHECK      _T("WOOSIM_PRT_OK")
UWM_RECEIVE_CHECK = ::RegisterWindowMessage(MSG_RECEIVE_CHECK)
```

Throughout the registered message, the printer sends the message with two parameters to the user terminal. The first parameter is sent the information including the data type and the length of the data. The received data type has one of the following values.

STATUS_ANSWER	0x53
DATA	0x44

Identified messages data is transferred in the type of the hexadecimal data to user terminal via another parameter. The data has different data and the data length depending on the data requested by the user. Please refer to the sample project attached to the SDK for more information.

2.3. Definition values

The API in the library has used as some predefined value for several API's return value, or for user convenience. The followings are predefined value separated by the types.

● Return Message

<Serial, Bluetooth Error>

UNABLE_TO_OPEN_THE_PORT	-2
UNABLE_TO_CONFIGURE_THE_SERIAL_PORT	-3
UNABLE_TO_SET_THE_TIMEOUT_PARAMETERS	-4

<WiFi Error>	
SOCKET_ERROR	-2
CONNECT_FAIL	-3

<USB Error>	
UNABLE_TO_MAKE_CONNECTION	-2
UNABLE_TO_GET_INFOMATION	-3
NOT_WOOSIM_PRINTER	-4

<Common Message>	
SUCCESS	1
ALREADY_OPENED	-1
TIMEOUT	-7
NOT_OPEN_THE_PORT	-11

● Barcode Type

UPC-A	65
UPC-E	66
EAN13	67
EAN8	68
CODE39	69
ITF	70
CODABAR	71
CODE93	72
CODE128	73

● Paper Width

_1INCH	192
_2INCH	384
_3INCH	576
_4INCH	832

● MCU

MCU_RX	2
MCU_ARM	1
MCU_M16C	0

2.4. Woosim Windows Library APIs

The API provided in the Woosim windows SDK can be categorized as follows; Printable Data Handling, Printer mode & setting, and Miscellaneous Device Handling APIs.

In the section of Printable Data Handling, it consists of the sets of the APIs sending data to printer and the printable data in Woosim printers.

Printer mode & setting section consists of the setting of the printable data, and the setting value of the mode offered from the printer.

Finally, Misc. Device Handling section consists of the API regarding the connection to user terminal and the extra devices that can be connected to printer.

The library is using the method to send data at once from a collected data in the library's buffer to printer. In the document, the buffer is referred to the "printing buffer" to distinguish for it from the printer buffer. Most of the APIs in library send the command character and the data to printer throughout this buffer. The "printing buffer" provides 200KB for storage. All data from a user's terminal would be stored in the storage.

Except for some APIs, most of them don't have any return value. Also the usage of the wrong method can be occurred errors; Printing data not wanted or No printing data. Therefore, it is recommended to read the document or sample project carefully before using the API.

2.4.1. Printable Data Handling APIs

- void **ClearSpool()**
Clear all the printing buffer data.

- void **CompressedBmpSaveSpool**(char* bmpFilePath)
Convert bitmap data to user defined bit image format data using compressed algorithm.
And then the data is stored in printing buffer.

- void **ControlCommand**(BYTE* Cmds, int Cmds_length)
The byte stream data is stored in printing buffer.

- void **DataMatrixSaveSpool**(int width, int height, int module, char* barcodeData)
Create printable DataMatrix 2D barcode data and store in printing buffer.

- void **GS1DatabarSaveSpool**(int type, int n, char* barcodeData)
Create printable GS1 databar data and store in printing buffer.

- void **LoadLogoSaveSpool**(int n)
Load pre-downloaded image in the printer.

- void **MaxicodeSaveSpool**(int mode, char* barcodeData)
Create printable Maxicode 2D barcode data and store in printing buffer.

- void **MicroPDF417SaveSpool**(int width, int column, int row, int ratio, char* barcodeData,
 BOOL HRI)
Create printable Micro PDF417 2D barcode data and stored the printing buffer.

- void **NormalBmpSaveSpool**(char* bmpFilePath)
Convert bitmap data to user defined bit image format data. And then the data is stored in printing buffer.

- void **OneDimensionBarcodeSaveSpool**(BYTE ucBarcodeType, int width, int height, BOOL HRI, char* barcodeData)
Create a printable "ucBarcodeType" type 1D barcode data and stored the printing buffer.
- void **Page_DotFeed**(int dots)
In page mode, move the writing offset position to next "dots" dots in vertical direction and create new line.
- void **Page_DrawBox**(int iXPos, int iYPos, int width, int height, int thickness)
Create printable straight line or box image data and store in printing buffer.
- void **Page_DrawEllipse**(int iXPos, int iYPos, int A_length, int B_length, int thickness)
Create printable ellipse image data and store in printing buffer.
- void **Page_DrawLine**(int iXPos, int iYPos, int iX2Pos, int iY2Pos, int thickness)
Create printable line connecting two points and store in printing buffer.
- void **Page_LineFeed**(int lines)
In page mode, move the writing offset position to the next "lines" lines.
- void **Page_Newline**()
In page mode, move the writing offset position to the next line.
- void **Page_Print**()
In page mode, print all data.
- void **Page_PrintStandardMode**()
In page mode, print all data and return to standard mode.
- void **PDF417SaveSpool**(int type, int width, int column, int level, int ratio, char* barcodeData, BOOL HRI)
Create printable PDF417 2D barcode data and store in printing buffer.
- void **PrintData**()
Print data and feed one line.
- void **PrintDotFeed**(int dots)
Print data and feed "dots" dots.
- void **PrintLineFeed**(int lines)
Print data and feed "lines" lines.
- int **PrintSpool**(BOOL bDelete_Spool)
Send data in the printing buffer to Woosim printer.
- void **PrintSpoolForTTF**(char* sData, BYTE iXFontSize, BYTE, iYFontSize)
Print the text data using true type font.

- void **QRCodeSaveSpool**(int version, char level, int module, char* barcodeData)
Create printable QR code 2D barcode data and store in printing buffer.
- void **TextSaveSpool**(char* textData)
Store the text data in the printing buffer.
- void **TruncatedPDF417SaveSpool**(int width, int column, int level, int ratio, char* barcodeData, BOOL HRI)
Create printable Truncated PDF417 2D barcode data and store in printing buffer

2.4.2. Printer Mode & Setting APIs

- void **GetPrinterModelName**()
Get the printer model name and MCU information from Woosim printer.
- void **GetFirmwareVersion**()
Get the printer firmware version and build date from Woosim printer.
- int **GetPrinterStatus**(int iTimeoutMsec)
Get current printer status data from Woosim printer.
- int **GetPrinterStatusEx**(int iTimeoutMsec)
Get current printer status data with battery information from Woosim printer.
- void **InitLineSpace**()
Set the line height as default value.
- void **InitPageMode**(int iXPos, int iYPos, int width, int height)
Initialize the printer state, change to page mode, delete current page mode data, and create printable area in page mode.
- void **InitPrinterStatus**()
Initialize the printer buffer and user configuration data.
- void **Page_ClearCurrentData**()
In page mode, clear all data.
- void **Page_SetArea**(int iXPos, int iYPos, int width, int height)
In page mode, create printable area.
- void **Page_SetDirection**(int n)
In page mode, set print direction.
- void **Page_SetPosition**(int iXPos, int iYPos)
In page mode, set the printing data position.
- void **Page_SetStandardMode**()

In page mode, change printer mode from page mode to standard mode.

- void **SetAbsPosition**(int distance)
Move the printing position from the beginning of the line.
- void **SetCharCodeTable**(int n, int MCU)
Set character code table.
- void **SetCharSpace**(int n)
Set right-side character spacing.
- void **SetFontForTTF**(char* ttfFile)
Select the true type font.
- void **SetFontSize**(int n)
Select the printer font size.
- void **SetLineSpace**(int n)
Set each line height value.
- void **SetPageMode**()
Change printer mode from standard mode to page mode.
- void **SetTextAlignment**(int n)
Set the text justification: left, center and right.
- void **SetTextStyle**(int underline, BOOL emphasize, int width, int height, BOOL reverse)
Set the text style: under line, emphasizing, character sizing, and reversing.
- void **SetUpsideDown**(BOOL set)
Set upside down printing.

2.4.3. Miscellaneous Device Handling APIs

- void **CancelMSRMode**()
Cancel magnetic card reading mode.
- void **CancelSCRMode**()
Cancel smart card reading mode.
- BOOL **ClosePrinterConnection**()
Close connection between user terminal and printer.
- int **ConnectSerialPrinter**(char* sPortName, int iBaudRate, int iTimeoutMsec, BOOL bProtocol)
Connect the Woosim printer using serial communication method via a COM port.
- int **ConnectUSBPrinter**(int iTimeoutMsec, BOOL bProtocol)

Connect the Woosim printer using USB communication method in user terminal.

int **ConnectWirelessPrinter**(char* sIP_ADDR, int iPortNum, int iTimeoutMsec, BOOL bProtocol)

Connect the Woosim printer using IP communication method in user terminal.

void **CutPaper**(int mode)

Select cut mode, and then cut the paper.

void **EnterMSRMode**(int n)

Enter the magnetic card "n" track reading mode.

void **EnterSCRMode**()

Enter the smart card reading mode.

void **FeedToMark**()

Print data and feed the paper to the black mark position.

void **SetPositionFromMark**(int distance)

Set distance from the black mark.

3. Woosim Windows APIs

The APIs provided in the Woosim Windows SDK save all the data to the buffer called by "printing buffer" in the library except for several APIs. The contents in the buffer are delivered to the printer, or deleted by `PrintSpool()` or `ClearSpool()`.

The followings are the list of the APIs that do not save the contents at the printing buffer.

<code>CancelMSRMode</code>	<code>CancelSCRMode</code>
<code>ConnectSerialPrinter</code>	<code>ConnectUSBPrinter</code>
<code>ConnectWirelessPrinter</code>	<code>EnterMSRMode</code>
<code>EnterSCRMode</code>	<code>GetPrinterModelName</code>
<code>GetPrinterStatus</code>	<code>GetFirmwareVersion</code>

Except for the functions associated with the connection between the user terminal and the printer, the others operate immediately. And then the result of operation is transferred to the terminal.

3.1. Printable Data Handling APIs

void **ClearSpool()**

It can remove the contents in the printing buffer of the library by using the function. The function does not delete the data in the printer buffer.

void **CompressedBmpSaveSpool**(char* bmpFilePath)

It converts the data of the designated bitmap file under 200KB to the type of the user-defined bit image to use data compression algorithm, and saves the converted data in the printing buffer. It is similar with the operation of `NormalBmpSaveSpool()`, but operates faster in Standard Mode.

Parameter

bmpFilePath The bitmap file path for printing.

void **ControlCommand**(BYTE* Cmds, int Cmds_length)

It transfers the received byte stream to the printing buffer. If it is used, please refer to the *Woosim Command Manual*.

Parameter

Cmds The byte stream of the command.

Cmds_length The length of byte streams.

```
void DataMatrixSaveSpool( int width, int height, int module, char* barcodeData)
```

It creates 2D barcode data to be printable as DataMatrix type, and saves to the printing buffer. Please refer to the *Appendix* for the data range as the type or the length of the barcode.

Parameter

<i>width</i>	The barcode width (0: auto size).
<i>height</i>	The barcode height (0: auto size).
<i>module</i>	The module size (1~8).
<i>barcodeData</i>	The DataMatrix 2D barcode source data.

```
void GS1DatabarSaveSpool( int type, int n, char* barcodeData )
```

It creates 2D barcode data to be printable as GS1 Databar type, and saves to the printing buffer. GS1 Databar can be used in only RX MCU printers to use the firmware released since December, 2012.

Parameter

<i>type</i>	The type of GS1 Databar (0~6): GS1 Databar Omnidirectional, Truncated, Stacked, Stacked Omnidirectional, Limited, Expanded, and Expanded Stacked.
<i>n</i>	Segments per row(2~20). This value only for type 6 and should be even number.
<i>barcodeData</i>	The GS1 Databar barcode source data. If select the GS1 type 0 to 4, maximum data length is less than 14. GS1 type 5 and 6, this type should comply with the data standard of the GS1 General Specifications. For AI, use '[' and ']' instead of '(' and ')'. '

```
void LoadLogoSaveSpool( int n )
```

It calls the pre-downloaded image in the printer. The image should be uploaded to the printer prior to calling it. If not uploading or selecting the image number not uploaded, the function can be ignored.

Parameter

<i>n</i>	The numeric value of image for use The number of value is limited by MCU type.
----------	---

void **MaxicodeSaveSpool**(int mode, char* barcodeData)

It creates 2D barcode data to be printable as Maxicode type, and saves to the printing buffer. Maxicode can be used in only RX MCU printers to use the firmware released since December, 2012. For more information, please refer to the *Woosim Command Manual*.

Parameter

<i>mode</i>	The mode of Maxicode (2~6).
<i>barcodeData</i>	The Maxicode 2D barcode source data.

void **MicroPDF417SaveSpool**(int width, int column, int row, int ratio, char* barcodeData, BOOL HRI)

It creates 2D barcode data to be printable as Micro PDF417 type, and saves to the printing buffer. Please refer to the *Appendix* regarding the data amount that can be represented regarding the rows and the columns of barcode.

Parameter

<i>width</i>	The barcode width (1 ~ 8). If the width set out of printable area, barcode is not printed.
<i>column</i>	The column number of Micro PDF417 barcode (1~4).
<i>row</i>	The row number of Micro PDF417 barcode (4~44, 0: auto size).
<i>ratio</i>	The horizontal and vertical ratio (2~5).
<i>barcodeData</i>	The Micro PDF417 2D barcode source data.
<i>HRI</i>	The HRI character printing option. If it is true, numeric values are printed at the bottom of barcode

void **NormalBmpSaveSpool**(char* bmpFilePath)

It converts the designated bitmap file under 200KB to the user-defined bit image type, and saves the converted data in the printing buffer of the library. It is recommended to call the function in Page Mode for a large image. It can prevent loss of images, and move to the desired position.

Parameter

<i>bmpFilePath</i>	The bitmap file path for printing.
--------------------	------------------------------------

void **OneDimensionBarcodeSaveSpool**(BYTE ucBarcodeType, int width, int height, BOOL HRI, char* barcodeData)

It creates the data to be printable as 1D barcode type of ucBarcodeType, and saves to the printing buffer.

Parameter

<i>ucBarcodeType</i>	1D barcode type (65~73). Refer to below table to see what barcode can be available.
<i>width</i>	The barcode width (1 ~ 8) If the width is out of printing area or width value, barcode is not printed
<i>height</i>	The barcode height by dot unit (0~255)
<i>HRI</i>	The HRI character printing option. If it is true, numeric values are printed at the bottom of barcode.
<i>barcodeData</i>	The 1D barcode source data. Each barcode type has different data length or range of the value. For more information, refer to following table.

Type #	Barcode System	Number of characters	Remarks
65	UPC-A	$11 \leq n \leq 12$	$48 \leq d \leq 57$
66	UPC-E	$11 \leq n \leq 12$	$48 \leq d \leq 57$
67	EAN13	$11 \leq n \leq 13$	$48 \leq d \leq 57$
68	EAN8	$7 \leq n \leq 8$	$48 \leq d \leq 57$
69	CODE39	$1 \leq n \leq 255$	$48 \leq d \leq 57$ $65 \leq d \leq 90$ $d = 32, 36, 37, 43, 45, 46, 47$
70	ITF	$1 \leq n \leq 255$ (even number)	$48 \leq d \leq 57$
71	CODABAR	$1 \leq n \leq 255$	$48 \leq d \leq 57$ $65 \leq d \leq 68$ $d = 36, 43, 45, 46, 47, 58$
72	CODE93	$1 \leq n \leq 255$	$0 \leq d \leq 127$
73	CODE128	$2 \leq n \leq 255$	$0 \leq d \leq 127$ $d=C1H$ (FNC1) $d=C2H$ (FNC2) $d=C3H$ (FNC3) $d=C4H$ (FNC4)

void **Page_DotFeed**(int dots)

In Page Mode, it moves the writing offset position to the next "dots" dots in the vertical direction and creates a new line.

Parameter

dots The paper feed length by dot unit (0~255).

void **Page_DrawBox**(int iXPos, int iYPos, int width, int height, int thickness)

It creates a printable straight line or box image data, and saves to the printing buffer. It works in only Page Mode. Moreover, a user can draw a horizontal line or a vertical line by the height or width value to zero.

Parameter

iXPos The x-coordinate of the origin in dot unit

iYPos The y-coordinate of the origin in dot unit

width The width of rectangle

height The height of rectangle

thickness The rectangle border thickness

void **Page_DrawEllipse**(int iXPos, int iYPos, int A_length, int B_length, int thickness)

It creates printable ellipse image data and saves in the printing buffer. The function is only worked in Page Mode. The adjustment of the horizontal length A and the vertical length B can draw a variety of shapes and sizes of the ellipse. For more information on the set of the value, please refer to the following image.

Parameter

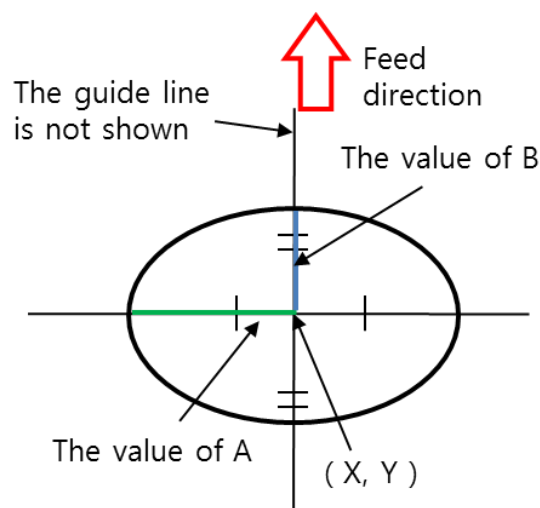
iXPos The x-coordinate of the origin in dot unit

iYPos The y-coordinate of the origin in dot unit

A_length The half the length of the x-axis of the ellipse

B_length The half the length of the y-axis of the ellipse

thickness The ellipse border thickness

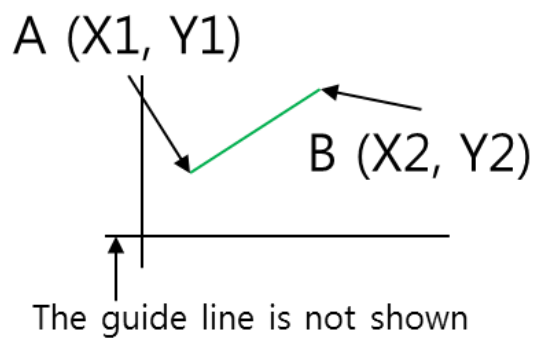


void **Page_DrawLine**(int iXPos, int iYPos, int iX2Pos, int iY2Pos, int thickness)

It creates a printable line connecting two points and saves in the printing buffer. The function works in Page Mode only, and draws a line to connect points A (X1, Y1) and B (X2, Y2).

Parameter

<i>iXPos</i>	The X1-coordinate of the point A in dot unit
<i>iYPos</i>	The Y1-coordinate of the point A in dot unit
<i>iX2Pos</i>	The X2-coordinate of the point B in dot unit
<i>iY2Pos</i>	The Y2-coordinate of the point B in dot unit
<i>thickness</i>	The line thickness



void **Page_LineFeed**(int lines)

In Page Mode, the function moves the writing offset position to the next "lines" line. The length of paper feed is affected by SetFontSize() or SetLineSpace().

Parameter

<i>lines</i>	The paper feed length by line unit (0~255).
--------------	---

void **Page_Newline**()

In Page Mode, this function moves the writing offset position to the next line according to current line spacing. The length of paper feed is affected by SetFontSize() or SetLineSpace().

void **Page_Print**()

In Page Mode, It prints current page mode area data.

If the page area in Page Mode is not set, it prints the paper to length of default page area.

The vertical length of default page area is 2400dot (30cm).

void **Page_PrintStandardMode**()

In Page Mode, it prints all data, and returns to standard mode.

If the page area in page mode is not set, it prints the paper to length of default page area.

The vertical length of default page area is 2400dot (30cm).

void **PDF417SaveSpool**(int width, int column, int level, int ratio, char* barcodeData, BOOL HRI)

It creates printable PDF417 2D barcode data and saves in the printing buffer.

Parameter

<i>width</i>	The barcode width (1 ~ 8). If the width is out of printing area or width value, barcode is not printed.
<i>column</i>	The column number of PDF417 barcode (1~30).
<i>level</i>	The security level to restore when barcode image is damaged (0~8).
<i>ratio</i>	The horizontal and vertical ration (2~5).
<i>barcodeData</i>	The PDF417 2D barcode source data.
<i>HRI</i>	The HRI character printing option. If it is true, numeric values are printed at the bottom of barcode.

void **PrintData**()

In Standard Mode, it prints the data and feeds the paper by one line in accordance with the current line spacing. The length of paper feed is affected by SetFontSize() or SetLineSpace().

void **PrintDotFeed**(int dots)

It prints data and feeds "dots" dots in Standard Mode.

Parameter

<i>dots</i>	The paper feed length by dot unit (0~255).
-------------	--

void **PrintLineFeed**(int lines)

It prints the data and feed "lines" lines in Standard Mode.. The length of paper feed is affected by SetFontSize() or SetLineSpace().

Parameter

<i>lines</i>	The paper feed length by line unit (0~255).
--------------	---

int **PrintSpool**(BOOL bDelete_Spool)

It sends data in the printing buffer to a printer. It sends all data in the printing buffer to a printer and the printer will process the data using received command data. Also the printer should choose to clear or not the data in the printing buffer using the received parameter value.

Parameter

<i>bDelete_Spool</i>	The initialization values of the printing buffer. If this value is set, the printing buffer initializes all data after sending the data to a printer.
----------------------	---

Return value

SUCCESS	Data transfer success.
NOT_OPEN_THE_PORT	The printer is not connected.

void **PrintSpoolForTTF**(char* sData, BYTE iXFontSize, BYTE, iYFontSize)

It prints the text data inputted using TTF set through SetFontForTTF(). The printed font size is used for "iXFontSize" and "iYfontSize".

Parameter

<i>sData</i>	The data user wants to print.
<i>iXFontSize</i>	The font width size
<i>iYFontSize</i>	The font height size

Void **QRCodeSaveSpool**(int version, char level, int module, char* barcodeData)

It creates 2D barcode data to be printable as QR code type, and saves to the printing buffer. For the level of 4kind, each version data and EC level is different. Please refer to QR code table in *Appendix* section.

Parameter

<i>version</i>	The version of symbol (0~40, 0: auto size)
<i>level</i>	The EC level (L: 7%, M: 15%, Q: 25%, H: 30%)
<i>module</i>	The module size (1~8)
<i>barcodeData</i>	The QR code 2D barcode source data. The data length and range is dependent on version and level value.

void **TextSaveSpool**(char* textData)

It is saved the text data for printing to the printing buffer in the library.

Parameter

<i>textData</i>	The text data user wants to print.
-----------------	------------------------------------

Void **TruncatedPDF417SaveSpool**(int width, int column, int level, int ratio, char* barcodeData,
BOOL HRI)

It creates printable Truncated PDF417 2D barcode data and saves in the printing buffer.

Parameter

<i>width</i>	The barcode width (1 ~ 8). If the width is out of printing area or width value, barcode is not printed.
<i>Column</i>	The column number of Truncated PDF417 barcode (1~30).

<i>Level</i>	The security level to restore when barcode image is damaged (0~8).
<i>Ratio</i>	The horizontal and vertical ration (2~5)
<i>barcodeData</i>	The Truncated PDF417 2D barcode source data.
<i>HRI</i>	The HRI character printing option. If it is true, numeric values are printed at the bottom of barcode.

3.2. Printer Mode & Setting APIs

void **GetPrinterModelName()**

It gets the printer model name and MCU information from the printer.

It transfers the hexadecimal response-data from a printer to user terminal. User can see the same data at result of the self-test.

void **GetFirmwareVersion()**

It gets the printer firmware version and build date information from the printer.

It transfers the hexadecimal response-data from a printer to user terminal. User can see the same data at result of the self-test.

int **GetPrinterStatus(int iTimeoutMsec)**

When it called, the printer sends hexadecimal data to user terminal as response result. The response-hexadecimal data is contained the sensor bit data in the target printer. For the additional information, please refer to *Woosim Command manual*.

Parameter

<i>iTimeoutMsec</i>	The valid time for data request.
	When iTimeoutMsec is over, it fails.

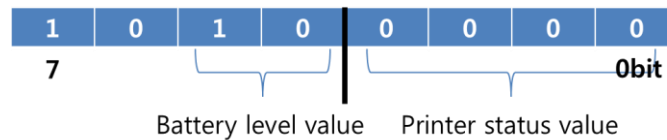
Return value

SUCCESS	The data transfer success.
TIMEOUT	The data transfer fails in valid time.

int **GetPrinterStatusEx(int iTimeoutMsec)**

It returns the battery information with the current status from the battery.

This method delivers hexadecimal data including the current printer status and the battery charging information to the user terminal different from the response result of GetPrinterStatus(). Therefore, it is not guaranteed to get normal response regarding products not to use a battery.



The size of the response result is 1byte(8 bits), and 2 kinds of data are delivered as divided to the upper 4 bits and the lower 4 bits. The upper 4 bits returns the battery residual, and can be checked by the 5th and 6th.

The followings show the battery residual approximately.

Voltage	Response
~ 7.3v	0x80 (1000 0000 ₂)
7.3v ~ 7.6v	0x90 (1001 0000 ₂)
7.6 ~ 8.0v	0xA0 (1010 0000 ₂)
8.0v ~	0xB0 (1011 0000 ₂)

※ The errors can be occurred as the environment and the measurement.

The lower 4 bits mean the values of the printer sensors. Please refer to the *Woosim Command Manual* for the values.

Parameter

iTimeoutMsec The valid time to get data. If it is over, the request fails.

Return value

SUCCESS The data transmission succeeds.

TIMEOUT The data transmission fails.

void **InitLineSpace()**

It initializes the current text spacing as default value 30 dots.

The value is not affected by any printer mode, but affected the current printer font size.

void **InitPageMode (int iXPos, int iYPos, int width, int height)**

It initializes the printer state, changes page mode, clears previous page mode data, and sets printable page area. The Page Mode can be easily set up.

The function can be replaced by the following functions: InitPrinterStatus(), SetPageMode(), Page_ClearCurrentData(), and Page_SetArea().

Parameter

iXPos The horizontal starting position to print in dot unit

iYPos The horizontal starting position to print in dot unit

Width The width values of printable area by dot unit. This value is always upper then 0. The maximum horizontal length is dependent on the target printer or the paper.

Height The width values of printable area by dot unit. This value is always upper then 0. The maximum vertical length is 2400dot (30cm).

Void **InitPrinterStatus()**

It initializes the printer buffer and user configuration data.

It is initialized the printer settings set using the API, such as "SetTextAlignment()" or "SetCharSpace()" and cleared the printable data in the printer buffer. But it is not cleared the data in the printing buffer. Also, it does not initialize mechanical settings and macro settings.

void **Page_ClearPageData()**

A user can erase the printer buffer data that containing the currently working contents in page mode by using this function. But this function is not clear the printing buffer in the library.

The function works in Page Mode only.

void **Page_SetArea(int iXPos, int iYPos, int width, int height)**

It sets the printable area in the page mode.

The starting position (iXPos, iYPos) is ignored when (iXPos, iYPos) value goes out of paper range.

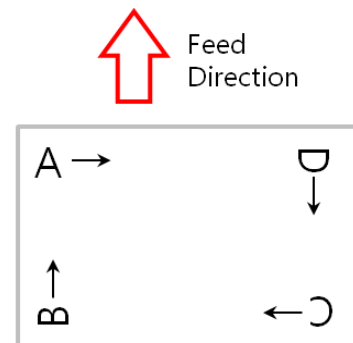
Parameter

<i>iXPos</i>	The x-coordinate of the printable area's starting position. This value is in dot unit.
<i>iYPos</i>	The y-coordinate of the printable area's starting position. This value is in dot unit.
<i>width</i>	The width values of printable area by dot unit. This value is always upper then 0. The maximum horizontal length is dependent on the target printer or the paper.
<i>height</i>	The width values of printable area by dot unit. This value is always upper then 0. The maximum vertical length is 2400dot (30cm).

void **Page_SetDirection**(int *n*)

In the page mode, it sets the print location and the direction on the pre-defined printable area by Page_SetArea(). The function should rotate counterclockwise.

<i>n</i>	Print direction	Starting position
<i>0</i>	Left to right	Upper left (A in the figure)
<i>1</i>	Bottom to top	Lower left (B in the figure)
<i>2</i>	Right to left	Lower right (C in the figure)
<i>3</i>	Top to bottom	Upper right (D in the figure)



Parameter

n Feed direction and print direction setting value (0~3).

void **Page_SetPosition**(int *iXPos*, int *iYPos*)

In page mode, it sets the printing data position.

Parameter

iXPos The horizontal starting position in dot unit

iYPos The vertical starting position in dot unit

void **Page_SetStandardMode**()

In page mode, it changes the printer mode from page mode to standard mode. Standard mode is default printer mode. It immediately prints the received input data in a line or dot units. Standard Mode has a faster print speed than the page mode.

The function works in the page mode only.

void **SetAbsPosition**(int *distance*)

The function moves the printed data's starting position. The movement sets if the data printing starts at a distance much horizontally in one line. If the movement is out of the printable range, the function is ignored.

Parameter

distance The horizontal distance from the starting position.

void **SetCharCodeTable**(int *n*, int *MCU*)

The function selects the code table of the characters to be printed. It has a close relationship with the printer MCU. Printers using M16C or ARM MCU is using only 7 code table. The printers using RX MCU, on the other hand, are supported to be able to use the code table more than the others. For more information, please refer to the following table.

Parameter

n The value of code table for use.

MCU The value of printer MCU (M16C: 0, ARM: 1, RX: 2).

- M16C, ARM MCU version

n	Character Code Table	Remark (size)
0	Page 0 [CP437 (USA, Standard Europe)]	Font-A: 12x24 Font-B: 9x24
1	Page 1 [Katakana]	
2	Page 2 [Multilingual CP850]	
3	Page 3 [Portuguese CP860]	
4	Page 4 [ISO8859-15 (Latin9)]	
5	Page 5 [Polish]	
255	DBCS (Double Byte Character System) ** One of them is installed of blank.	Font-A: Kor(24x24) Chn_Big5 (16x16) Chn_GB2312 (16x16) Jpn_Shift JIS (24x24)

※ The DBCS is only provided Font-A.

- RX MCU version

n	Character Code Table	Remark (size)
0	Page 0 USA, Standard Europe [CP437]	Font-A: 12x24 Font-B: 9x24 Font-C: 8x16
1	Page 1 Katakana	
2	Page 2 Multilingual(Latin-1) [CP850]	
3	Page 3Portuguese [CP860]	
4	Page 4 Canadian-French [CP863]	
5	Page 5 Nordic [CP865]	
6	Page 6 Slavic(Latin-2) [CP852]	
7	Page 7 Turkish [CP857]	
8	Page 8 Greek [CP737]	
9	Page 9 Russian(Cyrillic) [CP866]	
10	Page 10 Hebrew [CP862]	
11	Page 11 Baltic [CP775]	
12	Page 12 Polish	
13	Page 13 Latin-9 [ISO8859-15]	
14	Page 14 Latin1[Win1252]	
15	Page 15 Multilingual Latin I + Euro[CP858]	
16	Page 16 Russian(Cyrillic)[CP855]	

17	Page 17 Russian(Cyrillic)[Win1251]	
18	Page 18 Central Europe[Win1250]	
19	Page 19 Greek[Win1253]	
20	Page 20 Turkish[Win1254]	
21	Page 21 Hebrew[Win1255]	
22	Page 22 Vietnam[Win1258]	
23	Page 23 Baltic[Win1257]	
24	Page 24 Azerbaijani	
25 ~ 29	Reserved	
30	Thai[CP874]	12x24 9x24 (same as Page 0) 8x16 (same as Page 0)
31 ~ 39	Reserved	
40	Page 25 Arabic [CP720]	16x24 9x24 (same as Page 0) 8x16 (same as Page 0)
41	Page 26 Arabic [Win 1256]	
42	Page 27 Arabic (Farsi)	
43	Page 28 Arabic presentation forms B	
44 ~ 49	Reserved	
50	Page 29 Hindi_Devanagari	16x24 9x24 (same as Page 0) 8x16 (same as Page 0)
255	DBCS (Double Byte Character System) ** One of them is installed of blank.	Kor(24x24, 16x24) Chn_Big5 (24x24) Chn_GB18030 (24x24) Jpn_Shift JIS (24x24)

※The Korean character code table is provided Font-A (24x24) and Font-B (16x24). Other DBCSs only provided Font-A (24x24).

void **SetCharSpace**(int n)

It sets up the right-side character spacing. The function is used for the inter-character spacing, and is not affected by the printer mode.

Parameter

n The character right-side spacing value by dot unit (0~255).

void **SetFontForTTF**(char* ttfFile)

It is used that user want to print using TTF files instead of using the printer default font.

Before using it, the printer should be stored the TTF file. The parameter is TTF file name in the printer.

Parameter

ttfFile The TTF file name to use.

void **SetFontSize**(int n)

It selects the font size.

The printer can resize the print font. If MCU type is M16C or ARM, the printer only provided 2 type of font size. The RX MCU provides 3 type font sizes. The font size is dependent on the printer character code table. So, the font size is limited by some character code table. Based on the page 0 in character code table, the font size is as follows:

	RX	ARM, M16C
Font-A	12x24 dots	12x24 dots
Font-B	9x24 dots	9x24 dots
Font-C	8x16 dots	Not Supported

Before change the font size, please check the font table in use. The DBCS font has different font size from other character code table font. For more information about font-size, please refer to the table in the SetCharCodeTable().

When it used, the printer is required attention because automatic initializing character related settings such as font-style, line spacing or etc.

Parameter

n The Font size for printing (0~2): Font-A, Font-B, and Font-C

void **SetLineSpace**(int n)

The value affects to the result to print about text data. If setting value n is less than the default value, the text data can be missing. It can be set independently in Standard Mode and Page Mode.

Parameter

n The line spacing in dot unit. (0~255)

void **SetPageMode**()

It is changed the mode from standard mode to page mode.

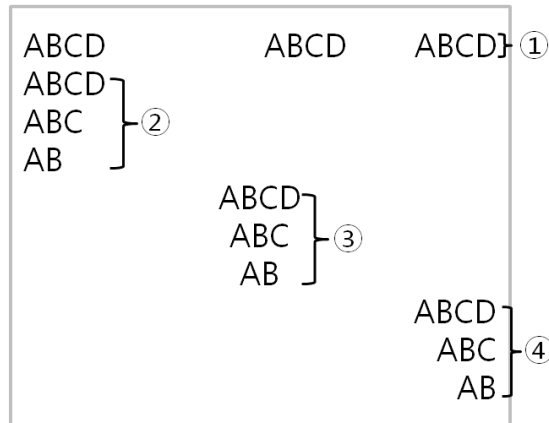
The Page Mode is stored the printable data in the printer internal buffer and prints at once. Also, it can be set the print data positon. This mode can create various types of printable data than Standard Mode.

It only works in Standard Mode.

void **SetTextAlignment**(int n)

It is set the text alignment to print out. The function keeps working until changing to another position.

Please refer to the following image for more information.



In the figure, ②, ③, and ④ is shown left, center and right alignment each other.

In case of ①, it shows the result of all alignment on a line. In the case of center alignment, the difference of ① and ③ is caused by differences in the recognition area. User must be aware of these characteristics before use.

Parameter

n The text justification number(0~2): Left, Center, and Right.

void **SetTextStyle**(int underline, BOOL emphasize, int width, int height, BOOL reverse)

It sets the text style: under line, emphasizing, character sizing, and reversing.

Parameter

underline The underline setting value (0~2): Turn off, 1dot underline, and 2 dots underline.

emphasize If this value is set, each character is emphasized.

width The number of extension value on horizontal direction. For more information, refer to following table.

height The number of extension value on vertical direction. For more information, refer to following table.

reverse Set the reverse mode. True set black/white reverse mode, false otherwise.

n	Width	Height
0	1 (Normal)	1 (Normal)
1	2 (Double width)	2 (Double height)
2	3	3
3	4	4

4	5	5
5	6	6
6	7	7
7	8	8

void **SetUpsideDown**(BOOL set)

It sets upside down mode. The function is only used in Standard Mode.

Upside-Down off



Upside-Down on



Parameter

set

The flag of upside-down.

3.3. Miscellaneous Device Handling APIs

void **CancelMSRMode**()

The function works in the track reading mode.

void **CancelSCRMode**()

It cancels the SCR mode.

BOOL **ClosePrinterConnection**()

It closes the connection between the terminal and the printer.

This function will terminate the connection at any connection mode, and also it release allocated printing buffer to the library. So, when user disconnects the connection using this function, please use after confirming the end of the print job.

Return value

TRUE The connection release success.

FALSE The connection release failure.

int **ConnectSerialPrinter** (char* sPortName, int iBaudRate, int iTimeoutMsec, BOOL bProtocol)

It connects the printer to use the serial communication in the terminal.

Also, when the user wants to connect a printer using Bluetooth serial method, this function provides the connection between user terminal and the printer. Prior of the use, the user must check the printer settings.

Parameter

<i>sPortName</i>	Serial port number.
<i>iBaudRate</i>	The printer transmission speed. (9600bps, 19200bps, 38400bps, 57600bps, and 115200bps)
<i>iTimeoutMsec</i>	If do not connect within this time is regarded as the port is not available.
<i>bProtocol</i>	This value is always FALSE.

Return value

SUCCESS	The printer connection is success.
ALREADY_OPENED	Printer connection already opened.
UNABLE_TO_OPEN_THE_PORT	The port is disabled.
UNABLE_TO_CONFIGURE_THE_SERIAL_PORT	Printer initializing failed.
UNABLE_TO_SET_THE_TIMEOUT_PARAMETERS	R/W time setting failed.
TIMEOUT	The connection fails within the valid time.

int **ConnectUSBPrinter**(int iTimeoutMsec, BOOL bProtocol)

It connects the printer to use USB communication method in the terminal.

The function connects the printer by using USB. It only works in the Woosim Printer.

ConnectSerialPrinter() should be used in the case of serial to USB like FTDI.

Parameter

<i>iTimeoutMsec</i>	If do not connect within this time is regarded as the port is not available.
<i>bProtocol</i>	This value is always FALSE.

Return value

SUCCESS	Success to connect
ALREADY_OPENED	Already connected
UNABLE_TO_MAKE_CONNECTION	Unable to make a connection
UNABLE_TO_GET_INFOMATION	Unable to initialize printer information

NOT_WOOSIM_PRINTER

This printer is unable to use.

TIMEOUT

The connection fails within the valid time.

int **ConnectWirelessPrinter** (char* sIP_ADDR, int iPortNum, int iTimeoutMsec, BOOL bProtocol)

It connects the printer by using IP communication method in user terminal.

The function connects the printer by using WIFI. Before using the function, the printer configuration must be checked.

Parameter

sIP_ADDR Printer's IP address

iPortNum Printer's port number

iTimeoutMsec If do not connect within this time is regarded as the port is not available. (Min value: 1000msec)

bProtocol This value is always FALSE.

Return value

SUCCESS Success to connect

SOCKET_ERROR Socket initialization failed.

CONNECT_FAIL Printer connection failed.

ALREADY_CONNECTED Printer already connected.

TIMEOUT The connection fails within the valid time.

void **CutPaper**(int mode)

It selects the cut mode, and then cut the paper.

It works normally in the printer with auto-cutting

Parameter

mode Select the mode(0~1): full cut or partial cut

void **EnterMSRMode**(int n)

It Run the MSR mode at the setting track.

Woosim printers support three types of card track mode on the MSR: 12Track, 23 Track, and 123 Track.

The following table shows the track information in each mode.

n	12 Track	23 Track	123 Track
0	Set 1Track	Set 2Track	Set 1Track
1	Set 2Track	Set 3Track	Set 2Track

2	Set 12Track	Set 23Track	Set 12Track
3	X	X	Set 123Track
4	X	X	Set 3Track

For more information on the data transmitted through reading, please refer to the *Magnetic Card Data Output Format* area of *Woosim Command manual*.

Parameter

n Select card track (0~4).

void **EnterSCRMode()**

It enters into the SCR mode.

void **FeedToMark()**

It prints the data, and feeds the paper until black mark is shown.

If the distance "n dots" which moves from black mark by the SetPositionFromMark function is set, it will feeds out the paper until

It works properly when the black mark sensor is set in printer settings. If user uses the no-black mark paper, the printer feeds out paper until default distance. The default distance is 30cm.

void **SetPositionFromMark(int distance)**

It sets the moving distance from black mark of the paper.

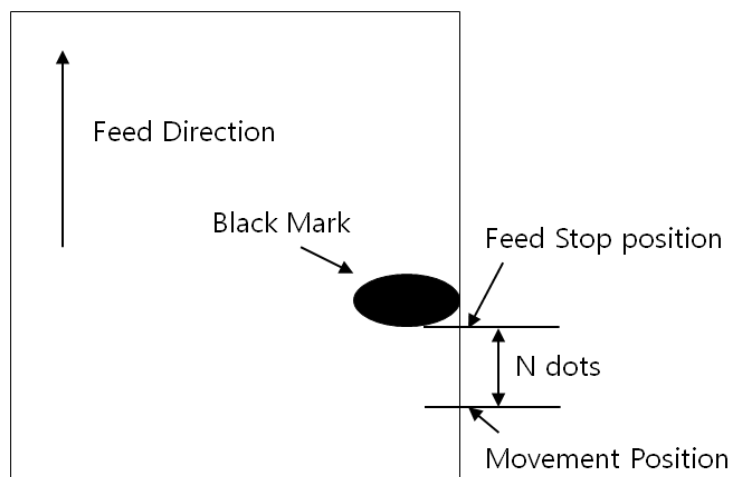
The function works only when black mark sensor is enabled in the printer configuration.

Please change the printer setting, and must equip with the correct paper before using.

It changes the hardware settings, so should not use repeatedly. It can be occurred a problem when using FeedToMark(). Therefore, it is recommended to use only to set the distance.

Parameter

distance The value of "distance" dot shifted from black mark.



4. Sample Codes

Woosim Windows SDK provides the following sample application to make the user convenient.

4.1. Sample test application

It is a traditional sample application. It includes following functions:

- Text data printing
- 1bpp bitmap printing
- Various 1D and 2D bar code printing
- MSR mode setting and show the magnetic card data by swiping
- Complex form data printing

The sample application is written in three languages such as Visual Basic, Visual C #, and Visual C ++. Also each sample project is written with the same content. Before using the sample application, the printer connecting status should be checked. Through the self-test result or the printer's LCD, the connection type of the printer can be checked. If wanting to connect the printer by using Bluetooth, it must check the allocated port number from the terminal and then should connect.

Prior to the barcode printing test, it should consider the MCU of the printer to use. In the Woosim Printer, RX MCU allows printing out all kind of presented barcode, but other MCU is not allowed to print on some 2d barcode.

The information of MCU is shown as a result of self-test. Also it can be seen as the hexadecimal data in user terminal by "GetPrinterModelName()". In the case of using MSR mode, the user should use the same track reading mode, such as the track of the card you want to use. If not, the printer cannot be guaranteed the hexadecimal reading data.

Finally, the text print using TTF is supported TTF file that is stored printer internal storage in advance. TTF for text printing is only supported the pre-stored TTF file in printer. So, the other TTF file cannot be guaranteed to use in the printer.

Appendix

Appendix A. Backward Compatibility

The library was created with the aim to new and convenient development method. Therefore, the deploying new libraries consist of most of new APIs, and provide various functions than the old library.

The SDK supports the backward compatibility politically. If you want to use additional functions by changing to the new library functions on the program that used old Woosim Windows's library, it can be used normally at the program that used old Woosim Windows's library when you add new libraries API information.

Now, most of the SDK's APIs in the previous of Woosim Windows SDK v4.0 is not officially supported. Therefore it is not recommended that a new project is made by using old version APIs prior to the libraries of SDK v4.0 Please use new library APIs since the version 4.0.

Appendix B. Convert data type between C/C++ and .Net language

The following is a matching table for data type conversion between Visual C++ and the language using .net framework such as Visual C# and Visual Basic. For more information, please refer to MSDN.

Unmanaged type in Wtypes.h	Unmanaged C language type	Managed class name	Description
BYTE	unsigned char	System::Byte	8 bits
SHORT	short	System::Int16	16 bits
WORD	unsigned short	System::UInt16	16 bits
INT	int	System::Int32	32 bits
UINT	unsigned int	System::UInt32	32 bits
LONG	long	System::Int32	32 bits
BOOL	long	System::Int32	32 bits
DWORD	unsigned long	System::UInt32	32 bits
CHAR	char	System::Char	Decorate with ANSI.
WCHAR	wchar_t	System::Char	Decorate with Unicode.
LPSTR	char*	System::String or System.Text::StringBuilder	Decorate with ANSI.
LPCSTR	Const char*	System::String or System.Text::StringBuilder	Decorate with ANSI.
LPWSTR	wchar_t*	System::String or System.Text::StringBuilder	Decorate with Unicode.

LPCWSTR	Const wchar_t*	System::String or System.Text::StringBuilder	Decorate with Unicode.
----------------	----------------	---	------------------------

Appendix C. Two-dimensional barcode table

The following tables are specific setup data about 2D barcode.

● DataMatrix size

Symbol size		Capacity (bytes)			*ECC(%)	Remark
Row	Column	Numeric	Alpha-numeric	Byte (8bit)		
10	10	6	3	3	62.5	
12	12	10	6	5	58.3	
8	18	10	6	5	58.3	Rectangular
14	14	16	9	8	55.6	
8	32	20	12	10	52.4	Rectangular
16	16	24	15	12	50.0	
12	26	32	21	16	46.7	Rectangular
18	18	36	24	18	43.8	
20	20	44	30	22	45.0	
12	36	44	30	22	45.0	Rectangular
22	22	60	24	30	40.0	
16	36	34	45	32	42.9	Rectangular
24	24	72	51	36	40.0	
26	26	88	63	44	38.9	
16	48	98	72	49	36.4	Rectangular
32	32	124	90	62	36.7	
36	36	172	126	86	32.8	
40	40	228	168	114	29.6	
44	44	288	213	144	28.0	
48	48	348	258	174	28.1	
52	52	408	303	204	29.2	
64	64	560	417	280	28.6	
72	72	736	549	368	28.1	
80	80	912	681	456	29.6	
88	88	1152	861	576	28.0	
96	96	1392	1041	696	28.1	

104	104	1632	1221	816	29.2	
120	120	2100	1572	1050	28.0	
132	132	2608	1953	1304	27.6	
144	144	3116	2334	1558	28.5	

* ECC (%): Error Correction Code rate (%).

● QR code size(version)

Version	Capacity (Codewords) by *ECC level			
	L (7%)	M (15%)	Q (25%)	H (30%)
1	19	16	13	9
2	34	28	22	16
3	55	44	34	26
4	80	64	48	36
5	108	86	62	46
6	136	108	76	60
7	156	124	88	66
8	194	154	110	86
9	232	182	132	100
10	274	216	154	122
11	324	254	180	140
12	370	290	206	158
13	428	334	244	180
14	461	365	261	197
15	523	415	295	223
16	589	453	325	253
17	647	507	367	283
18	721	563	397	313
19	795	627	445	341
20	861	669	485	385
21	932	714	512	406
22	1006	782	568	442
23	1094	860	614	464
24	1174	914	664	514
25	1276	1000	718	538
26	1370	1062	754	596

27	1468	1128	808	628
28	1531	1193	871	661
29	1631	1267	911	701
30	1735	1373	985	745
31	1843	1455	1033	793
32	1955	1541	1115	845
33	2071	1631	1171	901
34	2191	1725	1231	961
35	2306	1812	1286	986
36	2434	1914	1354	1054
37	2566	1992	1426	1096
38	2702	2102	1502	1142
39	2812	2216	1582	1222
40	2956	2334	1666	1276

*ECC level: Error correction code rate by level (%).

● Micro PDF417 size

Columns	Rows	Max Data Bytes	Max Alpha Characters	Max Digits
1	11	3	6	8
1	14	7	12	17
1	17	10	18	26
1	20	13	22	32
1	24	18	30	44
1	28	22	38	55
2	8	8	14	20
2	11	14	24	35
2	14	21	36	52
2	17	27	46	67
2	40	33	56	82
2	46	38	64	93
2	52	43	72	105
3	6	6	10	14
3	8	10	18	26
3	10	15	26	38
3	12	20	34	49

3	15	27	46	67
3	20	39	66	96
3	26	54	90	132
3	32	68	114	167
3	38	82	138	202
3	44	97	162	237
4	4	8	14	20
4	6	13	22	32
4	8	20	34	49
4	10	27	46	67
4	12	34	58	85
4	15	45	76	111
4	20	63	106	155
4	26	85	142	208
4	32	106	178	261
4	38	128	214	313
4	44	150	250	366