

# **Woosim embedded Windows SDK**

## **Programmer Reference**

---

Version 4.0

October 2015



## Contents

<b>1. OVERVIEW .....</b>	<b>3</b>
1.1. PURPOSE.....	3
1.2. GET STARTED .....	3
1.3. DEVELOPMENT ENVIRONMENT .....	4
1.4. DEFINITIONS AND ABBREVIATIONS .....	4
<b>2. SUMMARY .....</b>	<b>5</b>
2.1. INFORMATION.....	5
2.2. MESSAGE HANDLING .....	5
2.3. DEFINITION VALUES .....	5
2.4. WOOSIM EMBEDDED WINDOWS LIBRARY APIS .....	6
2.4.1. <i>Printable Data Handling APIs</i> .....	7
2.4.2. <i>Printer Mode &amp; Setting APIs</i> .....	9
2.4.3. <i>Miscellaneous Device Handling APIs</i> .....	10
<b>3. WOOSIM EMBEDDED WINDOWS APIS.....</b>	<b>12</b>
3.1. PRINTABLE DATA HANDLING APIS.....	12
3.2. PRINTER MODE & SETTING APIS .....	20
3.3. MISCELLANEOUS DEVICE HANDLING APIS .....	27
<b>4. SAMPLE CODES.....</b>	<b>31</b>
4.1. SAMPLE TEST APPLICATION.....	31
<b>APPENDIX.....</b>	<b>32</b>
APPENDIX A. BACKWARD COMPATIBILITY .....	32
APPENDIX B. CONVERT DATA TYPE BETWEEN C/C++ AND .NET LANGUAGE .....	32
APPENDIX C. TWO-DIMENSIONAL BARCODE TABLE.....	33

# 1. Overview

This release of the Woosim printer embedded Windows Software Development Kit (SDK) documentation provides information about embedded Windows OS based application development. Copyright © 2015 Woosim System Inc.

## 1.1. Purpose

본 문서는 *Woosim Command manual*의 16진수 명령어를 직접 사용하여 프린터를 제어하는 방식에서 벗어나 *Woosim Command manual*에서 제공하고 있는 기능들에 대해 사용이 편리하고 명확한 작성이 가능하도록 구성한 API들에 대해 소개하고 이것을 사용한 sample project에 대해 명시하고 있습니다.

## 1.2. Get Started

제공하는 SDK에는 본 문서와 라이브러리 파일들, 몇몇 sample project들이 포함되어 있습니다. 사용자의 개발 환경에 맞추어 2가지의 개발 환경의 라이브러리를 제공하고 있습니다. 사용자는 자신의 개발 환경에 맞추어 라이브러리를 사용하여야 합니다.

Woosim printer는 standard mode와 page mode라는 2가지 모드를 제공하고 있습니다. 사용자는 즉각적인 출력을 원한다면 standard mode를, 디자인된 형태의 결과를 원한다면 page mode를 사용하여야 합니다. Woosim embedded Windows SDK에서는 프린터 모드에 대한 API를 제공하고 있으며, 프린터뿐 아니라 라이브러리 내부에 buffer를 두어 데이터 출력을 제어하고 있습니다.

Woosim embedded Windows SDK는 MFC 라이브러리 API를 일부 포함한 DLL의 사용이 가능한 Visual C++ 그리고 Visual C#, Visual Basic에 대해 지원하고 있습니다.

### 1. Visual C++

Woosim embedded Windows SDK에서는 사용자가 DLL을 사용하는 것을 지원하기 위하여 DLL 파일과 linker input을 위한 lib 파일, APIs의 선언과 관련 정보가 포함된 header 파일을 제공합니다. 사용자는 사용 방법에 따라 lib 파일이나 header 파일을 사용하려는 프로젝트에 적합한 방식으로 추가하여 응용프로그램을 제작하여야 합니다.

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

일반적인 C/C++의 data type을 제공하지 않기 때문에, data type의 변환이 요구됩니다. SDK 내에 header 파일에 선언된 APIs와 관련 정보들을 .net 언어에 맞는 data type으로 변경하여 사용합니다. Data type converting과 관련된 matching table은 *Appendix B*를 참조 바랍니다.

세부적인 API의 선언과 data type 변경에 대한 example은 동봉된 sample project를 참조합니다.

### 1.3. Development Environment

Target platform	Windows CE5.0, CE6.0, Pocket PC 2003, Windows Mobile 5, 6
Target Language	eMbedded Visual C++, 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 embedded Windows SDK에서 제공하고 있는 sample project에는 source code뿐 아니라 실행파일도 같이 제공하고 있습니다. 사용자는 제공되는 실행파일을 통해서 라이브러리에서 제공하고 있는 다양한 기능들을 사용한 출력 결과를 확인 할 수 있습니다. 실행파일을 통해서 결과물을 확인하려면, 사용하려는 장비에 맞춘 라이브러리와 sample을 선택하여야만 정상적으로 결과를 확인할 수 있습니다.

본 SDK는 M16C과 ARM, RX의 MCU의 프린터를 지원하고 있습니다. 하지만 M37702 같은 오래된 MCU에 대해서는 제한적으로 제공됩니다. 대부분의 API는 RX MCU 프린터에 기반을 두고 있습니다. 소유하신 프린터의 MCU는 self-test를 통해 파악하실 수 있습니다.

### 2.2. Message Handling

모든 Woosim 프린터들은 몇 가지의 정보를 사용자의 단말기에 전달하도록 구성되어 있습니다. Windows에서 제공하는 다양한 메시지 사이에서 명확하게 제공되기 위하여, 다음과 같은 형식으로 메시지를 전달받습니다.

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

등록된 메시지를 통해 사용자의 단말기에 2개의 parameter가 포함된 메시지를 전달합니다. 첫 parameter에서는 데이터 타입과 전달하려는 데이터의 길이가 포함된 정보를 전달합니다. 전달받는 데이터 타입은 다음의 값 중 하나를 갖습니다.

STATUS_ANSWER	0x53
DATA	0x44

구분된 메시지의 데이터는 다른 parameter를 통해 사용자 단말기에 16진수 data로 전달됩니다. 이 data는 사용자가 프린터에 요청한 값에 따라서 서로 다른 길이를 갖습니다. 자세한 구성은 SDK에 첨부된 sample project를 참고하시길 바랍니다.

### 2.3. Definition values

본 라이브러리에서 사용하는 API들은 특정한 결과 값을 통해 상태를 나타내거나 혹은 사용의 편의성을 위해 몇 가지 값을 미리 선언하여 사용하고 있습니다. 다음은 미리 선언된 값들을 종류별로 구분한 것입니다.

#### ● 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

<Common Message>

SUCCESS 1  
 ALREADY\_OPENED -1  
 TIMEOUT -7  
 NOT\_OPEN\_THE\_PORT -11

#### ● Barcode Type

UPCA 65  
 UPCE 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 embedded Windows Library APIs

Woosim embedded Windows SDK에서 제공하는 API는 다음과 같이 구분이 가능합니다: Printable Data Handling, Printer mode & setting, and Miscellaneous Device Handling APIs.

Printable Data Handling 영역에서는 Woosim printer에서 출력 가능한 데이터와, 데이터를 프린터로 전달하는 API들로 구성되어 있습니다.

Printer mode & setting 영역에서는 프린터에서 제공하는 모드와 그 모드 관련 설정, 출력 가능한 데이터의 설정에 대한 부분으로 이루어져 있습니다.

마지막으로, Misc. Device Handling 영역에서는 프린터에 연결할 수 있는 장치들과 user terminal에 관련된 API들로 구성되어 있습니다.

본 라이브러리는 라이브러리 내 buffer를 통해 모아둔 API들의 데이터를 한번에 프린터로 전달하는 방식을 취하고 있습니다. 이 buffer는 본 문서에서 프린터의 buffer와 구분하기 위하여 printing buffer라는 명칭으로 사용되며, 대다수의 API들이 이 buffer를 거쳐서 프린터로 명령어를 전달합니다.

몇몇 API를 제외하고, 다수의 API에는 반환 값이 없으나 잘못된 API의 사용에 대해서는 원치 않는 출력 결과나 출력이 되지 않는 경우가 생길 수 있습니다. 따라서 사용에 앞서 API들에 대해서 명확히 숙지를 하는 것을 권장합니다.

### 2.4.1. Printable Data Handling APIs

void **ClearSpool**( )

Printing buffer의 내용을 지웁니다.

void **CompressedBmpSaveSpool**( TCHAR\* bmpFilePath )

압축알고리즘을 사용하는 사용자 정의 bit-image의 형태로 비트맵 데이터를 변환하고, printing buffer에 저장합니다.

void **ControlCommand**( BYTE\* Cmds, int Cmds\_length )

Printing buffer에 byte 배열로 된 데이터를 저장합니다.

void **DataMatrixSaveSpool**( int width, int height, int module, TCHAR\* barcodeData )

DataMatrix형태로 출력 가능한 2D 바코드 데이터를 만들고, printing buffer에 저장합니다.

void **GS1DatabarSaveSpool**( int type, int n, TCHAR\* barcodeData )

GS1 Databar 형태로 출력 가능한 바코드 데이터를 만들고, printing buffer에 저장합니다.

void **LoadLogoSaveSpool**( int n )

미리 프린터에 다운로드 된 이미지를 불러옵니다.

void **MaxicodeSaveSpool**( int mode, TCHAR\* barcodeData )

Maxicode 형태로 출력 가능한 2D 바코드 데이터를 만들고, printing buffer에 저장합니다.

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

Micro PDF417로 출력 가능한 2D 바코드 데이터를 만들고, printing buffer에 저장합니다.

void **NormalBmpSaveSpool**( TCHAR\* bmpFilePath )

사용자 정의 bit-image의 형태로 비트맵 데이터를 변환하고, printing buffer에 저장합니다.

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

ucBarcodeType 타입의 1D 바코드 형태로 출력 가능한 데이터를 만들고, printing buffer에 저장합니다.

- void **Page\_DotFeed**( int dots )  
출력 방향으로 n dot 만큼 이동하여 새로운 라인을 만들고 시작 지점에 출력 위치를 이동합니다.
- void **Page\_DrawBox**( int iXPos, int iYPos, int width, int height, int thickness )  
출력 가능한 직선 또는 사각형 이미지 데이터를 작성하고, printing buffer에 저장합니다.
- void **Page\_DrawEllipse**( int iXPos, int iYPos, int A\_length, int B\_length, int thickness )  
출력 가능한 타원형 이미지 데이터를 작성하고, printing buffer에 저장합니다.
- void **Page\_DrawLine**( int iXPos, int iYPos, int iX2Pos, int iY2Pos, int thickness )  
출력 가능한 두 점을 잇는 직선 데이터를 작성하고, printing buffer에 저장합니다.
- void **Page\_LineFeed**( int lines )  
현재 위치에서 lines 만큼 이동하고, 그 라인의 시작 지점에 쓰기 위치를 이동시킵니다.
- void **Page\_Newline**( )  
현재 위치에서 다음 라인의 시작지점으로 출력 위치를 이동합니다.
- void **Page\_Print**( )  
Page mode에서 데이터를 출력시킵니다.
- void **Page\_Print\_StandardMode**( )  
Page mode에서 데이터를 출력시키고 standard mode로 전환합니다.
- void **PDF417SaveSpool**( int width, int column, int level, int ratio, TCHAR\* barcodeData, BOOL HRI )  
PDF417type 타입의 2D 바코드 형태로 출력 가능한 데이터를 만들고, printing buffer에 저장합니다.
- void **PrintData**( )  
데이터를 출력시키고 다음 줄로 출력 시작 점을 이동시킵니다.
- void **PrintDotFeed**( int dots )  
데이터를 출력시키고 dots 만큼 이동한 지점으로 출력 시작점을 이동시킵니다
- void **PrintLineFeed**( int lines )  
데이터를 출력시키고 lines 만큼 이동한 지점으로 출력 시작점을 이동시킵니다
- int **PrintSpool**( BOOL bDelete\_Spool )  
Printing buffer에 저장된 데이터를 Woosim 프린터로 전달합니다.
- void **PrintSpoolForTTF**( TCHAR\* sData, BYTE iXFontSize, BYTE, iYFontSize )  
True type font를 사용하여 텍스트 데이터를 출력시킵니다.
- void **QRCodeSaveSpool**( int version, char level, int module, TCHAR\* barcodeData )  
QR code 형태로 출력 가능한 2D 바코드 데이터를 만들고, printing buffer에 저장합니다.



- void **TextSaveSpool**( TCHAR\* textData )  
 텍스트 데이터를 printing buffer에 저장합니다.
- void **TruncatedPDF417SaveSpool**( int width, int column, int level, int ratio,  
 TCHAR\* barcodeData, BOOL HRI )  
 Truncated PDF417 타입의 2D 바코드 형태로 출력 가능한 데이터를 만들고, printing buffer  
 에 저장합니다.

## 2.4.2. Printer Mode & Setting APIs

- void **GetPrinterModelName**( )  
 프린터로부터 모델명과 MCU에 대한 정보를 가져옵니다.
- void **GetFirmwareVersion**( )  
 프린터로부터 펌웨어 버전과 작성 날짜를 가져옵니다.
- int **GetPrinterStatus**( int iTimeoutMsec )  
 프린터로부터 현재 상태 정보를 가져옵니다.
- void **InitLineSpace**( )  
 한 줄에 할당되는 공간을 기본 값으로 변경합니다.
- void **InitPageMode**( int iXPos, int iYPos, int width, int height )  
 프린터 상태 초기화 그리고 page mode 전환, 현재 page mode의 데이터 삭제, 출력 가능  
 범위를 설정합니다.
- void **InitPrinterStatus**( )  
 프린터의 버퍼와 사용자가 설정한 설정에 대해 초기화 시킵니다.
- void **Page\_ClearCurrentData**( )  
 현재 모든 page mode 데이터를 삭제합니다.
- void **Page\_SetArea**( int iXPos, int iYPos, int width, int height )  
 Page mode의 출력 가능 범위를 설정합니다.
- void **Page\_SetDirection**( int n )  
 Page mode에서 출력하는 용지의 방향을 설정합니다.
- void **Page\_SetPosition**( int iXPos, int iYPos )  
 Page mode에서 출력하려는 데이터의 위치를 설정합니다.
- void **Page\_SetStandardMode**( )  
 Page mode에서 standard mode로 전환합니다.
- void **SetAbsPosition**( int distance )  
 문자의 시작지점을 이동시킵니다.

- void **SetCharCodeTable**( int n, int MCU )  
현재 사용하고 있는 문자 코드 테이블을 설정합니다.
- void **SetCharSpace**( int n )  
문자의 오른쪽 공백을 설정합니다.
- void **SetFontForTTF**( TCHAR\* ttffile )  
사용할 true type font파일을 선택합니다.
- void **SetFontSize**( int n )  
프린터의 기본 폰트 크기를 선택합니다.
- void **SetLineSpace**( int n )  
문자의 한 줄에 할당되는 공간을 변경합니다.
- void **SetPageMode**( )  
Standard mode에서 page mode로 전환합니다.
- void **SetTextAlignment**( int n )  
문자 출력 위치를 오른쪽 그리고 왼쪽, 가운데 중에서 선택합니다.
- void **SetTextStyle**( int underline, BOOL emphasize, int width, int height, BOOL reverse )  
출력하는 텍스트의 모양을 선택합니다.
- void **SetUpsideDown**( BOOL set )  
텍스트의 upside down방식 출력을 설정합니다.

### 2.4.3. Miscellaneous Device Handling APIs

- void **CancelMSRMode**( )  
마그네틱 카드 읽기 모드를 취소합니다.
- void **CancelSCRMode**( )  
스마트 카드 읽기 모드를 취소합니다.
- BOOL **ClosePrinterConnection**( )  
사용자 단말기와 프린터간의 연결을 해지합니다.
- int **ConnectSerialPrinter**( TCHAR\* sPortName, int iBaudRate, int iTimeoutMsec, BOOL bProtocol )  
사용자 단말에서 COM포트를 통한 serial 통신을 사용하여 Woosim 프린터와 연결합니다.
- int **ConnectWirelessPrinter**( TCHAR\* sIP\_ADDR, int iPortNum, int iTimeoutMsec, BOOL bProtocol )  
사용자 단말기에서 IP 통신을 사용하여 Woosim 프린터와 연결합니다.
- void **CutPaper**( int mode )

Cut 모드를 선택하고 용지를 자릅니다.

void **EnterMSRMode**( int n )

설정된 track mode로 마그네틱 카드 읽기 모드를 실행합니다.

void **EnterSCRMode**( )

스마트 카드 읽기 모드를 실행합니다.

void **FeedToMark**( )

Black mark가 나타날 때까지 프린터의 용지를 출력합니다.

void **SetPositionFromMark**( int distance )

용지의 black mark에서부터 이동할 거리를 설정합니다.

## 3. Woosim embedded Windows APIs

Woosim embedded Windows SDK에서 제공하는 API들은 특정 함수를 제외하고 모든 데이터를 라이브러리 내의 printing buffer라는 buffer내에 저장합니다. 이후 함수 PrintSpool()이나 ClearSpool()에 의해 printing buffer의 내용은 프린터로 전달되거나, 혹은 지워집니다.

다음은 printing buffer에 데이터를 저장하지 않는 API 목록입니다.

CancelMSRMode	CancelSCRMode
ConnectSerialPrinter	ConnectUSBPrinter
ConnectWirelessPrinter	EnterMSRMode
EnterSCRMode	GetPrinterModelName
GetPrinterStatus	GetFirmwareVersion

이 함수들 중에서 사용자 터미널과 프린터간의 연결과 관련된 함수들을 제외하고, 나머지 API들은 호출을 하게 되면 즉각적으로 프린터에서 동작을 수행하며, 동작의 결과를 사용자 단말로 전달합니다.

### 3.1. Printable Data Handling APIs

void **ClearSpool()**

사용자는 라이브러리의 printing buffer의 내용을 지우기 위해서 이 함수를 사용하여야 합니다. 이 함수는 프린터 내의 버퍼 데이터를 지우지는 않습니다.

void **CompressedBmpSaveSpool**( TCHAR\* bmpFilePath )

지정한 비트맵 파일의 데이터를 압축 알고리즘을 사용하는 사용자 정의 bit image 형태로 변환시키고, printing buffer에 저장합니다. 그 외의 동작은 NormalBmpSaveSpool함수와 동일하지만, standard mode에서 더 빠른 동작을 보여줍니다.

Parameter

*bmpFilePath*                      출력시킬 bmp 파일이 포함된 경로.

void **ControlCommand**( BYTE\* Cmds, int Cmds\_length )

이 함수는 입력 받은 byte stream을 printing buffer로 전달합니다. 만일 사용자가 이 함수를 사용하고자 한다면, *Woosim Command manual*을 참조해야 합니다.

Parameter

*Cmds*                                  명령어로 구성된 byte stream

*Cmds\_length*                      Byte stream의 길이

void **DataMatrixSaveSpool**( int width, int height, int module, TCHAR\* barcodeData)

DataMatrix형태로 출력 가능한 2D 바코드 데이터를 만들고, printing buffer에 저장합니다. 바코드의 크기나 타입 별 가능한 데이터 범위에 대해서는 *Appendix*를 참조 바랍니다.

Parameter

<i>width</i>	DataMatrix의 바코드 폭 (0: auto size)
<i>height</i>	DataMatrix의 바코드 높이 (0: auto size)
<i>module</i>	DataMatrix의 모듈 크기 (1~8)
<i>barcodeData</i>	DataMatrix에 사용될 데이터

void **GS1DatabarSaveSpool**( int type, int n, TCHAR\* barcodeData )

GS1 Databar 형태로 출력 가능한 2D 바코드 데이터를 만들고, printing buffer에 저장합니다. GS1 Databar는 2012년 12월 이후 release 된 firmware를 사용하는 RX MCU의 프린터에서만 사용 가능합니다.

Parameter

<i>type</i>	GS1 Databar 타입 (0~6): GS1 Databar Omnidirectional, Truncated, Stacked, Stacked Omnidirectional, Limited, Expanded, and Expanded Stacked.
<i>n</i>	Segments per row(2~20). 이 값은 오직 타입이 6인 경우에만 사용되며, 짝수 값만 사용 가능합니다.
<i>barcodeData</i>	GS1 Databar에 사용될 데이터 GS 1 Databar의 타입이 0~4로 선택되면, 데이터의 최대 길이는 14로 제약 됩니다. 타입 5와 6인 경우에는 GS1 표준 문서를 준수하며, '('와 ')'를 표기하기 위해 '['과 ']'를 사용합니다.

void **LoadLogoSaveSpool**( int n )

미리 프린터에 다운로드 된 이미지를 불러옵니다.

사용하기 앞서, 사용자는 프린터에 이미지를 업로드 해야 합니다. 만일 업로드 되지 않거나 잘못된 이미지 번호를 선택한다면 이 함수는 무시됩니다.

Parameter

<i>n</i>	사용할 이미지의 번호. 이 값은 프린터의 MCU에 따라 최대 범위가 다릅니다.
----------	--

void **MaxicodeSaveSpool**( int mode, TCHAR\* barcodeData )

Maxicode 형태로 출력 가능한 2D 바코드 데이터를 만들고, printing buffer에 저장합니다. Maxicode는 오직 RX MCU의 프린터에서만 제공되는 2D 바코드입니다. 사용에 대한 자세한 내용은 *Woosim Command manual*을 참조바랍니다.

Parameter

<i>mode</i>	Maxicode의 모드 (2~6)
<i>barcodeData</i>	Maxicode의 2D 바코드 데이터.

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

Micro PDF417 형태로 출력 가능한 2D 바코드 데이터를 만들고, printing buffer에 저장합니다. 바코드의 행과 열에 관련하여 표현 가능한 데이터 양에 대해서는 *Appendix*를 참조 바랍니다.

Parameter

<i>width</i>	바코드 폭의 값 (1 ~ 8) 바코드 폭의 값이 범위를 초과하게 되면 이 명령은 무시되고 바코드는 출력되지 않습니다.
<i>column</i>	Micro PDF417 바코드의 행의 개수 (1~4).
<i>row</i>	Micro PDF417 바코드의 열의 개수 (4~44, 0: auto size).
<i>ratio</i>	The horizontal and vertical ratio (2~5)
<i>Data</i>	Micro PDF417 2D 바코드 데이터.
<i>HRI</i>	HRI 문자 출력 옵션 이 값이 선언되면 바코드 하단에 입력 데이터가 출력됩니다.

void **NormalBmpSaveSpool**( TCHAR\* bmpFilePath)

지정된 비트맵 파일의 데이터를 사용자 정의 bit image 형태로 변환시키고, 라이브러리의 printing buffer에 저장합니다. 이 함수는 page mode에서 동작하는 것을 권장합니다. Page 모드의 특성상, 이미지의 손실을 막을 수 있으며 동시에 원하는 위치에 이동이 가능합니다.

Parameter

<i>bmpFilePath</i>	출력시킬 bmp 파일이 포함된 경로.
--------------------	----------------------

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

ucBarcodeType의 1D 바코드의 형태로 출력이 가능한 데이터를 만들고, printing buffer에 저장합니다.

Parameter

<i>ucBarcodeType</i>	1D 바코드의 타입. (65~73) 사용할 수 있는 바코드 타입은 아래의 테이블을 참조하십시오
----------------------	---

<i>width</i>	바코드의 폭 (2 ~ 8) 바코드 폭의 값이 범위를 초과하게 되면 이 명령은 무시되고 바코드는 출력되지 않습니다.
<i>height</i>	Dot 단위의 바코드 높이 (0~255)
<i>HRI</i>	HRI 문자 출력 옵션. 이 값이 설정되면 바코드 하단에 입력 데이터가 출력됩니다.
<i>barcodeData</i>	1D 바코드의 데이터 데이터의 길이나 값의 범위는 각 바코드 타입에 의해 결정됩니다. 자세한 내용은 아래의 테이블을 참고하기 바랍니다.

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)

Page 모드에서, 출력 방향으로 dots 만큼 이동하여 새로운 라인을 만들고 시작 지점에 출력 위치를 이동합니다.

Parameter

*dots* Dot 단위의 용지 출력 길이(0~255)

void **Page\_DrawBox**( int iXPos, int iYPos, int width, int height, int thickness )

출력 가능한 직선 또는 사각형 이미지 데이터를 작성하고, printing buffer에 저장합니다. 이 함수는 page 모드에서만 동작합니다. 또한 사용자는 height나 width 값을 0으로 함으로써 수평선이나 수직선을 그릴 수 있습니다.

Parameter

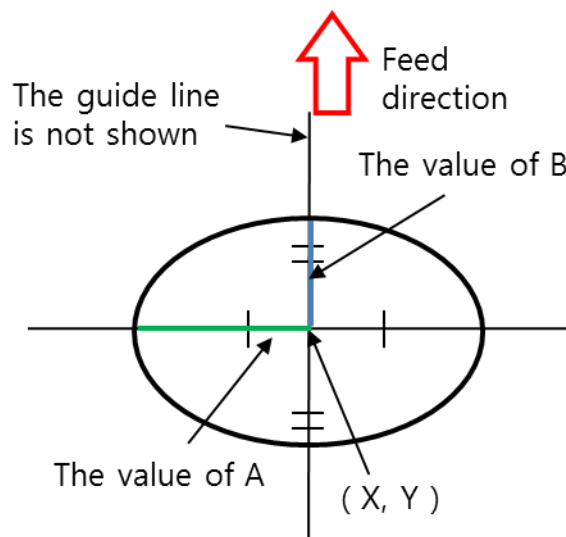
<i>iXPos</i>	Dot 단위의 원점의 X좌표
<i>iYPos</i>	Dot 단위의 원점의 Y좌표
<i>width</i>	그려질 사각형 이미지의 폭.
<i>height</i>	그려질 사각형 이미지의 높이
<i>thickness</i>	그려질 사각형 테두리의 두께

void **Page\_DrawEllipse**( int iXPos, int iYPos, int A\_length, int B\_length, int thickness )

출력 가능한 타원형 이미지 데이터를 작성하고, printing buffer에 저장합니다. 이 함수는 page 모드에서만 동작을 하며, 세로 길이 A와 가로 길이 B의 조정으로 다양한 형태와 크기의 원을 그릴 수 있습니다. 자세한 값의 설정은 다음의 이미지를 참조 바랍니다.

Parameter

<i>iXPos</i>	Dot 단위의 원점의 X좌표
<i>iYPos</i>	Dot 단위의 원점의 Y좌표
<i>A_length</i>	그려질 타원의 x 축의 절반 길이.
<i>B_length</i>	그려질 타원의 y 축의 절반 길이
<i>thickness</i>	그려질 타원 테두리의 두께



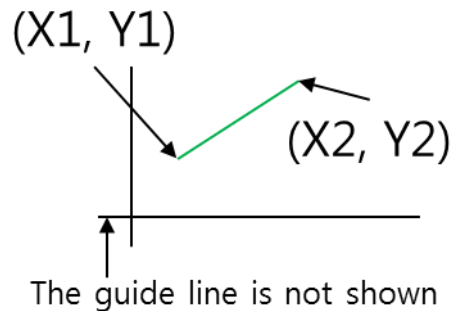


void **Page\_DrawLine**( int iXPos, int iYPos, int iX2Pos, int iY2Pos, int thickness )

출력 가능한 두 점을 잇는 직선 데이터를 작성하고, printing buffer에 저장합니다. 이 함수는 page 모드에서만 동작을 하며, 주어진 점 (X1, Y1)과 (X2, Y2)를 이어주는 선을 그립니다.

Parameter

<i>iXPos</i>	Dot 단위의 직선의 X1좌표
<i>iYPos</i>	Dot 단위의 직선의 Y1좌표
<i>iX2Pos</i>	Dot 단위의 직선의 X2좌표
<i>iY2Pos</i>	Dot 단위의 직선의 Y2좌표
<i>thickness</i>	그려질 직선의 두께



void **Page\_LineFeed**( int lines )

Page 모드에서, lines 만큼 이동하고, 그 라인의 시작 지점에 출력 위치를 이동시킵니다. 출력되는 길이는 SetFontSize()와 SetLineSpace()에 영향을 받습니다.

Parameter

<i>lines</i>	Line 단위의 용지 출력 길이 (0~255)
--------------	---------------------------

void **Page\_Newline**( )

Page 모드에서, 현재 line spacing에 맞추어 다음 라인의 시작지점으로 입력 위치를 이동시킵니다. 출력되는 길이는 SetFontSize()와 SetLineSpace()에 영향을 받습니다.

void **Page\_Print**( )

Page 모드에서 현재 페이지 영역의 데이터를 출력한다.

만일 사용자가 페이지 영역을 설정하지 않았다면, 기본 page 영역 값으로 출력 됩니다.

기본 page 영역의 세로 길이는 2400dot (30cm)입니다.

void **Page\_Print\_StandardMode**( )

Page mode에서, 설정된 현재 페이지 영역의 데이터 출력하고 standard mode로 복귀합니다. 만일 사용자가 페이지 영역을 설정하지 않았다면, 기본 page 영역 값으로 출력 됩니다.

기본 page 영역의 세로 길이는 2400dot (30cm)입니다.

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

PDF417 타입의 2D 바코드 형태로 출력 가능한 데이터를 만들고, printing buffer에 저장합니다.

Parameter

<i>width</i>	바코드 폭의 값 (1 ~ 8) 바코드 폭의 값이 범위를 초과하게 되면 이 명령은 무시되고 바코드는 출력되지 않습니다.
<i>column</i>	PDF417 바코드의 행의 개수 (1~30).
<i>level</i>	바코드가 손상될 경우에 복구될 수 있는 복구 레벨(0~8).
<i>ratio</i>	The horizontal and vertical ration (2~5)
<i>Data</i>	PDF417 2D 바코드 데이터.
<i>HRI</i>	HRI 문자 출력 옵션 이 값이 선언되면 바코드 하단에 입력 데이터가 출력됩니다.

void **PrintData**( )

Standard 모드에서 데이터를 출력하고 현재 line spacing에 맞추어 한 라인만큼 용지를 출력한다. 출력되는 길이는 SetFontSize()와 SetLineSpace()에 영향을 받는다.

void **PrintDotFeed**( int dots )

Standard 모드에서 데이터를 출력하고 dots 만큼 용지를 출력한다.

Parameter

<i>dots</i>	Dot 단위의 용지 출력 길이 (0~255)
-------------	--------------------------

void **PrintLineFeed**( int lines )

Standard 모드에서 데이터를 출력하고 lines 만큼 용지를 출력한다. 출력되는 길이는 SetFontSize()와 SetLineSpace()에 영향을 받는다.

Parameter

<i>lines</i>	Line 단위의 용지 출력 길이 (0~255)
--------------	---------------------------

int **PrintSpool**( BOOL bDelete\_Spool )

Printing buffer에 저장된 데이터를 Woosim 프린터로 전달합니다.

이 함수를 사용하면 printing buffer의 내용을 프린터로 전달하게 되며, 프린터는 전달받는 명령에 따라 데이터를 처리합니다. 또한 parameter를 통해 데이터 전달 후, printing buffer의 내용을 지울 것인 가를 결정합니다.

Parameter

<i>bDelete_Spool</i>	Printing buffer의 초기화 변수. 이 값이 설정되면 프린터로 데이터
----------------------	---

를 전달한 이후, printing buffer의 내용을 초기화 합니다.

Return value

SUCCESS	데이터 전송 성공
NOT_OPEN_THE_PORT	프린터가 연결 되지 않음

void **PrintSpoolForTTF**( TCHAR\* sData, BYTE iXFontSize, BYTE, iYFontSize )

SetFontForTTF()를 통해 설정된 TTF 파일을 사용하여 입력된 텍스트 데이터를 출력합니다. 이때 출력되는 폰트의 크기는 입력 받는 폰트 사이즈를 사용합니다.

Parameter

<i>sData</i>	출력하려고 하는 데이터
<i>iXFontSize</i>	출력하는 폰트의 width.
<i>iYFontSize</i>	출력하는 폰트의 height.

void **QRCodeSaveSpool**( int version, char level, int module, TCHAR\* barcodeData )

QR code 형태로 출력 가능한 2D 바코드 데이터를 만들고, printing buffer에 저장합니다. 4가지 level에 대해서 version별 가용 데이터 양이 서로 다르며, level에 따라 ECC-level 역시 다릅니다. *Appendix* 영역의 *QR code table*을 참조 바랍니다.

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>	QR code로 만들 입력 데이터 Data 이 데이터의 길이와 값의 범위는 version 값과 level 값에 의해 결정됩니다.

void **TextSaveSpool**( TCHAR\* textData)

출력하고자 하는 텍스트를 라이브러리 내의 printing buffer에 저장합니다.

Parameter

<i>textData</i>	Printing buffer에 출력할 텍스트 데이터를 저장합니다.
-----------------	--------------------------------------

void **TruncatedPDF417SaveSpool**( int width, int column, int level, int ratio, TCHAR\* barcodeData, BOOL HRI )

Truncated PDF417 타입의 2D 바코드 형태로 출력 가능한 데이터를 만들고, printing buffer에 저장합니다.

Parameter

<i>width</i>	바코드 폭의 값 (1 ~ 8) 바코드 폭의 값이 범위를 초과하게 되면 이 명령은 무시되고 바코드는 출력되지 않습니다.
<i>column</i>	Truncated PDF417 바코드의 행의 개수 (1~30).
<i>level</i>	바코드가 손상될 경우에 복구될 수 있는 복구 레벨(0~8).
<i>ratio</i>	The horizontal and vertical ration (2~5)
<i>Data</i>	Truncated PDF417 2D 바코드 데이터.
<i>HRI</i>	HRI 문자 출력 옵션 이 값이 선언되면 바코드 하단에 입력 데이터가 출력됩니다.

## 3.2. Printer Mode & Setting APIs

### void **GetPrinterModelName( )**

프린터로부터 모델명과 MCU에 대한 정보를 가져옵니다.

이 함수를 실행하게 되면 프린터로부터 사용자 단말기에 16진수 형식의 data로 응답 결과를 전달합니다. 전달된 16진수 data는 프린터의 self test에서와 같은 결과를 사용자 단말기에 전달합니다.

### void **GetFirmwareVersion( )**

프린터로부터 펌웨어 버전과 작성 날짜를 가져옵니다.

이 함수를 실행하게 되면 프린터로부터 사용자 단말기에 16진수 형식의 data로 응답 결과를 전달합니다. 전달된 16진수 data는 프린터의 self-test에서와 같은 결과를 사용자 단말기에 전달합니다.

### int **GetPrinterStatus( int iTimeoutMsec )**

프린터로부터 현재 상태 정보를 가져옵니다.

이 함수를 선언하게 되면 프린터로부터 사용자 단말기에 16진수 data로 응답 결과를 전달합니다. 전달된 16진수 상태 값 data는 printer의 센서 값을 bit 부호로 표기하고 있습니다. 추가적인 정보에 대해서는 *Woosim Command manual*을 참조하기 바랍니다

Parameter

iTimeoutMsec	데이터 요청의 유효시간. 이 시간을 넘기게 되면 요청 실패로 간주.
--------------	---------------------------------------

Return value

SUCCESS	데이터 전송 성공
TIMEOUT	유효시간 내에 데이터 처리 실패

**void InitLineSpace( )**

기본 문자열의 공간으로 현재 문자열 공간을 기본 값 30dot로 설정합니다.

이 값은 프린터의 mode에 관계없이 동작하지만, 프린터의 현재 사용 중인 폰트 크기에 영향을 받습니다.

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

프린터 상태 초기화 그리고 page mode 전환, 이전 page mode의 데이터 삭제, 출력 가능 범위를 설정합니다. 사용자는 손쉽게 page 모드로 설정 할 수 있습니다.

이 함수는 다음의 함수들로 대체가 가능합니다: InitPrinterStatus(), SetPageMode(), Page\_ClearCurrentData(), Page\_SetArea().

Parameter

<i>iXPos</i>	출력을 위한 x축 시작 위치. 이 값은 dot단위로 제공됩니다.
<i>iYPos</i>	출력을 위한 y축 시작 위치. 이 값은 dot단위로 제공됩니다.
<i>width</i>	Dot 단위의 출력 영역의 width 값. 이 값은 0이상이어야 합니다 수평 값의 최대 길이는 target 프린터나 용지에 의해 결정됩니다.
<i>height</i>	Dot 단위의 출력 영역의 height값. 이 값은 0이상이어야 합니다. 최대 수직 길이는 2400 dot 입니다.

**void InitPrinterStatus( )**

현재 프린터의 출력 설정과 데이터를 초기화 시킵니다. SetTextAlignment()나 SetCharSpace()와 같은 API를 통한 설정과 프린터에서 출력하려는 데이터에 대해서 초기화가 실행됩니다. 하지만 printing buffer의 내용을 clear하지는 않습니다. 또한 프린터에서 직접 설정한 기기적인 설정이나 macro 설정에 대해서는 초기화가 진행되지 않습니다.

**void Page\_ClearPageData( )**

사용자는 이 함수를 사용하여 page mode에서 현재 작업한 내용이 담긴 printer buffer의 데이터를 지울 수 있습니다. 하지만 라이브러리의 printing buffer의 data를 지우지 않습니다. 이 함수는 오직 page mode에서만 동작합니다.

**void Page\_SetArea( int iXPos, int iYPos, int width, int height )**

Page mode의 출력 가능 범위를 설정합니다.

출력가능 범위의 원점이 출력 가능 범위를 넘어가면 이 함수는 무시됩니다. 또한 설정한 폭 또는 높이의 값이 용지보다 크면, 이 값은 기본 값으로 변경됩니다.

Parameter

<i>iXPos</i>	출력을 위한 x축 시작 위치. 이 값은 dot단위로 제공됩니다.
<i>iYPos</i>	출력을 위한 y축 시작 위치. 이 값은 dot단위로 제공됩니다.
<i>width</i>	Dot 단위의 출력 영역의 width 값. 이 값은 0이상이어야 합니다

수평 값의 최대 길이는 target 프린터나 용지에 의해 결정됩니다.

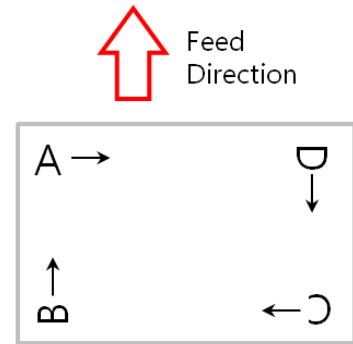
*height*

Dot 단위의 출력 영역의 height값. 이 값은 0이상이어야 합니다.  
최대 수직 길이는 2400 dot 입니다.

void **Page\_SetDirection**( int n )

Page mode에서 Page\_SetArea함수에 의해 설정된 출력 영역에서의 출력 위치를 설정합니다. 이 값은 반 시계방향으로 회전합니다.

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



Parameter

*n* 출력방향과 출력 방향의 설정 값 (0~3)

void **Page\_SetPosition**( int iXPos, int iYPos )

Page mode에서 데이터의 위치를 설정합니다.

Parameter

*iXPos* 출력을 위한 x축 시작 위치. 이 값은 dot단위로 제공됩니다.

*iYPos* 출력을 위한 y축 시작 위치. 이 값은 dot단위로 제공됩니다.

void **Page\_SetStandardMode**( )

Page mode에서 standard mode로 전환합니다.

Standard mode는 프린터의 default모드입니다. 입력 받은 데이터에 대해 라인 단위 혹은 dot단위로 즉시 출력합니다. Page 모드에 비해 더 빠른 출력 속도를 갖습니다.

이 함수는 page 모드일 때만 동작합니다.

void **SetAbsPosition**( int distance )

출력되는 데이터의 시작지점을 이동시킵니다. 이 이동은 한 라인에서 가로로 얼마만큼 떨어진 지점에서 데이터 출력이 시작될 것인가를 설정합니다. 만일 출력 가능한 범위를 초과한다면, 이 명령은 무시됩니다.

Parameter

*distance* 시작 위치로부터 이동할 거리

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

출력할 문자의 code table을 선택합니다. 이것은 프린터의 MCU와 밀접한 관계를 갖습니다. M16C나 ARM MCU를 사용하는 프린터는 오직 7개의 code table을 사용할 수 있습니다. 반면에 RX MCU를 사용하는 프린터는 이보다 더 많은 code table을 사용할 수 있도록 지원하고 있습니다. 자세한 table 정보는 아래의 표를 참조바랍니다.

Parameter

*n* 사용할 code table의 값

*MCU* 프린터의 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)

※ DBCS는 Font-A만 제공됩니다.

● 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)

※ 한글 code table은 Font-A (24x24) and Font-B (16x24)가 제공됩니다.

다른 DBCS들은 Font-A(24x24)만 제공됩니다.

void **SetCharSpace**( int n )

문자들의 오른쪽 공백의 크기를 설정합니다. 이 값은 문자간 간격 조절에 사용되며, 프린터의 mode에 영향을 받지 않습니다.

Parameter



*n* 문자 공백의 값. Dot 단위로 설정이 가능합니다. (0~255)

**void SetFontForTTF( TCHAR\* ttfFile )**

프린터 기본 폰트가 아닌, Woosim에서 추가 제공되는 TTF폰트를 사용하려고 할 때 사용됩니다. 사용하기 앞서, 프린터 내에 TTF파일이 저장되어 있어야 하며, 사용하려는 TTF파일 이름을 parameter로 입력해야 합니다.

Parameter

*ttfFile* 사용하려는 프린터 내의 TTF file name.

**void SetFontSize( int n )**

프린터의 기본 폰트 크기를 선택합니다.

Woosim printer에서는 출력되는 문자 폰트의 크기를 조절 할 수 있습니다. MCU가 M16C 나 ARM인 프린터에서는 2가지 폰트를 제공하며, RX MCU의 프린터에서는 3가지의 폰트 크기를 제공하고 있습니다. 폰트의 크기는 사용하고 있는 프린터의 character code table에 종속적이며, 경우에 따라서는 제공되는 폰트 크기가 제한될 수 있습니다. Character code table의 page 0번을 기준으로 폰트 크기는 다음과 같습니다.

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

DBCS와 같은 특별한 character형태에서는 지원하는 font 개수와 font size가 다른 table과 다름으로 사용에 주의 바랍니다. 자세한 table 별 크기 지원은 SetCharCodeTable()을 참조 바랍니다.

이 함수를 사용한다면, 프린터는 font-style, line spacing등의 character관련 설정들을 자동으로 초기화 하기 때문에 주의를 요합니다.

Parameter

*n* 출력에 사용할 폰트 크기(0~2): Font-A, Font-B, and Font-C

**void SetLineSpace( int n )**

하나의 라인에 대한 세로 영역을 설정합니다. 이 값은 텍스트 데이터의 출력에 많은 영향을 미칩니다. 이 값이 기본 값보다 작으면 출력하는 텍스트 데이터에 손실이 나올 수 있습니다. 이 값은 프린터의 mode에 영향을 받지 않습니다.

Parameter

*n* Dot단위의 라인의 공간 크기 (0~255)

**void SetPageMode( )**

Standard mode에서 page mode로 전환합니다.

Page 모드는 출력 가능한 데이터를 프린터 내부 버퍼에 저장해 두었다가 한꺼번에 출력하는 모드입니다. 또한 출력 위치를 지정할 수 있으며 standard 모드에 비해 다양한 형태의 데이터를 만들 수 있습니다.

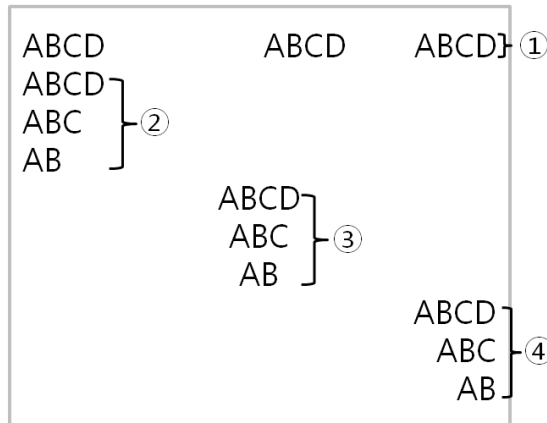
이 함수는 standard 모드일 때만 동작합니다.

void **SetTextAlignment**( int n )

출력하려는 텍스트의 정렬 위치를 정합니다. 이 함수는 실행한 위치부터 다른 위치로 변경할 때까지 지속됩니다.

자세한 동작은 아래 이미지를 참조하십시오.

이미지에서 ②, ③, ④은 각각 왼쪽 그리고 가운데, 오른쪽 정렬을 보여줍니다. ①의 경우에



는 한 줄에서 왼쪽, 가운데, 오른쪽 정렬을 모두 수행한 결과를 보여줍니다. 가운데 정렬인 경우에서, ①의 가운데 정렬인 부분과 ③의 가운데 정렬인 경우는 명령어가 인식하는 영역의 차이로 인해 생기는 부분입니다. 사용자는 이러한 특성에 유의하여야 합니다.

Parameter

*n* 정렬 번호(0~2): 왼쪽, 가운데, 오른쪽

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

텍스트의 밑줄과 강조, 반전, 크기 확장을 포함하는 텍스트 모양을 설정합니다.

Parameter

*underline* 밑줄 설정 값(0~2): 0: 없음, 1: 1dot 밑줄, 2: 2dot 밑줄.

*emphasize* 이 값을 설정 시, 문자가 강조됩니다.

*width* 수평방향으로 문자 크기 확대.  
자세한 설정은 아래 테이블을 참조하십시오.

*height* 수직방향으로 문자 크기 확대.  
자세한 설정은 아래 테이블을 참조하십시오

*reverse* 반전 모드를 설정합니다. 이 값이 설정되면 흑백 반전이 됩니다.

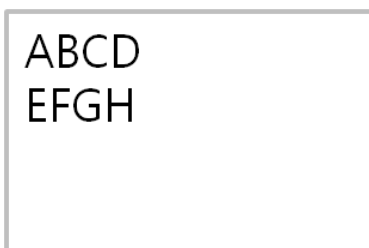
n	Width	Height
0	1 (Normal)	1 (Normal)

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 )

Upside down을 설정합니다. 이 함수는 오직 standard mode에서만 동작합니다.

Upside-Down off



Feed  
Direction

Upside-Down on



Parameter

*set*                      The flag of upside-down.

### 3.3. Miscellaneous Device Handling APIs

void **CancelMSRMode**( )

MSR 모드를 취소합니다.

이 함수는 어떠한 카드 읽기 모드에서도 동작을 수행합니다.

void **CancelSCRMode**( )

SCR 모드를 취소합니다.

BOOL **ClosePrinterConnection**( )

사용자 단말기와 프린터간의 연결을 해지합니다.

이 함수는 어떠한 연결 모드에서도 연결을 해지하며, 할당된 라이브러리의 버퍼 역시 해지가 됩니다. 따라서 이 함수를 사용하여 연결을 해지할 경우에는, 이전 작업에 대한 종료를 명확히 인지한 이후에 사용하기 바랍니다.

Return value

TRUE	연결 해지 성공
FALSE	연결 해지 실패

int **ConnectSerialPrinter** ( TCHAR\* sPortName, int iBaudRate, int iTimeoutMsec, BOOL  
bProtocol )

사용자 단말기에서 serial 통신을 사용하여 Woosim 프린터와 연결합니다.

serial 통신을 사용하여 프린터와 연결하는 경우, 이 함수를 사용하면 프린터와 연결이 가능합니다. 이 함수를 사용하기 앞서서, 프린터의 설정을 확인하기 바랍니다.

Parameter

<i>sPortName</i>	연결하려는 serial 포트 번호
<i>iBaudRate</i>	프린터의 통신 속도 ( 9600bps, 19200bps, 38400bps, 57600bps, 115200bps )
<i>iTimeoutMsec</i>	이 시간 내에 연결이 되지 않으면 연결 실패로 간주됨.
<i>bProtocol</i>	이 값은 항상 FALSE 입니다

Return value

SUCCESS	연결 성공
ALREADY_OPENED	이미 프린터가 연결됨
UNABLE_TO_OPEN_THE_PORT	포트 사용 불가
UNABLE_TO_CONFIGURE_THE_SERIAL_PORT	프린터 초기화 실패
UNABLE_TO_SET_THE_TIMEOUT_PARAMETERS	읽기 쓰기 시간 설정 실패
TIMEOUT	유효시간 내에 연결 실패

int **ConnectWirelessPrinter** ( TCHAR\* sIP\_ADDR, int iPortNum, int iTimeoutMsec, BOOL  
bProtocol )

사용자 단말기에서 IP 통신을 사용하여 Woosim 프린터와 연결합니다.

WiFi를 통해 사용자가 프린터를 사용하고자 한다면, 이 함수를 사용하면 프린터와 연결이 가능합니다. 이 함수를 사용하기 앞서서, 프린터의 설정을 확인한 이후 사용하기 바랍니다.

Parameter

<i>sIP_ADDR</i>	프린터가 할당 받은 IP address
<i>iPortNum</i>	프린터와 통신이 가능한 프린터의 port number
<i>iTimeoutMsec</i>	이 시간 내에 연결이 되지 않으면 연결 실패로 간주됨.
<i>bProtocol</i>	이 값은 항상 FALSE 입니다.

Return value

SUCCESS	연결 성공
---------	-------

SOCKET_ERROR	Socket 초기화 실패
CONNECT_FAIL	프린터 연결 실패
ALREADY_CONNECTED	이미 프린터가 연결됨
TIMEOUT	유효시간 내에 연결 실패

**void CutPaper( int mode )**

Cut 모드를 선택하고 용지를 자릅니다.  
Auto-cutter가 장착된 프린터에서 정상 동작합니다.

Parameter

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

**void EnterMSRMode( int n )**

설정된 track로 MSR 모드를 실행합니다.

Woosim printer에서는 MSR에 대해 3가지 타입의 카드 track 모드를 지원합니다: 12Track, 23Track, 123Track.

다음은 각 모드 별 지원 track 정보입니다.

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

읽기를 통해 전달되는 데이터에 대한 정보는 *Woosim Command manual*의 *Magnetic Card Data Output Format* 영역을 참조바랍니다.

Parameter

*n*                              Select card track (0~4).

**void EnterSCRMode( )**

SCR 모드를 실행합니다.

**void FeedToMark( )**

Black mark가 나타날 때까지 프린터의 용지를 출력합니다.

만일 SetPositionFromMark 함수로 인해 black mark에서 이동할 거리 n dots가 설정되어 있다면, 이 함수는 black mark에서 n dots 떨어진 위치까지 계속 paper feed를 할 것입니다. 이 함수는 프린터 설정에서 black mark sensor가 사용으로 되어 있는 경우에 정상적으로 동작하며, black mark가 없는 용지를 이용하는 경우에는 기본 용지 출력 값 30cm로 용지를 feed 합니다.

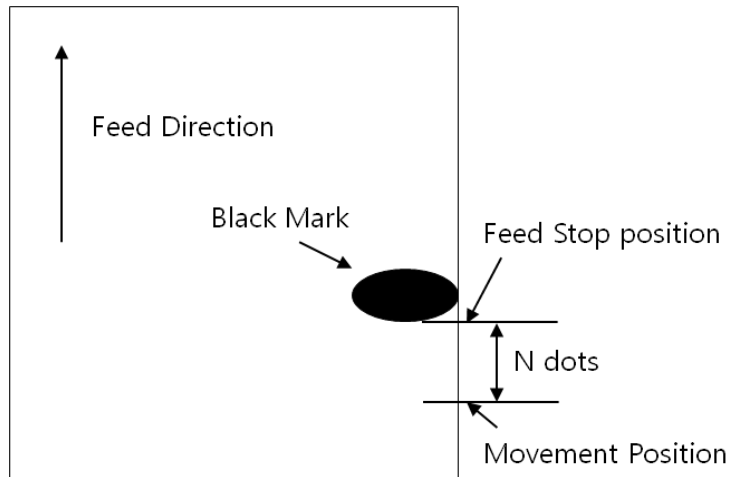
void **SetPositionFromMark**( int distance )

용지의 black mark에서부터 이동할 거리를 설정합니다.

이 명령어는 오직 프린터의 설정에서 black mark sensor가 사용으로 되어 있는 경우에만 동작을 합니다. 사용하기 전에, 프린터의 설정을 변경하고 올바른 용지를 장착하여야 합니다.

Parameter

*distance*      프린터가 인식한 black mark에서 n dot만큼 이동시키는 값



## 4. Sample Codes

Woosim embedded Windows SDK는 사용자가 편리하게 사용할 수 있도록 도와주기 위해 다음의 sample application을 제공하고 있습니다.

### 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

Sample application은 Visual Basic과 Visual C#, Visual C++의 3종류의 언어로 작성되어 있으며, 각각의 sample project들은 모두 같은 구성으로 작성되어 있습니다. 사용자는 sample application을 사용하기 앞서, 사용하려는 프린터의 연결 상태를 확인한 이후 사용하여야 합니다. 프린터의 self-test 결과나 LCD를 통해 프린터의 연결 방식을 확인 할 수 있습니다. 만일 Bluetooth 방식으로 연결을 원한다면, 사용자 단말기에서 Bluetooth에 할당된 port를 확인한 이후에 연결하여야 합니다.

바코드 출력 테스트에 앞서, 사용자는 사용하려는 프린터의 MCU를 고려하여야 합니다. Woosim printer는 RX MCU에 대해 모든 바코드 출력이 가능하며, 이외의 MCU에 대해서는 일부 2D 바코드에서 출력이 진행되지 않을 수 있습니다. MCU에 대한 정보는 프린터의 self test의 결과로 확인이 가능하며, sample project에서 GetPrinterModelName()를 사용하여 확인이 가능합니다. 또한 MSR mode를 사용하는 경우, 카드의 track을 먼저 확인 한 이후 올바른 읽기 모드를 사용하여야 정상적인 hexadecimal data를 얻을 수 있습니다.

마지막으로, TTF를 사용한 문자 출력은 Woosim printer에 미리 저장되는 TTF에 대해서만 지원을 하고 있습니다. 따라서 별도의 TTF에 대해서는 정상적인 동작을 보장받을 수 없습니다.

# Appendix

## Appendix A. Backward Compatibility

본 라이브러리는 신규 개발 및 편리한 개발 방법을 목표로 작성되었습니다. 따라서 배포되는 라이브러리는 대부분 새로운 API들로 구성되어 있으며, 기존의 라이브러리에 비해 다양한 기능들을 제공합니다.

배포되는 SDK는 정책적으로 backward compatibility를 지원하고 있습니다. 기존의 Woosim embedded Windows 라이브러리를 사용한 프로그램에서 신규 라이브러리로 변경하여 추가된 기능들을 사용하고자 하신다면, 기존 라이브러리에서 선언한 정보를 유지하면서 신규 라이브러리의 정보를 추가하면 이전 버전의 API와 새로운 API를 병행하여 사용하실 수 있습니다.

이제 Woosim embedded Windows SDK v4.0 이전의 SDK에서의 대부분의 API들은 공식적으로 지원되지 않습니다. 따라서 기존의 사용자들이 새로 배포되는 Woosim embedded Windows SDK를 사용하여 신규 프로젝트를 진행한다면, SDK v4.0 이전 라이브러리의 API를 사용하여 작업하는 것을 권장하지 않습니다. 신규 라이브러리의 API를 사용하여 작성하길 바랍니다.

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

다음은 C/C++ 언어의 data type과 .net language를 사용하는 visual basic, C#의 data type의 변환에 대한 matching table입니다. 자세한 내용은 MSDN을 참조하기 바랍니다.

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	TCHAR*	System::String or System.Text::StringBuilder	Decorate with ANSI.
LPCSTR	Const TCHAR*	System::String or System.Text::StringBuilder	Decorate with ANSI.



<b>LPWSTR</b>	wchar_t*	System::String or System.Text::StringBuilder	Decorate with Unicode.
<b>LPCWSTR</b>	Const wchar_t*	System::String or System.Text::StringBuilder	Decorate with Unicode.

## Appendix C. Two-dimensional barcode table

다음은 2D 바코드의 세부 설정 table 입니다.

### ● 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 (%).

### ● QRCode 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: level별 error correction code rate (%).

#### ● 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