

RFID Printer 프로그래밍 가이드 For BT-200

Version 2.4.3.2

2024 / 04



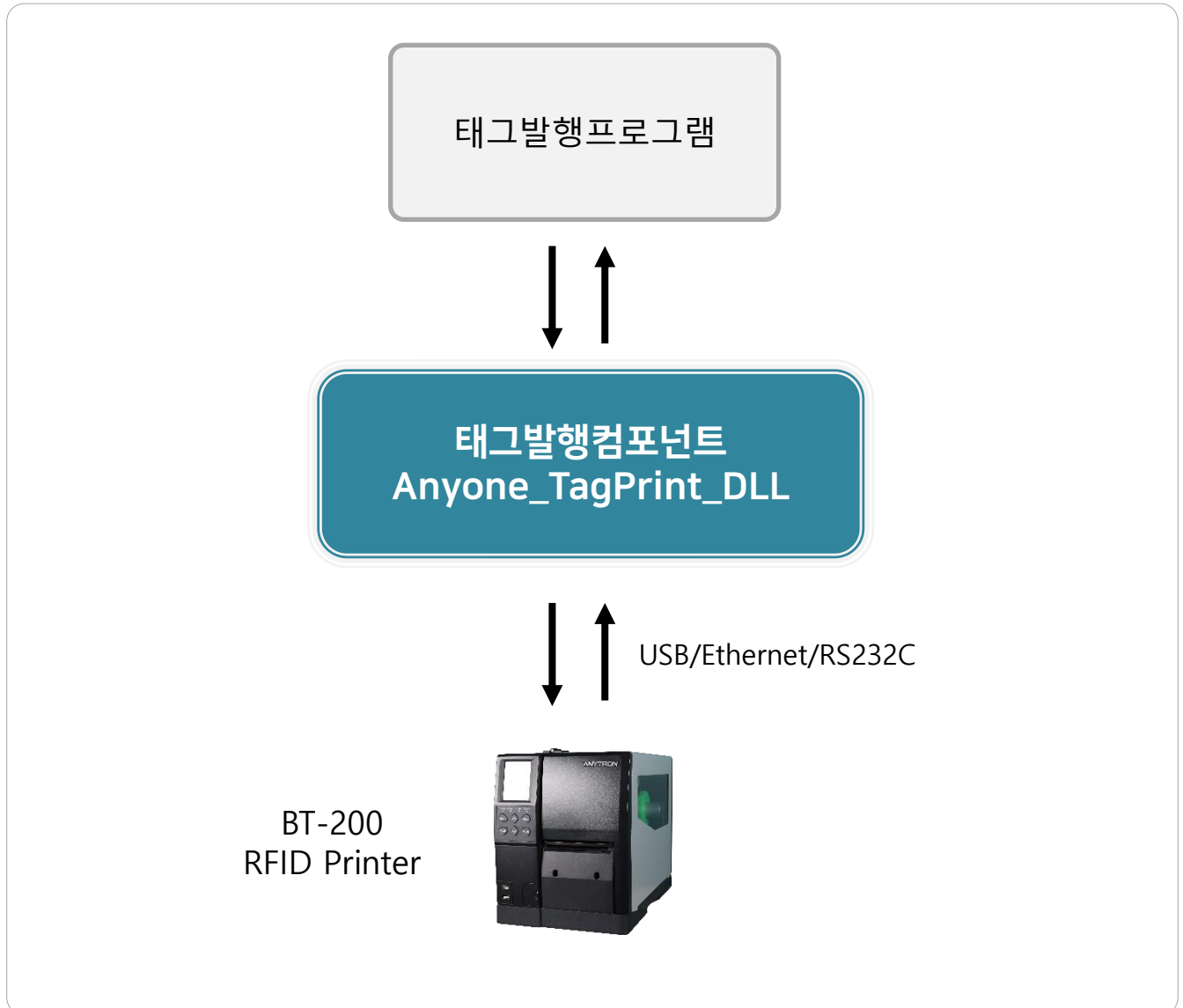
1. Anyone_TagPrint_DLL for BT-200

1.1 개요

본 문서는 BT-200 RFID 프린터를 이용한 태그발행프로그램 개발에 필요한 라이브러리의 인터페이스 규약에 대해 설명합니다.

Anyone_TagPrint_DLL 라이브러리는 C++로 작성이 되었으며, Windows 환경에서 C++, 닷넷, 델파이 등의 개발언어에서 사용이 가능합니다.

Anyone_TagPrint_DLL은 ANYONE 발행기와 USB, Ethernet, RS232C 중 선택적으로 연결 사용이 가능합니다.



2. 메소드

2.1 Connect

원형	int Connect(int iType, int iPort, LPCTSTR strIpAddress, LPCTSTR strModel);	
반환값	실행결과. 0:성공, 그외의 경우는 에러. 에러코드는 별도의 에러코드 참조	
인자	int iType	사용할 인터페이스의 종류 0:RS232, 1:USB, 2:Ethernet
	int iPort	TCP/IP 포트번호
	LPCTSTR strIpAddress	TCP/IP일 경우 IP주소, 기타의 경우 ""
	LPCTSTR strModel	연결할 프린터의 모델명, 특별히 지정하지 않을 경우 ""
설명	<p>태그발행기에 지정된 포트를 사용하여 연결합니다. USB일 경우 포트번호는 0을 지정합니다.</p> <p>RS232는 COM1, COM2 ... 등의 포트를 사용하는데, 이때 iType = 0, iPort의 값은 COM뒤의 인덱스 번호를 지정합니다. Ex) COM4 : iType=0, iPort=4</p> <p>Ethernet인 경우는 iType=2, iPort=9100, strIpAddress는 태그발행기에 설정된 IP주소를 지정합니다.</p>	
예제	<pre>// USB인경우 Connect(1,0, "", "BT-200"); // Ethernet인 경우 Connect(2,9100,"192.168.0.198" , "BT-200"); // RS232 COM4인 경우 Connect(0,4,"", "BT-200");</pre>	

2.2 Connect2

원형	int Connect2();	
반환값	int	수행결과 0 : 성공, 1 : 실패
설명	<p>설정화면에서 지정한 연결방법과 연결정보를 토대로 태그발행기와 연결을 시도합니다.</p> <p>연결정보를 직접 지정하고자 할 경우에는 Connect 메소드를 사용하십시오.</p>	

2.3 ClosePort

원형	int ClosePort(LPCTSTR strSerial);	
반환값	int	수행결과 0 : 성공, 1 : 실패
인자	LPCTSTR strSerial	프린터 일련번호, 특별히 지정하지 않을 경우 ""
설명	연결된 프린터와의 통신 연결을 닫습니다.	

2.4 IsConnected

원형	int IsConnected();	
반환값	int	프린터 통신연결 상태 0 : 연결안됨, 1 : 연결됨
설명	프린터의 통신연결상태를 반환합니다. 프린터가 호스트와 통신이 연결된 상태이면 1, 연결되어 있지 않으면 0을 반환합니다.	
예제	bool bConnected = TagPrintWrapper.IsConnected ("");	

2.5 SendBytes

원형	void SendBytes(LPBYTE pData, ULONG count);	
반환값	없음	
인자	LPBYTE pData	발행기로 전송할 바이너리 데이터
	ULONG count	발행기로 전송할 바이너리 데이터의 길이
설명	연결된 태그발행기로 이진데이터를 전송합니다.	

2.6 SendControlCommand

원형	void SendControlCommand(LPCTSTR strCmd);	
반환값	없음	
인자	LPCTSTR strCmd	발행기로 전송할 문자열 형태의 제어명령
설명	연결된 태그발행기로 파라미터로 전달된 제어명령을 전송합니다.	
예제	<pre>string strCmd = "& V1 do ₩blackmark_learn₩"; TagPrintWrapper.SendControlCommand(strCmd);</pre>	

2.7 SendFile

원형	void SendFile(LPCTSTR strFileName);	
반환값	없음	
인자	LPCTSTR sFileName	파일명 (절대경로)
설명	인자로 전달된 파일을 읽어서 연결된 프린터로 파일 전체를 전송합니다.	
예제	<pre>string strFileName = "C:\\\\Test\\file1.txt"; TagPrintWrapper.SendFile(strFileName);</pre>	

2.8 SendStr

원형	void SendStr(LPCTSTR strCmd);	
반환값	없음	
인자	LPCTSTR strCmd	프린터로 보내는 문자열 명령 또는 데이터
설명	<p>태그발행과는 상관없이 사용자가 프린터로 명령을 보내는데 사용하는 메소드임. 주로 발행전에 프린터의 상태체크를 한다거나, 프린터를 직접 제어하기 위해 사용됩니다.</p> <p>SendControlCommand 메소드와의 차이점은 SendControlCommand는 주로 프린터의 내부 제어명령을 전송하는데 사용하고, SendStr메소드는 ZPL 명령을 전송하는데 사용하는 차이가 있습니다.</p> <p>ZPL 명령을 프린터로 보내고자 할 경우에는 SendStr 메소드를 사용하십시오.</p>	
예제	<pre>string strCmd = "~SD12WrWn"; // 인쇄농도를 12로 설정 TagPrintWrapper.SendStr(strCmd); strCmd = "~PHWrWn"; // 라벨 1장 이송 TagPrintWrapper.SendStr(strCmd);</pre>	


2.9 SetFileName

원형	int SetFileName(LPCTSTR strFileName);	
반환값	실행결과. 0:성공, 그외의 경우는 에러. 에러코드는 별도의 에러코드 참조	
인자	LPCTSTR strFileName	확장자를 포함한 서식파일명(절대경로)
설명	<p>출력할 라벨서식에 대한 서식파일명을 지정합니다.</p> <p>서식파일명은 경로명과 함께 절대경로 형식으로 지정하여야 합니다.</p> <p>서식파일의 형식은 텍스트 파일이며, 라벨의 크기, 출력할 내용, RFID 관련 명령 등을 담고 있습니다.</p> <p>서식파일에 대한 자세한 명령 및 사용방법은 ZPL 프로그래밍 매뉴얼을 참조.</p>	
예제	TagPrintWrapper.SetFileName("C:\\\\Test\\\\myForm.txt");	

2.10 SetVariableData

원형	int SetVariableData(LPCTSTR strData, LPCTSTR seperator);	
반환값	실행결과. 0:성공, 그외의 경우는 에러. 에러코드는 별도의 에러코드 참조	
인자	LPCTSTR strData	가변데이터 문자열
	LPCTSTR seperator	가변데이터 문자열의 항목을 구분하기 위한 구분자. 일반적으로 "!!"를 사용
설명	<p>서식파일내의 변수(^Vxx)에 매핑될 데이터를 지정합니다.</p> <p>각 데이터의 구분은 seperator에 의해 구분합니다.</p> <p>데이터 매핑은 가변데이터 문자열의 순서대로 ^V00, ^V01, ^V02 ... 순으로 대응됩니다.</p>	
예제	<div style="display: flex; align-items: center;"> <div style="flex: 1;"> <p>TagPrintWrapper.SetFileName("myTest.txt");</p> <p>TagPrintWrapper.SetVariableData("홍길동!!11223344556677889900AABBCCDDEEFF!!20090712", "!!");</p> <p><myTest.txt의 내용 예시></p> <pre> ^XA ^LH75,130 ^FO100,100^A@N,32,32 ^FD^V00^FS ^FO100,150^A@N,32,32 ^FD^V02^FS ^RFW,H,1,18,1 ^FD41B1^V01 ^XZ </pre> </div> <div style="flex: 0.1; text-align: center; margin: 0 10px;">➔</div> <div style="flex: 1;"> <p><발행기로 전달되는 내용></p> <pre> ^XA ^LH75,130 ^FO100,100^A@N,32,32 ^FD홍길동^FS ^FO100,150^A@N,32,32 ^FD41B111223344556677889900AAB BCCDDEEFF^FS ^RFW,H,1,18,1 ^FD41B1^V01 ^XZ </pre> </div> </div>	

2.11 SetUp

원형	void SetUp();
반환값	없음
인자	없음
설명	<p>연결설정, RFID 옵션, 인쇄옵션, 인쇄위치 등의 설정에 사용됩니다.</p> <p>메소드를 호출하면, 예제에서와 같은 대화창을 보여주며, "확인" 버튼을 누르면 지정된 값이 레지스트리에 저장되고, 데이터를 프린터로 전송한 후 대화창을 닫습니다. "취소" 버튼을 누르면 설정을 적용하지 않고 대화창을 닫습니다.</p> <p>대화창에 표시되는 항목은 컴포넌트 버전에 따라 다소 상이할 수 있습니다.</p> <p>SetUp 메소드를 호출하기 전에 Connect 메소드의 호출이 선행되어야 합니다. 지정된 포트에 연결할 수 없을 경우에는 설정값은 레지스트리에 저장되나, 프린터로 설정값이 전송되지 않습니다.</p>
예제	

2.12 TagPrint

원형	int TagPrint(LPTSTR IpResult, LPTSTR IpErrorMessage)	
반환값	실행결과. 0:성공, 그외의 경우는 에러. 에러코드는 별도의 에러코드 참조	
인자	LPTSTR IpResult	형식 : EPC(16진수), 발행결과코드(16진수) EPC코드와 발행결과코드를 ","로 구분하여 반환.
	LPCTSTR IpErrorMessage	발행 오류가 있을 경우의 오류코드 내용
설명	<p>SetFileName()과 SetVariableData()에 의해 지정된 서식파일과 데이터를 가지고 태그 발행을 진행합니다.</p> <p>TagPrint()를 호출하기에 앞서 반드시 서식파일이 미리 지정되어 있어야 합니다.</p> <p>성공여부와 에러코드는 반환값을 통해 확인이 가능하고, 매개변수를 통해 발급한 태그의 EPC 데이터와 실패한 경우의 오류내용을 확인할 수 있습니다.</p>	
예제	<pre> <C++ MFC 에서 이용하는 경우 예시> Int iResult = -1; TCHAR szEpcBuf[512] = {0x00,}; TCHAR szErrorMessage[128] = {0x00,}; SetFileName(_T("C:\\\\Test\\\\myForm.txt")); SetVariableData(_T("홍길동!!123456!!10000!!100A1011230D0D0A0CDD11"), _T("!!")); iResult = TagPrint(szEpcBuf, szErrorMessage); if (iResult == 0) { // 성공처리 } else { // 실패처리 } <C#에서 이용하는 경우 예시> int iResult = -1; StringBuilder sbResult= new StringBuilder(512); StringBuilder sbErrorMessage = new StringBuilder(128); TagPrintWrapper.SetFileName("C:\\\\Test\\\\myForm.txt"); TagPrintWrapper.SetVariableData("홍길 동!!123456!!10000!!100A1011230D0D0A0CDD11", "!!"); iResult = TagPrintWrapper.TagPrint(sbResult, sbErrorMessage); if (iResult == 0) { // 성공처리 } else { // 실패처리 } </pre>	

2.13 TagPrint2

원형	int TagPrint2(LPTSTR IpResult, LPTSTR IpErrorMessage, LPINT IpErrorCount)	
반환값	실행결과. 0:성공, 그외의 경우는 에러. 에러코드는 별도의 에러코드 참조	
인자	LPTSTR IpResult	형식 : EPC(16진수), 발행결과코드(16진수) EPC코드와 발행결과코드를 ","로 구분하여 반환.
	LPCTSTR IpErrorMessage	발행 오류가 있을 경우의 오류코드 내용
	LPINT IpErrorCount	발행 오류 횟수
설명	<p>SetFileName()과 SetVariableData()에 의해 지정된 서식파일과 데이터를 가지고 태그 발행을 진행합니다.</p> <p>TagPrint()를 호출하기에 앞서 반드시 서식파일이 미리 지정되어 있어야 합니다.</p> <p>성공여부와 에러코드는 반환값을 통해 확인이 가능하고, 매개변수를 통해 발급한 태그의 EPC 데이터와 실패한 경우의 오류내용을 확인할 수 있습니다.</p> <p>발행 오류가 있어 프린터 내부적으로 Retry가 있을 경우 IpErrorCount를 통해 오류매수 확인이 가능합니다.</p>	
예제	<p><C++ MFC 에서 이용하는 경우 예시></p> <pre> Int iResult = -1; TCHAR szEpcBuf[512] = {0x00,}; TCHAR szErrorMessage[128] = {0x00,}; Int iErrorCount = 0; SetFileName(_T("C:\\\\Test\\\\myForm.txt")); SetVariableData(_T("홍길동!!123456!!10000!!100A1011230D0D0A0CDD11"), _T("!!")); iResult = TagPrint2(szEpcBuf, szErrorMessage, &iErrorCount); if (iResult == 0) { // 성공처리 } else { // 실패처리 } </pre> <p><C#에서 이용하는 경우 예시></p> <pre> int iResult = -1; Int iErrorCount = 0; StringBuilder sbResult= new StringBuilder(512); StringBuilder sbErrorMessage = new StringBuilder(128); TagPrintWrapper.SetFileName("C:\\\\Test\\\\myForm.txt"); TagPrintWrapper.SetVariableData("홍길 동!!123456!!10000!!100A1011230D0D0A0CDD11", "!!"); iResult = TagPrintWrapper.TagPrint2(sbResult, sbErrorMessage, out iErrorCount); if (iResult == 0) { // 성공처리 } else { // 실패처리 } </pre>	

2.14 ReadResponse

원형	void ReadResponse(LPTSTR lpResp);	
반환값	없음	
인자	LPTSTR lpResp	태그발행기로부터 수신된 데이터
설명	<p>태그발행기에서 보내온 응답문자열을 파라미터로 반환합니다.</p> <p>ReadResponse()명령에 의해 발행기와 연결되어 있는 포트의 수신 버퍼 내용을 읽어 반환하며, 읽혀진 내용은 수신버퍼에서 지워집니다.</p> <p>응용 프로그램내에서 사용할 때에는 타이머등을 이용하여 주기적으로 수신버퍼의 내용을 읽어주도록 구현하는 것이 필요합니다.</p>	
예제	<pre> TagPrintWrapper.SetFilename("C:WWTestWWtest.txt"); TagPrintWrapper.SetVariableData("111!!112233445566778899AABBCC!!ABC", "!!"); iResult = TagPrint(sbEpcData, sbErrorMessage); TagPrintWrapper.ReadResponse(sbBufResponse); ---> sbBufResponse : "1 page printed.EPC:112233445566778899AABBCC.error- code:0000WrWn" </pre>	

2.15 SetPrinterSetting

원형	void SetPrinterSetting(ST_PrinterSetting prtSet);	
반환값	없음	
인자	ST_PrinterSetting prtSet	프린터 세팅정보를 위한 구조체
설명	<p>프린터 세팅정보를 갖는 구조체를 매개변수로 전달하여 설정을 수행합니다. ST_PrinterSetting 구조체의 원형은 아래와 같습니다.</p> <pre>typedef struct _stPrinterSetting { LPCTSTR strModelName; // 모델명 LPCTSTR strIpAddress; // IP주소 LPCTSTR strSerialPort; // RS232포트 int iConnectType; // 연결유형 int iTcpPort; // TCP포트번호 int iRfAttenW; // RFID Write 출력감쇄값 int iRfAttenR; // RFID Read 출력감쇄값 int iRfEncOffset; // RFID 엔코딩위치 오프셋 int iCutPos; // 커팅위치 오프셋 int iDarkness; // 인쇄농도 int iSpeed; // 인쇄속도 int iMediaType; // 용지유형 int iSensorType; // 센서유형 int iHorOffset; // 수평인쇄 위치 오프셋 int iVerOffset; // 수직인쇄 위치 오프셋 bool bUseUMI; // UMI Field 세팅 } ST_PrinterSetting;</pre>	
예제	<pre>ST_PrinterSetting stPrtConfig; // 구조체의 값 채우기 stPrtConfig.strModelName = "ANYONE"; ... // 프린터 설정 TagPrintWrapper.SetPrinterSetting(stPrtConfig);</pre>	

2.16 ExecuteBT200InternalCommand

원형	int ExecuteBT200InternalCommand(LPCTSTR strCommand, LPTSTR lpResponse);	
반환값	실행결과. 0-정상, 그외의 경우 에러	
인자	LPCTSTR strCommand	발행기로 전송할 문자열 형태의 내부 명령구문
	LPTSTR lpResponse	실행결과 문자열 데이터
설명	<p>연결된 태그발행기로 파라미터로 전달된 내부 명령구문을 전송하고, 그결과를 문자열 형태의 파라미터로 반환합니다.</p> <p>주로 프린터 내부의 설정값을 조회할 때 사용한다.</p> <p>명령구문 형식 : & V1 getval 질의종류</p> <p>질의종류는 아래 참조.</p> <ul style="list-style-type: none"> - 프린터 모델 : printer - 시리얼 번호 : serial_no - MAC 주소 : eth_mac - IP 주소 : eth_ip - DHCP : eth_dhcp - 서브넷마스크 : eth_mask - Gateway : eth_gateway - 프린터 상태 : printer_status - 버전 : version - 마일리지 : mileage - 센서 : sensor_select - 용지종류 : media_sensor - 인쇄농도 : density - 인쇄속도 : speed - RFID옴셋 : rfid_pos_offset - RFID읽기 파워 : rfid_write_power - RFID쓰기 파워 : rfid_read_power 	
예제	<pre>string strCmd = "& V1 getval WprinterW"; StringBuilder sb = new StringBuilder(); TagPrintWrapper.ExecuteBT200InternalCommand(strCmd, sb); String strPrinterName = sb.ToString();</pre>	

3. 반환코드 리스트

반환코드값	내 용	비고
0x0000	OK	
0x0010	RFID 리더모듈 데이터 전송프레임 에러	
0x0011	RFID 리더모듈 데이터 형식 에러	
0x0012	RFID 리더모듈 BCC 에러	
0x0021	RFID 설정 에러	
0x0022	RFID 리더모듈에서 지원하지 않는 명령임	
0x0023	RFID 리더모듈 펌웨어 형식 에러	
0x0024	RFID 리더모듈의 Flash 메모리 에러	
0x0025	RFID 리더모듈 펌웨어 기록 에러	
0x0071	RFID 읽기 에러	
0x0072	RFID 쓰기 에러	
0x0073	RFID 태그 감지 에러	
0x0074	RFID 메모리 주소 에러	
0x0076	RFID 지원하지 않는 태그 명령	
0x0079	RFID 오퍼레이션 타임아웃	
0x0080	RFID 파라미터 설정 에러	
0x00FF	RFID 모듈관련 UNKNOWN 에러	
0x0100	용지 없음	
0x0200	용지 걸림 또는 용지 센싱 오류	
0x0300	리본 에러	
0x0400	프린트 헤드 열림	
0x0900	프린터의 상태가 에러상태임	
0x0A00	프린터 커터 걸림	
0x0E00	프린터 펌웨어 업데이트 실패	
0x0F00	프린트 헤드 에러	
0x1000	프린트 헤드 과열 감지	
0x1100	프린터 메모리 쓰기 에러	
0x6500	프로그램 타임아웃	
0x6600	포트연결 실패	
0x6700	서식파일 없음	
0x6800	서식파일 열기 실패	
0xCA00	데이터 전송 실패	

4. 닷넷환경에서의 개발

닷넷환경에서 C++로 작성된 DLL을 사용하기 위해서는 아래와 같이 Wrapper Class를 별도로 작성해서 사용하면 아주 간단하면서도 편리하게 사용이 가능합니다. 다른 플랫폼간의 매개변수 호환성을 위해 마샬링을 사용하는 점에 유의해 주시기 바랍니다.

```
using System.Runtime.InteropServices;
using System.Text;

namespace ANYONE_LIB
{
    [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]
    public struct ST_PrinterSetting
    {
        [MarshalAs(UnmanagedType.LPWStr)]
        public string strModelName;

        [MarshalAs(UnmanagedType.LPWStr)]
        public string strIpAddress;

        [MarshalAs(UnmanagedType.LPWStr)]
        public string strSerialPort;

        public int iConnectType;
        public int iTcpPort;
        public int iRfAttenW;
        public int iRfAttenR;
        public int iRfEncOffset;
        public int iCutPos;
        public int iDarkness;
        public int iSpeed;
        public int iMediaType;
        public int iSensorType;
        public int iHorOffset;
        public int iVerOffset;

        [MarshalAs(UnmanagedType.I1)]
        public bool bUseUMI;
    }
}
```

```

public static class TagPrintWrapper
{
    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern void SetUp();

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern int SetFileName([MarshalAs(UnmanagedType.LPWStr)] string strFileName);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern int SetVariableData([MarshalAs(UnmanagedType.LPWStr)] string strTagDts,
    [MarshalAs(UnmanagedType.LPWStr)] string strDelim);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern int TagPrint([MarshalAs(UnmanagedType.LPTStr, SizeConst = 512)] StringBuilder sResult,
    [MarshalAs(UnmanagedType.LPTStr, SizeConst = 128)] StringBuilder sErrorMessage);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern int TagPrint2([MarshalAs(UnmanagedType.LPTStr, SizeConst = 512)] StringBuilder sResult,
    [MarshalAs(UnmanagedType.LPTStr, SizeConst = 128)] StringBuilder sErrorMessage, [MarshalAs(UnmanagedType.I4)
    out int nErrorCount];

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern void SendStr([MarshalAs(UnmanagedType.LPWStr)] string strSend);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern void ReadResponse([MarshalAs(UnmanagedType.LPTStr, SizeConst = 512)] StringBuilder
    sResponse);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern void SendFile([MarshalAs(UnmanagedType.LPWStr)] string strFilePath);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern void SendBytes([MarshalAs(UnmanagedType.LPArray)] byte[] pPuf, ulong count);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern int ClosePort([MarshalAs(UnmanagedType.LPWStr)] string strSerial);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern int IsConnected();

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern void SendControlCommand([MarshalAs(UnmanagedType.LPWStr)] string strCommand);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern int Connect(int iType, int iPortIndex, [MarshalAs(UnmanagedType.LPWStr)] string strIp,
    [MarshalAs(UnmanagedType.LPWStr)] string strModel);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern int Connect2();

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern void SetPrinterSetting(ST_PrinterSetting setupValue);

    [DllImport("Anyone_TagPrint_DLL.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern int ExecuteBT200InternalCommand([MarshalAs(UnmanagedType.LPWStr)] string
    strCommand, [MarshalAs(UnmanagedType.LPTStr, SizeConst = 512)] StringBuilder sResponse);

    static TagPrintWrapper()
    {
    }
}

```


Wrapper Class 정의가 끝나면, 아래에서와 같이 직접 참조하여 메소드를 호출하여 사용이 가능합니다.

보다 세부적인 내용에 대해서는 함께 제공하는 별도의 샘플소스 코드를 참고하십시오.

```
private void btnSetup_Click(object sender, EventArgs e)
{
    TagPrintWrapper.Setup();
}

private void btnPrintStart_Click(object sender, EventArgs e)
{
    int iResult = 0;
    string strFileName = @"C:\Test\myForm.txt";
    TagPrintWrapper.SetFileName(strFileName);
    TagPrintWrapper.SetVariableData("홍길동!!대한민국!!12345678!!112233445566778899AABBCC","!!");

    // USB로 발행기 연결
    iResult = TagPrintWrapper.Connect(1, 0, "", "BT-200");

    if (iResult != 0)
    {
        MessageBox.Show("발행기 연결 실패");
        return;
    }

    StringBuilder sbResult = new StringBuilder(512);
    StringBuilder sbErrorMessage = new StringBuilder(128);

    int iResult = TagPrintWrapper.TagPrint(sbResult, sbErrorMessage);

    if (iResult == 0)
    { // 발행 성공 처리 }
    else
    { // 발행 실패 처리 }
}
```