OPTO ELECTRONICS CO., LTD.

OPN-2002i SERIES/OPN-3002i SERIES

DATA collector

OPN SDK ユーザーズガイド（MFC）

株式会社 オプトエレクトロニクス

改版履歴

資料管理番号：SI13026

発行管理番号 ：DM-130706

製　品　名 ：OPN-2002i Series / OPN-3002i Series

版	日付	変更箇所	内容
Rev 1.0	2013/7/22	－	新規作成

はじめに

本書は、データコレクタ「OPN-3002i」、「OPN-2002i」を利用した MFC 向けアプリケーションの開発に利用されることを目的としています。

データコレクタの操作方法や、MFC 開発で必要となる一般的な技術情報につきましては、それぞれの説明書などを参照してください。

本書の構成

1. 開発環境

本 SDK を利用できる条件や手順について、解説しています。

2. APIの利用

クラスライブラリの主な API について、概要と利用方法をサンプルコードと合わせて解説しています。

コマンドリファレンスについては、「OPN コマンドリファレンス.doc」を参照して下さい。API の詳細については、本書には記載していません。「OPNTermSDK_API リファレンス_COM.doc」を参照してください。

目次

はじめに 2

本書の構成 2

1. 開発環境 2

2. APIの利用 2

目次 3

1. 開発環境 4

1.1 ビルド環境 4

1.2 実行環境 4

1.3 開発環境でのSDK使用手順 4

■MFCでファーストアプリを作成 5

2. APIの利用 13

■ OPN2002iBluetoothServiceを生成・取得する 13

■ デリゲートの設定 13

■ MFC側がデータコレクタからの接続を待ち受ける 14

■ MFC側がデータコレクタからの接続待ちを中断する 14

■ 指定したデバイスにMFC側をクライアントとして接続します 14

■ 切断する 14

■ API標準のコマンドを実行する 17

1. 開発環境

1.1 ビルド環境

本アプリケーションのビルド環境は以下の通りです。

・ Visual Studio 2012 Update3 (RC2)

1.2 実行環境

本 SDK を利用して開発できる MFC アプリケーションは、以下のバージョンの OS をターゲットとしたアプリケーションです。

OS
Windows XP SP3
Windows Vista SP2
Windows 7
Windows 8

また本プログラムをご利用の前に、Bluetooth 設定で「新規の Bluetooth デバイスによる、接続試行時には警告する」のチェックボックスを外してください。

1.3 開発環境でのSDK使用手順

本 SDK のクラスライブラリを利用するには、以下の点を対応する必要があります。

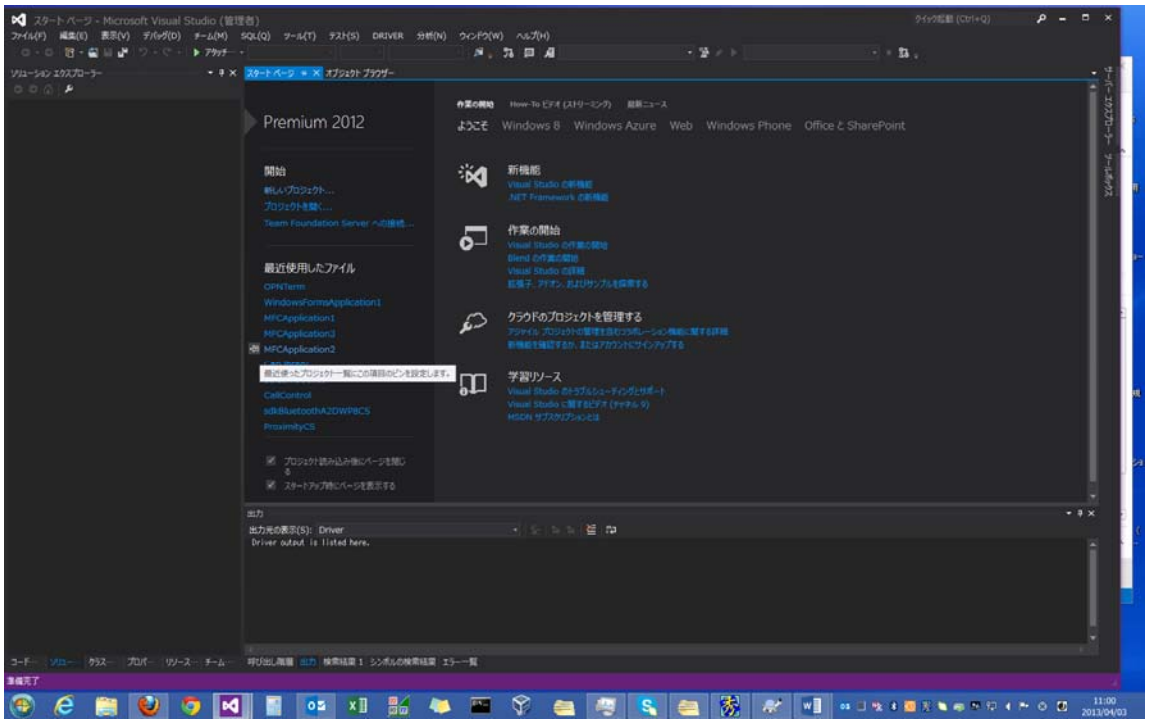
- インクルードファイルおよびライブラリを開発環境に取り込む

（前提）

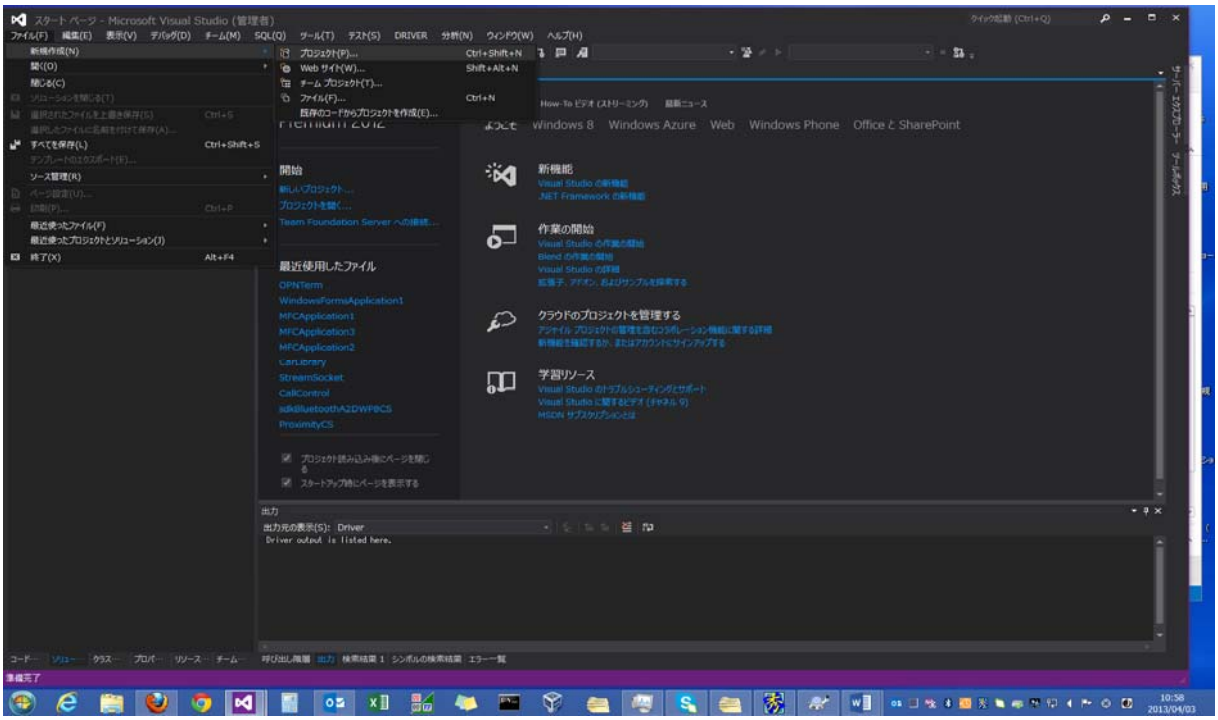
Visual Studio 2012 上で MFC アプリケーションを開発するための一般的な環境が整っている状態を前提とします。以下の例では、MFC プロジェクトの作成手順から解説します。また、本書では、開発環境として Visual Studio 2012 を使った場合の手順を解説します。

■MFCでファーストアプリを作成

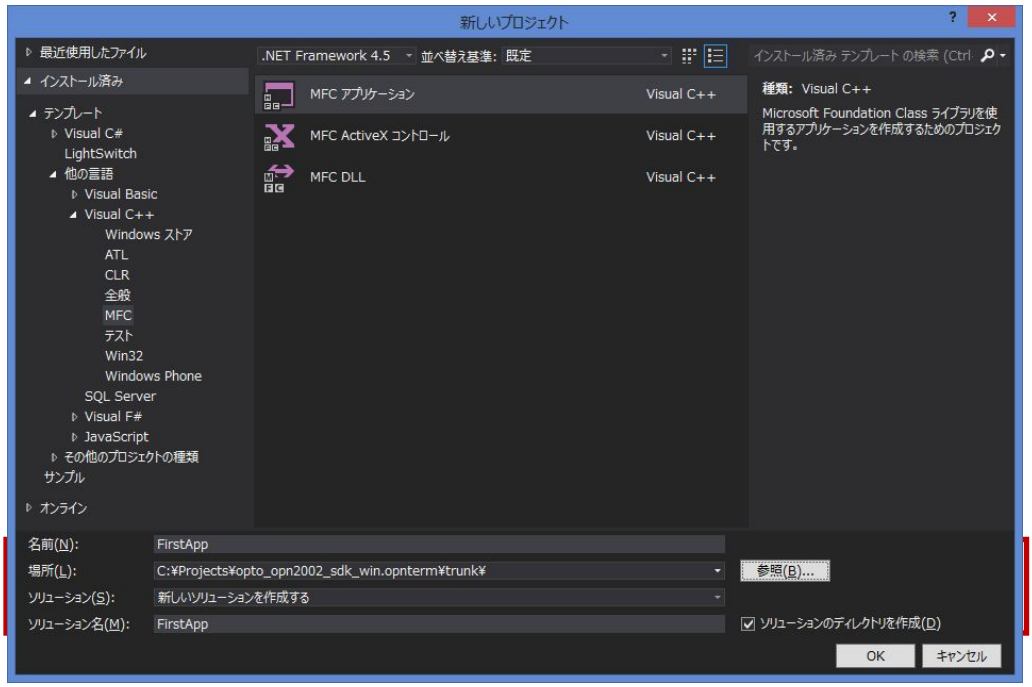
① VisualStudio2012 を起動します。



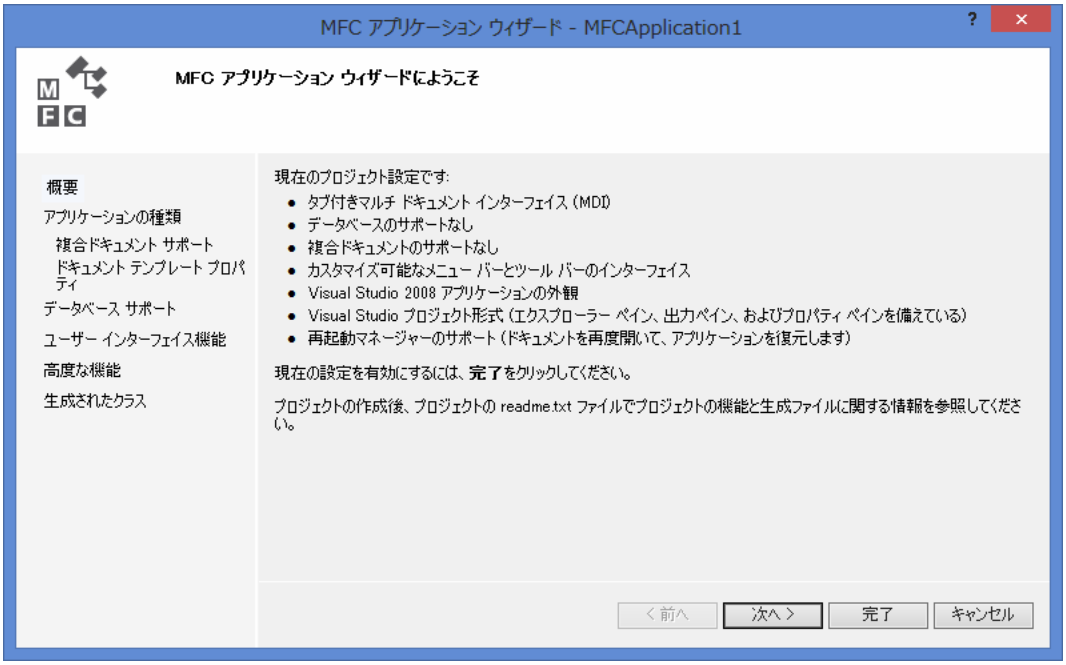
② ファイル→新規作成→プロジェクトを選択します。



③ MFC アプリケーションを選択する。（下のダイアログが表示される）
名前、場所、ソリューション名を入力し、「OK」をクリックします。

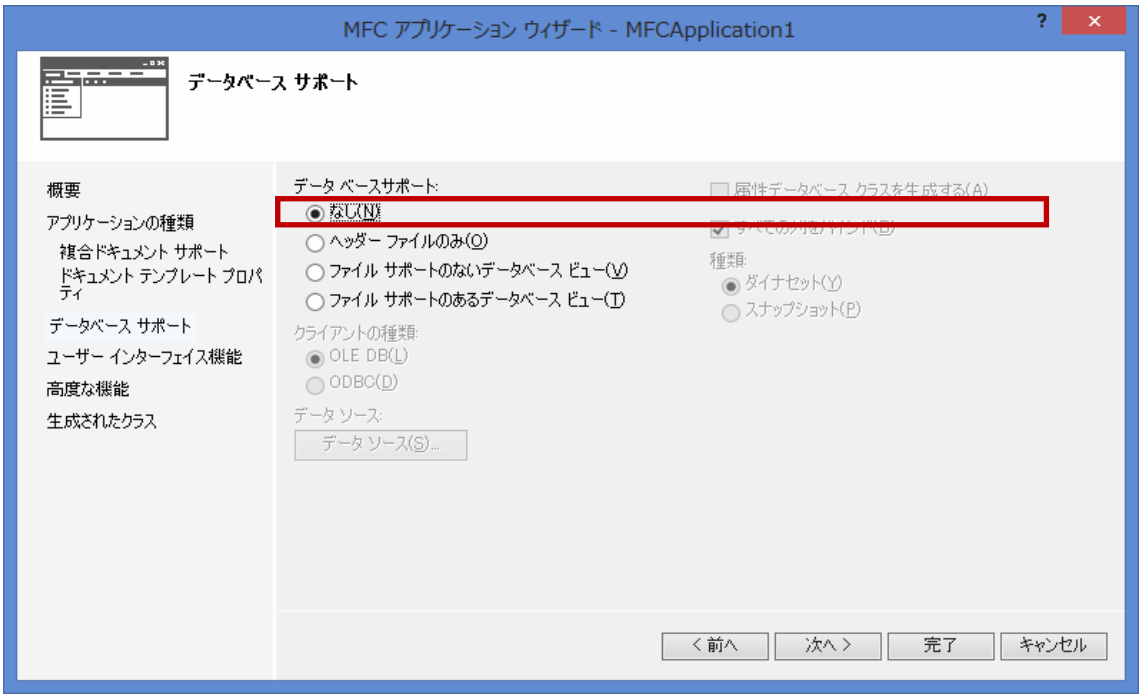


④ 「次へ」をクリックします。

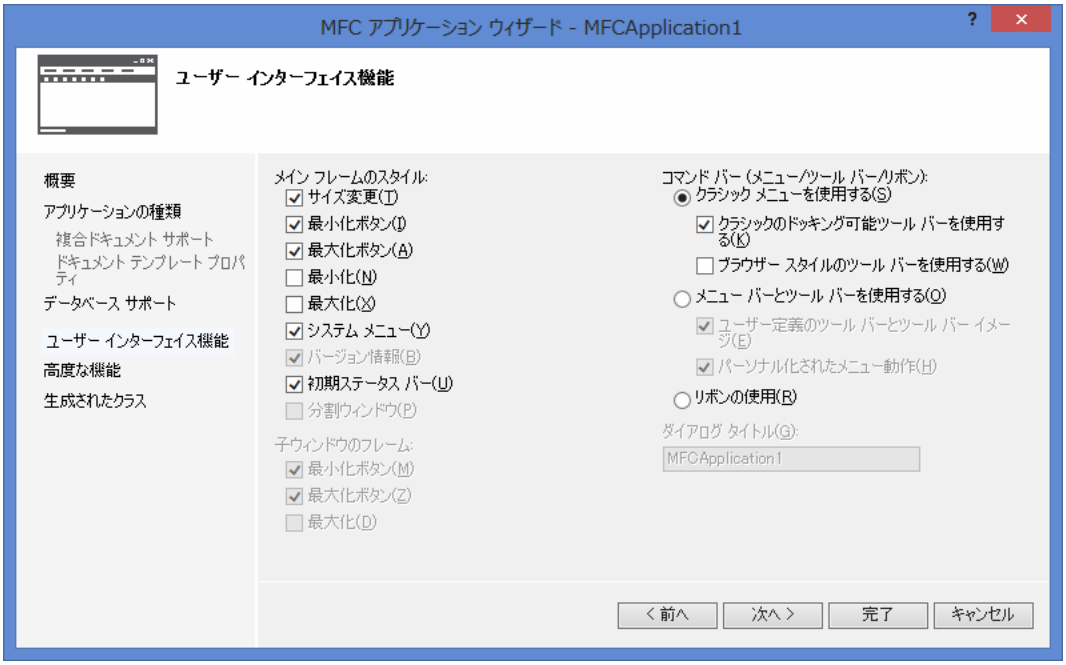


⑤ 「シングルドキュメント」を選択し、上図のように「Unicode ライブラリを使用する」などのチェックを外し、「次へ」をクリックします。

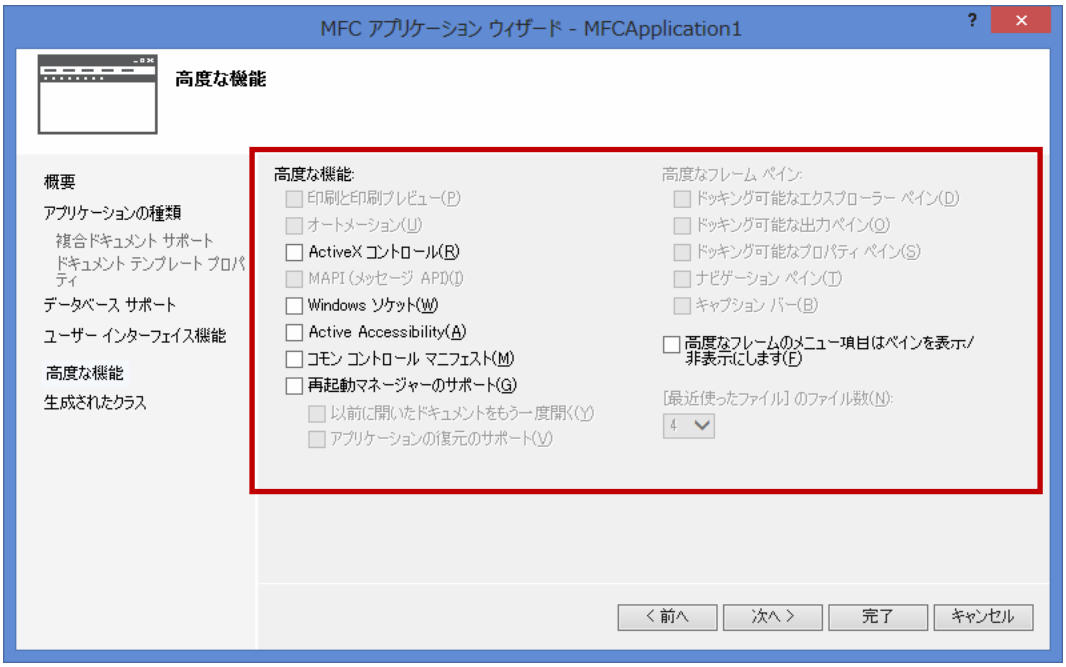
⑥ 「次へ」をクリックします。



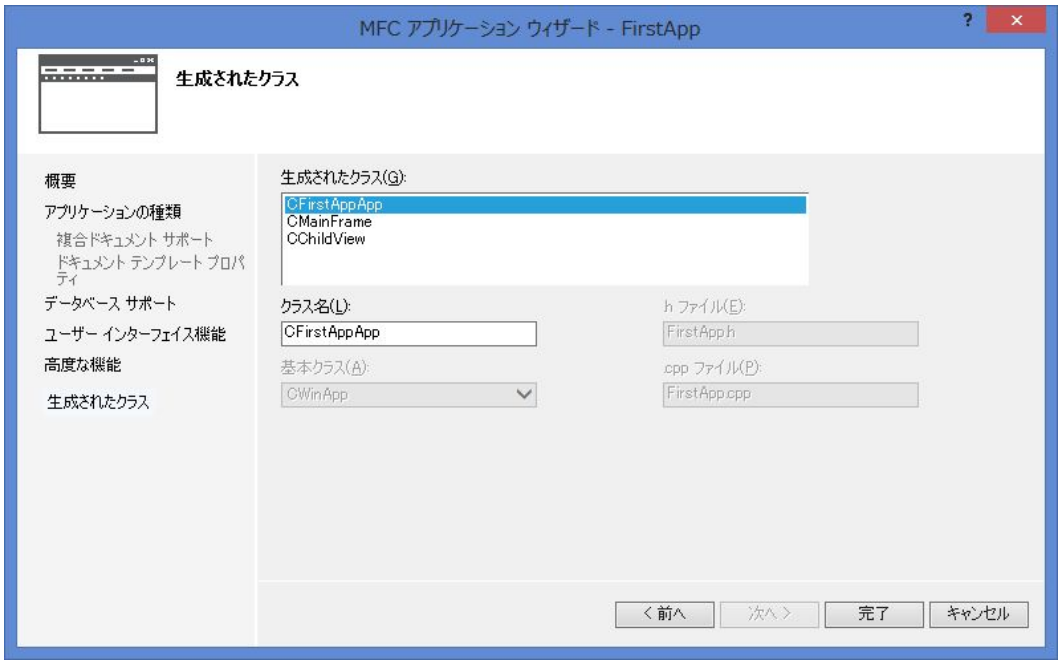
⑦ 「次へ」をクリックします。



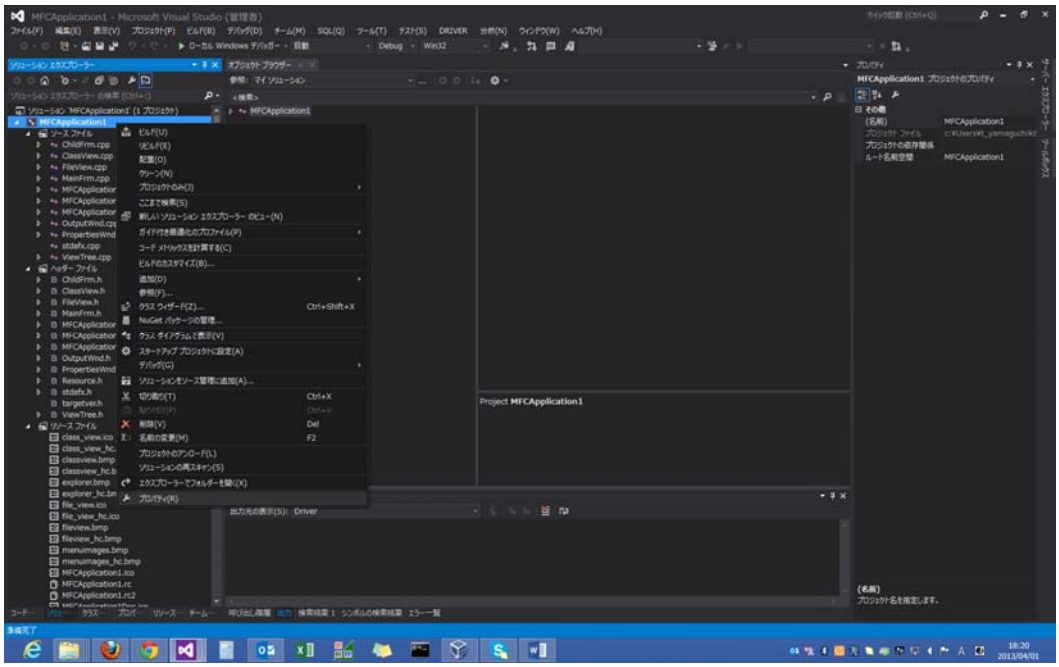
⑧ ダイアログのすべてのチェックを外し、「次へ」をクリックします。



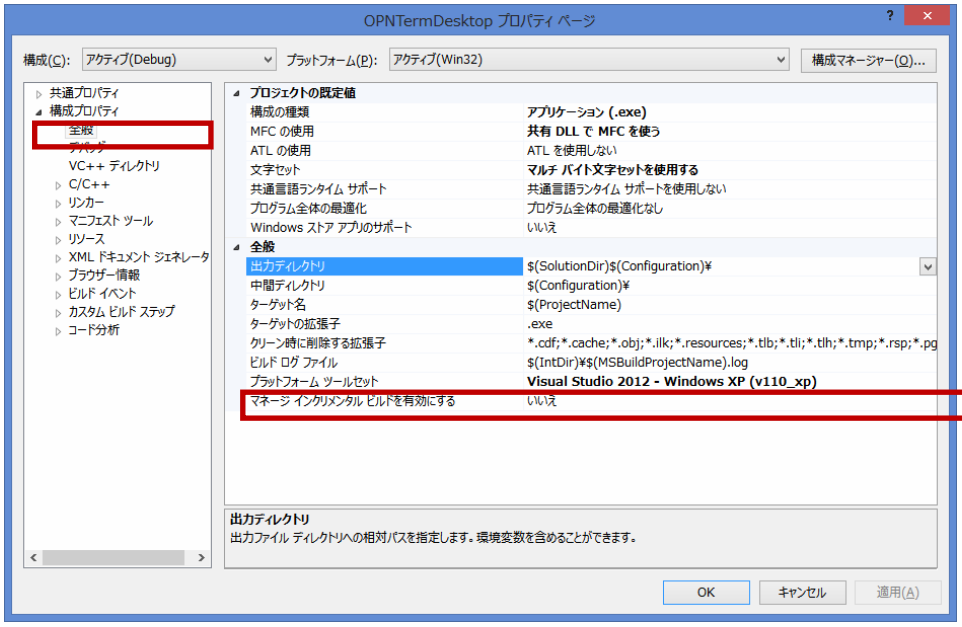
⑨ 「完了」をクリックします。



⑩ 作成したプロジェクトのプロパティを開きます。



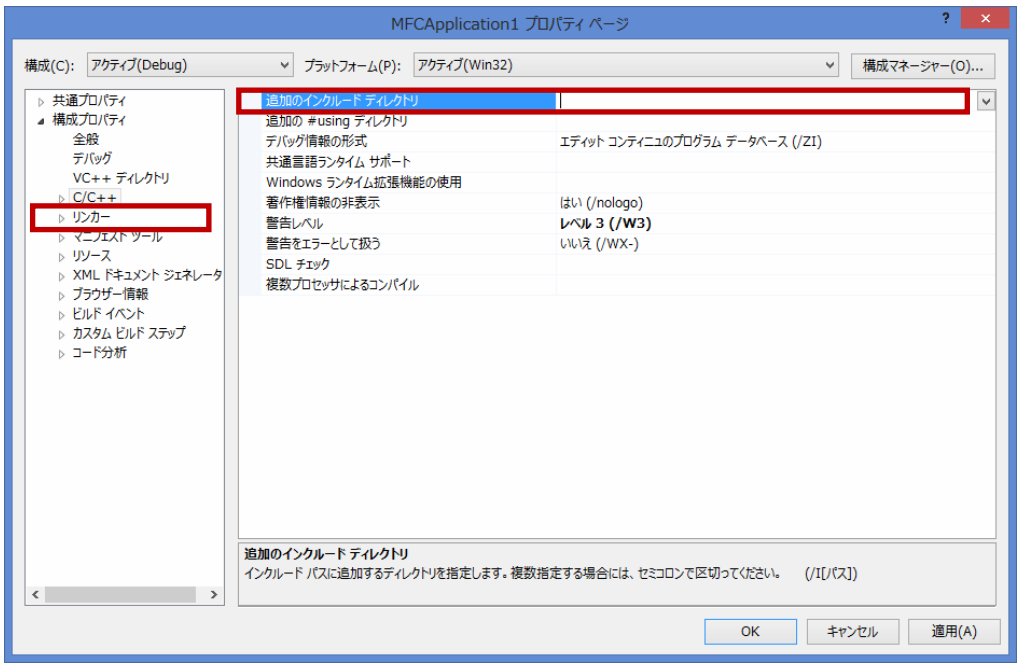
⑪ 「構成プロパティ」の「全般」を選択し、「プラットフォーム ツールセット」を「Windows XP」に変更します。



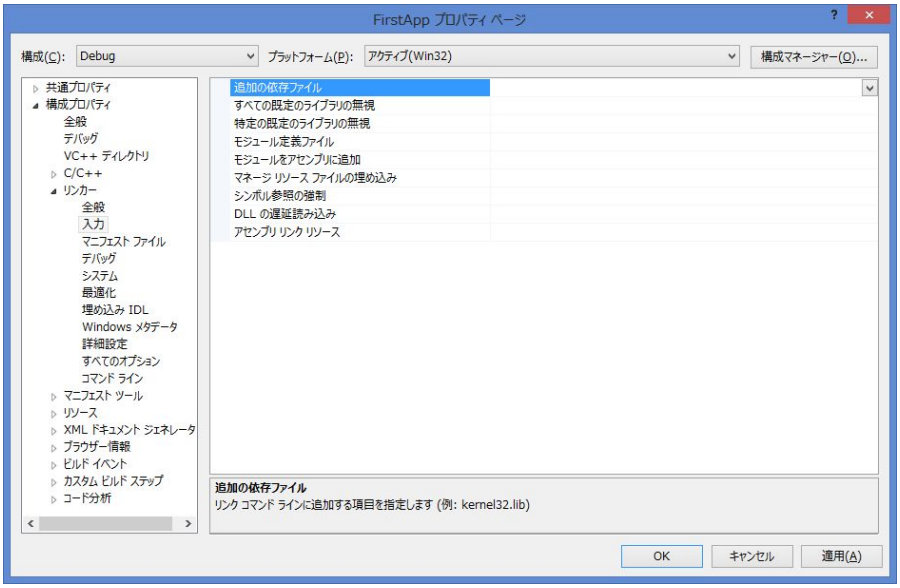
⑫ 「C/C++」を選択し、「追加のインクルードディレクトリ」を指定します。

Debug：¥¥OPNTerm¥sdk¥include¥Degub¥

Release：¥¥OPNTerm¥sdk¥include¥Release¥



⑬ 「リンカー」の「入力」を選択し、「追加の依存ファイル」に『OPNBT32.lib』を指定します。

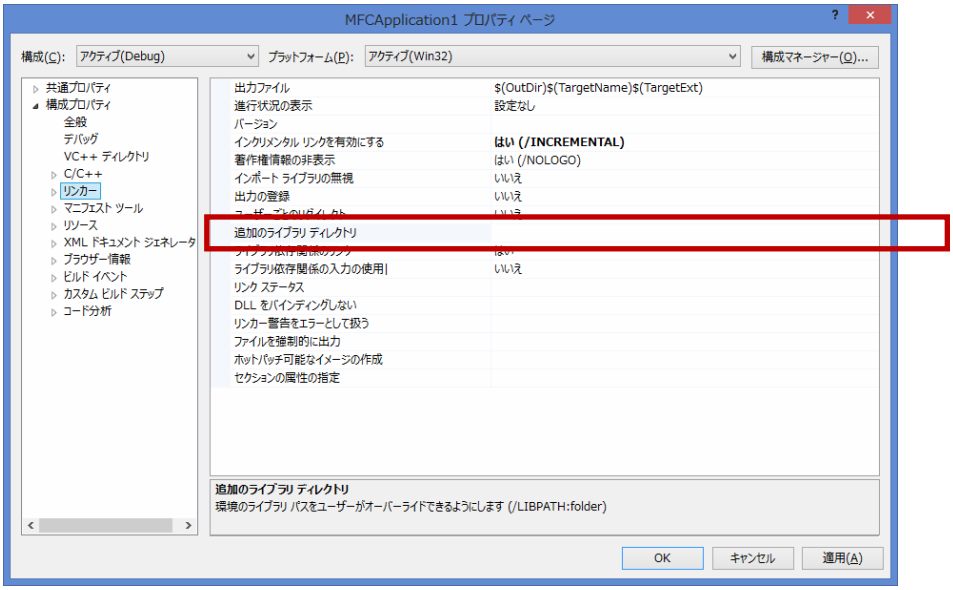


⑭ 「リンカー」の「全般」を選択し SDK ライブラリのディレクトリを指定し、「OK」をクリックします。

ライブラリディレクトリは、Debug と Release でディレクトリが異なります。

Debug：¥¥OPNTerm¥sdk¥lib¥Debug¥

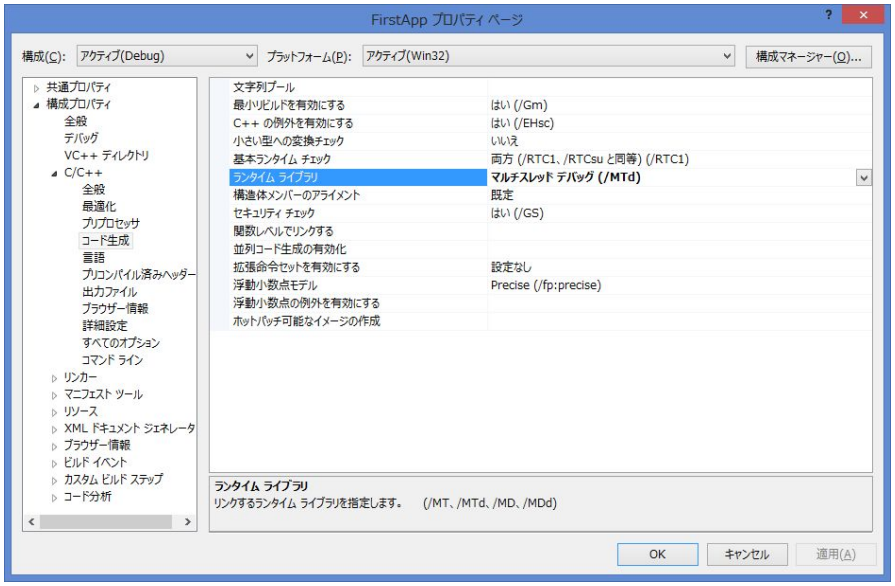
Release：¥¥OPNTerm¥sdk¥lib¥Relese¥



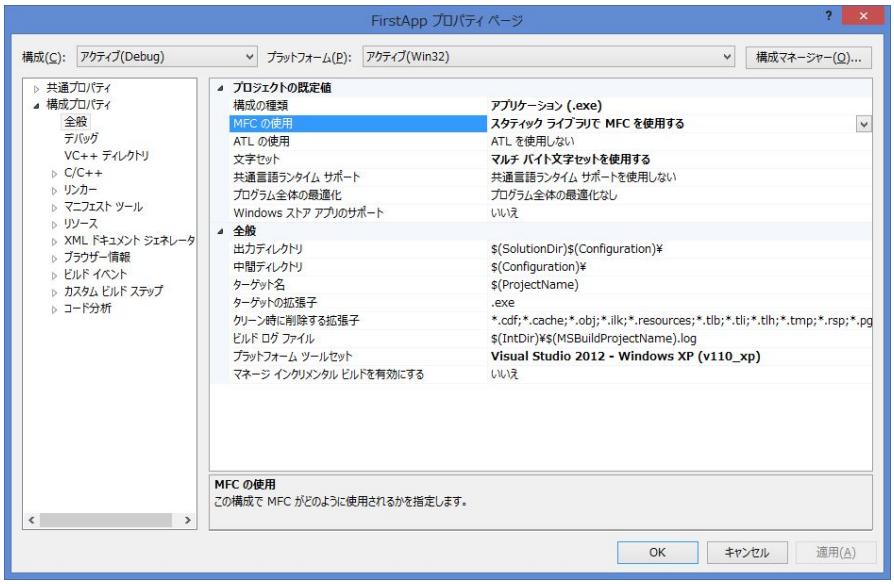
⑮ 「構成プロパティ」>「C/C++」>コード生成を選択し、ランタイムライブラリを以下のように設定してください。

Degug:マルチスレッドデバック（/MTd）

Release：マルチスレッド（/MT）



⑯ 「構成プロパティ」の「全般」を選択し、「MFC の使用」を『スタティックライブラリで MFC を使用する』を選択してください。



2. APIの利用

主な API の使用方法について、サンプルコードを用いて解説します。
なお、SDK の使用には、以下のヘッダーを読み込む必要があります・

```
#include "OPN2002iBluetoothService.h"
```

■ OPN2002iBluetoothServiceを生成・取得する

データコレクタとの Bluetooth 通信は、全て OPN2002iBluetoothService クラスを介して行います。

本サンプルでは、各メソッド中で、OPN2002iBluetoothService クラスのインスタンスを生成・取得をしています。

```
OPN2002iBluetoothService  &service = OPN2002iBluetoothService ::GetInstance();
```

■ デリゲートの設定

データコレクタから取得したデータを使用する場合は、通知を受け取るクラスで『IOPN2002iBluetoothServiceDelegate』を継承し、各 API に対応したデリゲートを実装してください。

CMainFrame.h

```
class CFirstAppDlg : public CDialogEx,IOPN2002iBluetoothServiceDelegate
 {
        void OnDidGetFirmware( OPNBluetoothService *pService, BYTE* pData );
…
```

CMainFrame.cpp

```
void CFirstAppDlg::OnBnClickedStartServerButton()
{
        OPN2002iBluetoothService &service = OPN2002iBluetoothService::GetInstance();
        service.StartServer();
}
```

■ MFC側がデータコレクタからの接続を待ち受ける

```
OPN2002iBluetoothService  &service = OPN2002iBluetoothService ::GetInstance();
service.StartServer();
```

データコレクタからの接続要求があった場合、接続を確立させます。

本サンプルでは、Connect（Server）ボタンが押下されたときに、本処理を実行しています。

■ MFC側がデータコレクタからの接続待ちを中断する

```
OPN2002iBluetoothService  &service = OPN2002iBluetoothService ::GetInstance();
service.StoptServer();
```

データコレクタとの接続待ちの状態を中断します。

■ 指定したデバイスにMFC側をクライアントとして接続します

```
OPN2002iBluetoothService::GetInstance().Connect
                                ( [デバイス名], [セキュア or インセキュア] );
```

OPN2002iBluetoothService#connect()メソッドを実行することで、データコレクタへの接続が開始されます。

本サンプルでは、Connect（Slave）ボタンが押下されたときにリストボックスで選択されたデバイスに接続を試みてみます。

［デバイス名］：接続先のデバイス名を指定してください。

［セキュア or インセキュア］：TRUE = セキュア / FALSE → インセキュア

■ 切断する

```
OPN2002iBluetoothService::GetInstance().Disconnect();
```

OPN2002iBluetoothService#Disconnect()メソッドを実行することで、データコレクタとの接続が解除されます。

本サンプルでは、DisConnect ボタンが押下されたときに、本処理を実行しています。

■ データコレクタとの通信で発生したイベントを処理する

データコレクタから取得したデータを使用する場合は、通知を受け取るクラスで『IOPN2002iBluetoothServiceDelegate』を継承し、各状態遷移に対応したデリゲートを実装してください。

CFirstAppDlg.h

```
class CFirstAppDlg : public CDialogEx,IOPN2002iBluetoothServiceDelegate
 {
…
```

CFirstAppDlg.cpp

```
// 接続されると呼び出されます
void CFirstAppDlg::OnDidConnectDevice
                (OPNBluetoothService *pService,OPNBluetoothDevice* pDevice ){
}
// 切断されると呼び出されます
void CFirstAppDlg:: OnDidDisconnectDevice
                (OPNBluetoothService *pService,OPNBluetoothDevice* pDevice) {
}
```

データ受信

```
void CFirstAppDlg::OnReceiveBarcode( OPNBluetoothService *pService, BYTE* pData )
{
…
```

データコレクタで、バーコードを読んだときなどに発生するイベントを処理します。
本サンプルでは、取得したバーコードデータをエディットボックスに追加しています。

OnDidDisconnectDevice: 切断

```
void CFirstAppDlg::OnDidDisconnectDevice( OPNBluetoothService
*pService, OPNBluetoothDevice* pDevice ) {
        MessageBox("デバイスと切断されました。", MB_OK );
}
```

データコレクタとの通信が切断されたとき発生する「切断」イベントを処理します。
本サンプルでは、MessageBox を使ってユーザに「切断されたこと」を通知しています。

OnDidConnectDevice: 接続完了

```
void CFirstAppDlg::OnDidConnectDevice( OPNBluetoothService *pService,
OPNBluetoothDevice* pDevice ) {
        MessageBox("デバイスと接続されました。", MB_OK );
}
```

データコレクタとの SPP 接続が完了したとき発生する「接続完了」イベントを処理します。
本サンプルでは、MessageBox を使ってユーザに「接続が完了したこと」を通知しています。

OnRequestPinCodeEvent: Pinコード要求

```
BOOL CFirstAppDlg::OnRequestPinCode(OPNBluetoothService *pService,
LPSTR lpszPinCode ) {
    CopyMemory( lpszPinCode, "1234", 4 );
    return TRUE;
}
```

データコレクタで PIN コードが必要になると OnRequestPinCodeEvent イベントが呼び出されます。送信する PIN コードを引数 lpszPinCode に設定し、TRUE を返します。返り値に FALSE を指定すると認証要求がキャンセルされ、Connect 関数から WOPN_AUTH_FAILED が戻ります。

connectFailed: 接続失敗

```
@Override
public void connectFailed() {
    Toast.makeText(this, "connection failed ", Toast.LENGTH_LONG).show();
}
```

MFC 側からデータコレクタへの接続が失敗した時に発生する「接続失敗」イベントを処理します。

本サンプルでは、MessageBox を使ってユーザに「接続が失敗したこと」を通知しています。

■ API標準のコマンドを実行する

```
// バージョンボタン押下
void CFirstAppDlg::OnBnClickedGetFirmwareVersionButton(){
        OPN2002iBluetoothService &service = OPN2002iBluetoothService::GetInstance();
        // デリゲートを設定
        service.SetDelegate(this);
        service.GetFirmwareVersion();
}
```

任意のコマンド送信とは違い、データコレクタに関するコマンドの中には、OPN2002iBluetoothService クラスのメソッドとして、提供されているものがあります。このメソッドは、非同期で実行されるため、非同期処理が完了したときに呼び出されるコールバックを指定する必要があります。コールバックは、どれも共通して IOPN2002iBluetoothServiceDelegate を実装したクラスのオブジェクトになっていて、非同期処理が完了したときには、OnDidGetFirmware (OPNBluetoothService *, BYTE*)メソッドが実行されます。

本サンプルでは、『GetFirmwareVersion』ボタンが押下されたときに、OPN2002iBluetoothService クラスの GetFirmwareVersion()メソッドを実行しています。コールバックには、IOPN2002iBluetoothServiceDelegate を実装している GetFirmwareVersionCallback クラスの匿名クラスを指定していて、処理が完了したときには、callbackExcecute()メソッドが実行されます。また、GetFirmwareVersion を実行する前に SetDelegate()メソッドでデリゲートを指定する必要があります。

■任意のコマンドを実行する

```
// Buzzer ON押下
void CFirstAppDlg::OnBnClickedGetFirmwareVersionButton(){
	OPN2002iBluetoothService &service = OPN2002iBluetoothService::GetInstance();
	service.PushCommand( "P3" );
	service.PushCommand( "B" , OPNApiBeepBuzzer, 10 );
	service.StartSendCommands();
}
```

OPN2002iBluetoothService クラスのメソッドとして、提供されていないメソッドを実行する場合は、OPN2002iBluetoothService#PushCommand()で送信するコマンドを追加し、OPN2002iBluetoothService#StartSendCommands() で送信をスタートさせます。

本サンプルでは、「BuzzerON」ボタンが押下されたときに、OPN2002iBluetoothService クラスの PushCommand ()、StartSendCommands() メソッドを実行しています。

利用できるコマンドについては、「3．コマンドリファレンス」を参照してください。

弊社製品名: OPN-2002i Series
OPN-3002i Series
発行管理番号: DM-130706
管理番号: SI13026

株式会社オプトエレクトロニクス

E-Mail:sales@opto.co.jp URL:http://www.opto.co.jp