

PM250TAH(2次元モデル) スキャナー SDK

コマンドリファレンス・マニュアル

Ver 1.3



ここに含まれる情報は、浜松東亜電機株式会社が独占的な権利を持ち、浜松東亜電機株式会社の書面による事前の許諾を得ずに全部もしくは一部を配布・複製または開示してはならないものとする。

目次

1 ドキュメントについて	5
1.1 改定履歴	5
2 序論	7
2.1 概要	7
2.1.1 ユーザーレイヤー	7
2.1.2 SDK	7
2.1.3 ハードウェアインタフェースレイヤー	7
3 定義、API関数、メッセージ	8
3.1 APIの構成	8
3.1.1 スキャナー関数	8
BcdOpen	8
BcdClose	8
BcdEnable	8
BcdIsEnabled	8
BcdSetHWND	9
BcdSetTerminator	9
BcdSetResultType	9
BcdGetResultType	9
BcdEnableBeep	10
BcdSetSuccessSoundFileName	10
BcdSetFailSoundFileName	10
BcdSetSuccessSound	10
BcdSetFailSound	10
BcdEnableVibrator	11
BcdSetVibratorInterval	11
BcdEnableLED	11
BcdEnablePrefix	11
BcdSetPrefix	11
BcdGetPrefix	12
BcdEnablePostfix	12
BcdSetPostfix	12
BcdGetPostfix	12
BcdSetTrigTimeout	12

BcdEanbleAutoScan.....	13
BcdSetAutoScanInterval	13
BcdSetConfig	13
BcdGetConfig	13
BcdGetBarTypeString	13
BcdGetEnableLength	14
BcdGetBarLength	14
BcdGetDefaultBarLength	14
BcdSetBarLength	14
BcdTransmitBarID	15
BcdGetBarID	15
BcdSetBarID	15
3.1.2 インジコン関数	16
BcdSetDefault	16
BcdEnableBarcode.....	16
BcdEnableALLBarcode.....	16
BcdIsBarcodeEnabled	16
BcdSetBarFlag.....	16
BcdGetBarFlagArrayPtr	17
BcdGetOCRFlag	17
BcdSetOCRFlag.....	17
BcdTrigTurnOn	17
BcdGetResult	17
BcdGetString.....	18
BcdGetBarType	18
BcdGetLength	18
BcdGetVersion	18
BcdGetDecoderVersion.....	18
BcdGetApiVersion	19
BcdGetScaDriverVersion	19
BcdSetSCANPCLK	19
BcdIs24MHzPCLK.....	19
BcdEnableCenterWindow.....	19
BcdSetDecodeMode	20
BcdSetDelayDecoding.....	20
BcdCreateDIBSection	20
BcdDestroyDIBSection	20
BcdGetCaptureImage.....	20
BcdImageStreamInit.....	21
BcdImageStreamStart.....	21
BcdImageStreamRead	21

BcdImageStreamStop.....	21
BcdAimerOn	21
BcdIlluminationOn	22
BcdSetScanLightsMode	22
BcdGetEngineConfig	22
BcdSetScanImageMode.....	22
BcdTriggerEnable	22
BcdIsTriggerEnabled	23
3.1.3 その他の関数	24
BcdGetModelName	24
BcdGetModelSerialNumber	24
3.2 関数と関数ポインタ	25
3.3 スキャナ-APIの使い方	27
3.4 構造体と定義の構成	28
3.4.1 スキャナ構造体と定義	28
3.4.2 インジック構造体と定義	35
3.5 メッセージ	37
(1) WM_SCANNED	37
(2) WM_IMAGE	37
4 BCD.NET.DLL	38
4.1 BCD.NET.dllとは.....	38
4.2 BCD.NET.dll の構成	38
4.3 BCD.NET.dll 使用サンプル	39

1 ドキュメントについて

このドキュメントは、PM250TAH(PM250 2次元モデル)のイメージスキャナーに関するSDK(コマンドリファレンスマニュアル)です。

OSイメージ版数と2DスキャナーSDKとの対応を下表に示します。

このドキュメントの内容は、下記の最新版数に対応しています。

(版数の確認方法は、PM250ユーザーマニュアルを参照して下さい。)

OSイメージ 版数	対応2DスキャナーSDK
25.04(Sep 01 2009)	PMx50_SDK_HHP5300_rev2.3(090831).zip
25.05(Nov 12 2009)	PMx50_SDK_HHP5300_rev2.8(091103).zip
25.06(Dec 15 2009)	PMx50_SDK_HHP5300_rev3.0(091208).zip
25.07(Dec 23 2009)	
25.08(Aug 30 2010)	PMx50_SDK_HHP5300_rev3.3(100811).zip

SDKは、本書以外に下記のものがあります。

- ・UnitAPI SDK(コマンドリファレンスマニュアル)
- ・PM250TA(1次元モデル)スキャナーSDK(コマンドリファレンスマニュアル)

1.1 改定履歴

バージョン	日付	説明	作者
1.0	2009年10月1日	1.日本語版 初版発行	浜松東亜電機株式会社
1.1	2009年12月1日	1. OS(V25.05)に対応 <ul style="list-style-type: none"> ・「BcdGetDefaultBarLength」追加 ・「BcdGetOCRFlag」追加 ・「BcdSetOCRFlag」追加 ・「BcdTriggerEnable」追加 ・「BcdIsTriggerEnabled」追加 ・「BcdEnableVibrate」削除 ・「BcdSetVibrateTime」削除 ・「BcdVibrateOperation」削除 ・「BcdSetScanKey」削除 ・「BcdSetCurBarType」削除 ・「BcdGetSettingStructure」削除 ・「BcdInit」削除 ・「BcdOEMInit」削除 ・「BcdSetExposureSettings」削除 ・「BcdGetResultType」バグFIX 	浜松東亜電機株式会社

		<ul style="list-style-type: none"> ・「BcdGetModelName」バグFIX ・スキャナ構成の「Prefix」「Postfix」設定のバグFIX ・バーコードのデフォルト設定値変更 <ul style="list-style-type: none"> - Code128 デフォルトバーコード長 - Code93/93i デフォルトバーコード長 - Ean8 デフォルト有効化 - Ean8 Send check character デフォルト有効化 - Ean13 Send check character デフォルト有効化 - QR デフォルト有効化 - UPCE 'send number system character デフォルト有効化 - CodablockF デフォルトバーコード長 ・「BcdCreateDIBSection」バグFIX ・「BcdGetModelSerialNumber」バグFIX ・「BcdSetScanLightsMode」説明修正 ・「BAR_FLAG」「OCR_FLAG」説明追記 ・「BCD.NET.dll」修正 	
1.2	2009年12月25日	1. OS(V25.07)に対応 <ul style="list-style-type: none"> ・「BcdEnableALLBarcode」追加 ・「BcdSetBarLength」復帰値変更 ・スキャナ構成にて間違ったバーコード長設定を出来なくした ・スキャナにてGSI Datamatrix をサポート 	浜松東亜電機株式会社
1.3	2010年9月1日	1. OS(V25.08)に対応 <ul style="list-style-type: none"> ・「BcdGetCaptureImage」関数説明追加 ・「ENGINECONFIG」構造体のコメント追加 ・バーコードバッファ長を260 3900に変更 	浜松東亜電機株式会社

2 序論

2.1 概要

このSDKは、PM250(2次元)のスキャナ制御を行うものです。

このSDKは、スキャナ制御のための各種定義、API、メッセージから構成されます。

EVC++ 4.0(Embedded Visual C++), C#, VB.NETからこのSDKを使うことによりPM250の2次元スキャナを制御できます。

ユーザーレイヤー
SDK
ハードウェアインタフェースレイヤー

2.1.1 ユーザーレイヤー

ユーザーレイヤーはSDKを通してEVC++, C#, VB.NET で作られるスキャナに関するユーザープログラムを意味します。

2.1.2 SDK

SDKは、PM250 2次元スキャナ制御のための各種定義、API、メッセージから構成されます。

(SDKの構成)

BCDApi.h : [BCDCore.dll]のヘッダファイル。各定義、構造体、API関数を含んでいます。

BCDCore.dll : ユーザーレイヤー側とリンクするためのDLLです。(直接PDAのWindowsの中にあります)

2.1.3 ハードウェアインタフェースレイヤー

ハードウェアインタフェースレイヤーはPM250 2次元スキャナ制御のためのハードウェア層です。

3 定義、API関数、メッセージ

3.1 APIの構成

3.1.1 スキャナー関数

BcdOpen

スキャナーをオープンし、ハンドルを取得します。

HANDLE BcdOpen(Void)

復帰値

HANDLE

パラメータ

Void

BcdClose

スキャナーをクローズし、ハンドルをクローズします。

Void BcdClose(Void)

復帰値

Void

パラメータ

Void

BcdEnable

スキャナーを有効化/無効化します。

Void BcdEnable(BOOL bEnable)

復帰値

Void

パラメータ

bEnable

この変数でスキャナーの有効/無効を設定します。

[TRUE]を設定するとスキャナーは有効となります。

BcdIsEnabled

スキャナー状態(有効/無効)を取得します。

BOOL BcdIsEnabled(Void)

復帰値

BOOL: スキャナーが有効の場合、[TRUE]が返ります。

パラメータ

Void

BcdSetHWND

スキャン結果を受け取るためのアプリケーションのWindowハンドルを設定します。

Void BcdSetHWND(HWND hWnd)

復帰値

Void

パラメータ

hWnd

アプリケーションWindowハンドル

BcdSetTerminator

スキャンが成功した時の終了コードを設定します。

Void BcdSetTerminator(BYTE cTerminator)

復帰値

Void

パラメータ

cTerminator

TERMINATOR_NONE: 無し

TERMINATOR_CRLF: [CR]コード+[LF]コード

TERMINATOR_CR: [CR]コード

TERMINATOR_LF: [LF]コード

TERMINATOR_SPACE: [SPACE]コード

TERMINATOR_TAB: [TAB]コード

TERMINATOR_STXETX: 先頭に[STX]コード、最後に[ETX]コード

BcdSetResultType

スキャンが成功した時のスキャン結果通知タイプを設定します。

Void BcdSetResultType(DWORD dwType)

復帰値

Void

パラメータ

dwType

RESULT_USERMSG: スキャナ使用アプリケーションのWindowハンドルにユーザーメッセージで伝える

RESULT_KBDMSG: キーボードメッセージで伝える

RESULT_COPYPASTE: コピー&ペーストで伝える

BcdGetResultType

スキャン成功時のスキャン結果通知タイプを取得します。

BYTE BcdGetResultType(Void)

復帰値

RESULT_USERMSG

RESULT_KBDMSG

RESULT_COPYPASTE

パラメータ

Void

BcdEnableBeep

.....

スキャン後のビープ音を有効にします。

Void BcdEnableBeep(BOOL bBeep)

復帰値

Void

パラメータ

bBeep:[TRUE]の場合、ビープ音を鳴動します。

BcdSetSuccessSoundFileName

.....

スキャン成功音のファイルを設定します。(Wavファイルは[Windows]ディレクトリに存在しなければならない。)

Void BcdSetSuccessSoundFileName(TCHAR *szSFileName)

復帰値

Void

パラメータ

szSFileName: スキャン成功時WAVファイル名

BcdSetFailSoundFileName

.....

スキャン失敗音のファイルを設定します。(Wavファイルは[Windows]ディレクトリに存在しなければならない。)

Void BcdSetFailSoundFileName(TCHAR * szSFileName)

復帰値

Void

パラメータ

szSFileName: スキャン失敗時WAVファイル名

BcdSetSuccessSound

.....

スキャン成功音のファイルを設定します。(Wavファイルは[Windows]ディレクトリに存在しなければならない。)

Void BcdSetSuccessSound(LPWSTR strSFileName)

復帰値

Void

パラメータ

strSFileName: スキャン成功時WAVファイル名

BcdSetFailSound

.....

スキャン失敗音のファイルを設定します。(Wavファイルは[Windows]ディレクトリに存在しなければならない。)

Void BcdSetFailSound(LPWSTR strFFFileName)

復帰値

Void

パラメータ

strFFFileName: スキャン失敗時WAVファイル名

BcdEnableVibrator

スキャン後のバイブレータ使用を決定します。

Void BcdEnableVibrator(BOOL bVibrate)

復帰値

Void

パラメータ

bVibrate:[TRUE]の場合、バイブレータが使用されます。

BcdSetVibratorInterval

スキャン成功後とスキャン失敗後のバイブレータ作動時間を設定します。

Void BcdSetVibratorInterval(DWORD dwSuccessInterval, DWORD dwFailInterval)

復帰値

Void

パラメータ

dwSuccessInterval: スキャン成功後のバイブレータ作動時間(単位:ミリ秒)

dwFailInterval: スキャン失敗後のバイブレータ作動時間(単位:ミリ秒)

(注意)

パラメータの最大値は、「3000」(3秒)です。それ以上の値を設定しても最大値で設定されます。

BcdEnableLED

スキャン後のLEDの使用を決定します。

Void BcdEnableLED(BOOL bLED)

復帰値

Void

パラメータ

bLED:[TRUE]の場合、スキャン後にLEDを点灯します。

BcdEnablePrefix

スキャン後のプレフィックスの使用を決定します。

Void BcdEnablePrefix(BOOL bPrefix)

復帰値

Void

パラメータ

bPrefix:[TRUE]の場合、スキャン後のプレフィックスを使用します。

BcdSetPrefix

プレフィックスを使用する場合、プレフィックス文字列を設定します。

Void BcdSetPrefix(TCHAR *szPrefix)

復帰値

Void

パラメータ

szPrefix: プレフィックス文字列

BcdGetPrefix

.....

プレフィックスを使用する場合、プレフィックス文字列を取得します。

Void BcdGetPrefix(TCHAR *szPrefix)

復帰値

Void

パラメータ

szPrefix: プレフィックス文字列を読み込む変数

BcdEnablePostfix

.....

スキャン後のホストフィックスの使用を決定します。

Void BcdEnablePostfix(BOOL bPostfix)

復帰値

Void

パラメータ

bPostfix: [TRUE]の場合、スキャン後にホストフィックスを使用します。

BcdSetPostfix

.....

ホストフィックスを使用する場合、ホストフィックス文字列を設定します。

Void BcdSetPostfix(TCHAR *szPostfix)

復帰値

Void

パラメータ

szPostfix: ホストフィックス文字列

BcdGetPostfix

.....

ホストフィックスを使用する場合、ホストフィックス文字列を取得します。

Void BcdGetPostfix(TCHAR *szPostfix)

復帰値

Void

パラメータ

szPostfix: ホストフィックス文字列を読み込む変数

BcdSetTrigTimeout

.....

スキャン開始とスキャン終了の間隔を設定します。

Void BcdSetTrigTimeout(DWORD dwSeconds)

復帰値

Void

パラメータ

dwSeconds: 間隔時間(単位: ミリ秒)

BcdEanbleAutoScan

.....

オートスキャンの使用を決定します。

Void BcdEanbleAutoScan(BOOL bAutoScan)

復帰値

Void

パラメータ

bAutoScan: [TRUE]なら、オートスキャンは有効です。

BcdSetAutoScanInterval

.....

オートスキャンの間隔を設定します。

Void BcdSetAutoScanInterval(DWORD dwInterval)

復帰値

Void

パラメータ

dwInterval: オートスキャン間隔 (単位: ミリ秒)

BcdSetConfig

.....

[SCD_CONFIG]の値を設定します。

Void BcdSetConfig(PBCD_CONFIG pbcdConfig)

復帰値

Void

パラメータ

pbcdConfig: [SCD_CONFIG]へのポインタ

BcdGetConfig

.....

[SCD_CONFIG]を取得します。

Void BcdGetConfig(PBCD_CONFIG pbcdConfig)

復帰値

Void

パラメータ

pbcdConfig: [SCD_CONFIG]へのポインタ

BcdGetBarTypeString

.....

バーコードタイプからバーコードタイプの文字列を取得します。

LPWSTR BcdGetBarTypeString(BAR_TYPE barType)

復帰値

LPWSTR: バーコードタイプ文字列

パラメータ

barType: バーコードタイプ (=0 ~ 46が有効です。)

BcdGetEnableLength

.....

指定されたバーコードタイプがバーコード長を返すか否かを識別します。

BOOL BcdGetEnableLength(BAR_TYPE barType)

復帰値

BOOL: [TRUE]なら指定されたバーコードタイプはバーコード長を返します。

パラメータ

barType: バーコードタイプ (=0 ~ 46が有効です。)

BcdGetBarLength

.....

指定されたバーコードタイプの最大長と最小長を取得します。

Void BcdGetBarLength(BAR_TYPE barType, int* puMin, int* puMax)

復帰値

Void

パラメータ

barType: バーコードタイプ (=0 ~ 46が有効です。)

puMin: 最小長

puMax: 最大長

BcdGetDefaultBarLength

.....

指定されたバーコードタイプのデフォルトの最大長と最小長を取得します。

Void BcdGetDefaultBarLength(BAR_TYPE barType, int* puMin, int* puMax)

復帰値

Void

パラメータ

barType: バーコードタイプ (=0 ~ 46が有効です。)

puMin: デフォルト最小長

puMax: デフォルト最大長

BcdSetBarLength

.....

指定されたバーコードタイプの最大長と最小長を設定します。

BOOL BcdSetBarLength(BAR_TYPE barType, int uMin, int uMax)

復帰値

BOOL: [TRUE]の場合、正常終了。

パラメータ

barType: バーコードタイプ (=0 ~ 46が有効です。)

uMin: 最小長設定値

uMax: 最大長設定値

(注意)

uMin値およびuMax値は、BcdGetDefaultBarLength関数にて取得できるデフォルト最小長～デフォルト最大長の範囲内で指定して下さい。

BcdTransmitBarID

.....

スキャン後のバーコードIDの送信を有効にします。

Void BcdTransmitBarID(BOOL bTransmit)

復帰値

Void

パラメータ

bTransmit:[TRUE]ならバーコードIDを送信します。(バーコードIDがバーコードデータの前に付加されます。)

BcdGetBarID

.....

指定されたバーコードタイプのIDを取得します。

BYTE BcdGetBarID(BAR_TYPE barType)

復帰値

BYTE : 取得したバーコードID

パラメータ

barType:バーコードタイプ (=0 ~ 46が有効です。)

BcdSetBarID

.....

指定したバーコードタイプにIDを設定します。

Void BcdSetBarID(BAR_TYPE barType, BYTE barID)

復帰値

Void

パラメータ

barType:バーコードタイプ (=0 ~ 46が有効です。)

barID:バーコードID設定値

3.1.2 エンジン関数

BcdSetDefault

.....

工場出荷モードにリセットします。

Void BcdSetDefault(void)

復帰値

Void

パラメータ

Void

BcdEnableBarcode

.....

指定されたバーコードタイプを有効化/無効化します。

Void BcdEnableBarcode(BAR_TYPE barType, BOOL bEnable)

復帰値

Void

パラメータ

barType: バーコードタイプ (=0 ~ 46が有効です。)

bEnable: [TRUE]の場合、有効化

BcdEnableALLBarcode

.....

全バーコードタイプを有効化/無効化します。

BOOL BcdEnableBarcode(BOOL bEnable)

復帰値

BOOL: [TURE]の場合正常終了。(4 ~ 7秒後に復帰します。)

パラメータ

bEnable: [TRUE]の場合、有効化

BcdIsBarcodeEnabled

.....

指定されたバーコードタイプの有効/無効状態を取得します。

BOOL BcdIsBarcodeEnabled(BAR_TYPE barType)

復帰値

BOOL

パラメータ

barType: バーコードタイプ (=0 ~ 46が有効です。)

BcdSetBarFlag

.....

指定されたバーコードタイプの詳細構成情報を設定します。

Void BcdSetBarFlag(BAR_TYPE barType, int nIndex, BOOL bEnable)

復帰値

Void

パラメータ

barType: バーコードタイプ (=0 ~ 46が有効です。)

nIndex: 詳細設定のインデックス

bEnable: [TRUE]の場合、そのインデックスの詳細設定は有効化

BcdGetBarFlagArrayPtr

.....

指定されたバーコードタイプの詳細構成情報を取得します。

PBAR_FLAG BcdGetBarFlagArrayPtr(BAR_TYPE barType)

復帰値

PBAR_FLAG

パラメータ

barType: バーコードタイプ (=0 ~ 46が有効です。)

BcdGetOCRFlag

.....

OCRバーコードタイプの詳細設定情報を取得します。

OCR_FLAG BcdGetOCRFlag(void)

復帰値

OCR_FLAG: OCRコードの詳細設定情報

パラメータ

Void

BcdSetOCRFlag

.....

OCRバーコードタイプの詳細設定情報を設定します。

Void BcdSetOCRFlag(OCR_FLAG OCRFlag)

復帰値

Void

パラメータ

OCR_FLAG: OCRコードの詳細設定情報

BcdTrigTurnOn

.....

スキャンを開始/終了します。

Void BcdTrigTurnOn(BOOL bTrigger)

復帰値

Void

パラメータ

bTrigger: [TRUE]の場合、スキャンを開始します。

BcdGetResult

.....

[WM_SCANNED]メッセージを取得した時に、この関数にて[SCAN_RESULT]構造体を取得します。

PSCAN_RESULT BcdGetResult(void)

復帰値

PSCAN_RESULT

パラメータ

Void

BcdGetString

.....

現在の[SCAN_RESULT]からハーフコード値を取得します。

LPWSTR BcdGetString(void)

復帰値

LPWSTR

パラメータ

Void

BcdGetBarType

.....

現在の[SCAN_RESULT]からハーフコードタイプを取得します。

BAR_TYPE BcdGetBarType(void)

復帰値

BAR_TYPE

パラメータ

Void

BcdGetLength

.....

現在の[SCAN_RESULT]からハーフコードデータの長さを取得します。

UINT BcdGetBarType(void)

復帰値

UINT

パラメータ

Void

BcdGetVersion

.....

スキャナーのデコーダーのバージョンを取得します。

Void BcdGetVersion(TCHAR *szVersion)

復帰値

Void

パラメータ

szVersion: スキャナーデコーダーのバージョン

BcdGetDecoderVersion

.....

スキャナーのデコーダーのバージョンを取得します。(BcdGetVersion()と同じ関数)

Void BcdGetDecoderVersion(TCHAR *szVersion)

復帰値

Void

パラメータ

szVersion: スキャナーデコーダーのバージョン

BcdGetApiVersion

.....

スキャナーのデコーダーのAPIバージョンを取得します。

Void BcdGetApiVersion(TCHAR *szVersion)

復帰値

Void

パラメータ

szVersion: スキャナーデコーダーのAPIバージョン

BcdGetScaDriverVersion

.....

スキャナーのドライバのバージョンを取得します。

Void BcdGetScaDriverVersion(TCHAR *szVersion)

復帰値

Void

パラメータ

szVersion: スキャナードライバのバージョン

BcdSetSCANPCLK

.....

スキャナーの[Pclock]を設定します。

Void BcdSetSCANPCLK(DWORD PCLK)

復帰値

Void

パラメータ

PCLK: [0]の場合[Pclock]は12MHz、[1]以上の場合[Pclock]は24MHz

BcdIs24MHzPCLK

.....

スキャナーの[Pclock]を取得します。

This function is get scanner's Pclock.

BOOL BcdIs24MHzPCLK(Void)

復帰値

BOOL: [TRUE]の場合、[Pclock]は24MHz、[FALSE]の場合[Pclock]は12MHz.

パラメータ

Void

BcdEnableCenterWindow

.....

[center window]機能の使用を決定します。

[center window]機能を使用する場合はスキャン時にバーコード位置が目標領域の中心でなければならない。 Void

BcdEnableCenterWindow(BOOL bUse)

復帰値

Void

パラメータ

bUse: [TRUE]の場合、[Center window]機能を使用します。

BcdSetDecodeMode

.....

スキャナーのデコードモードを決定します。

Void BcdSetDecodeMode(BYTE btMode)

復帰値

Void

パラメータ

btMode

STANDARD : 標準のモード

ADVANCED_LINEAR : 1次元バーコードに適したモード

QUICK_OMNI : 標準より早いモード

BcdSetDelayDecoding

.....

遅延時間後にデコードする場合の遅延時間を設定します。

Void BcdSetDelayDecoding(DWORD btDelay)

復帰値

Void

パラメータ

btDelay: 遅延時間 (単位: ミリ秒)

BcdCreateDIBSection

.....

イメージータを書込むために[BmpFormat]からビットマップハンドルを作成します。

HBITMAP BcdCreateDIBSection(BmpFormat * pBmp)

復帰値

HBITMAP : Bitmap handle.

パラメータ

pBmp: [BmpFormat]のポインタ

BcdDestroyDIBSection

.....

ビットマップハンドルを削除します。

Void BcdDestroyDIBSection(HBITMAP hBitmap)

復帰値

Void

パラメータ

hBitmap: ビットマップハンドル

この関数は[BcdCreateDIBSection]関数とペアで使用しなければならない。

BcdGetCaptureImage

.....

イメージータを取得する。

BOOL BcdGetCaptureImage(PBYTE plmageBuffer, DWORD * plmageSize)

復帰値

BOOL : [TRUE]の場合、正常復帰

パラメータ

plmageBuffer: イメージータバッファへのポインタ

plmageSize: イメージータ長へのポインタ

BcdImageStreamInit

.....

イメージビューの準備を行う。

BOOL BcdImageStreamInit(Void)

復帰値

BOOL : [TRUE]の場合、正常復帰

パラメータ

Void

BcdImageStreamStart

.....

イメージビューを開始します。

BOOL BcdImageStreamStart(Void)

復帰値

BOOL : [TRUE]の場合、正常復帰

パラメータ

Void

BcdImageStreamRead

.....

連続的なイメージビューを維持します。

BOOL BcdImageStreamRead(unsigned char * pImageBuffer, DWORD * pdwSize)

復帰値

BOOL : [TRUE]の場合、正常復帰

パラメータ

pImageBuffer: イメージデータのバッファへのポインタ
pdwSize: イメージのサイズ

BcdImageStreamStop

.....

イメージビューを停止します。

BOOL BcdImageStreamStop(Void)

復帰値

BOOL : [TRUE]の場合、正常復帰

パラメータ

Void

BcdAimerOn

.....

照準の使用を決定します。

BOOL BcdAimerOn(BOOL enable)

復帰値

BOOL : [TRUE]の場合、正常復帰

パラメータ

Enable: [TRUE]の場合、照準をONします。[FALSE]の場合OFFします。

BcdIlluminationOn

照明の使用を決定します。

BOOL BcdIlluminationOn(BOOL enable)

復帰値

BOOL : [TRUE]の場合、正常復帰

パラメータ

Enable : [TRUE]の場合、照明をONします。[FALSE]の場合OFFします。

BcdSetScanLightsMode

イメージキャプチャあるいはプレビューのとき照準と照明の使用を決定します。(この関数は[Image Mode]時に使用できます。)

BOOL BcdSetScanLightsMode(SCANLIGHTSMODE mode)

復帰値

BOOL : [TRUE]の場合、正常復帰

パラメータ

mode

SCANLIGHTSMODE_OFF= 0,	: 照準、照明ともにOFF
SCANLIGHTSMODE_ILLUM_ONLY_ON,	: 照明のみON
SCANLIGHTSMODE_AIMER_ONLY_ON,	: 照準のみON
SCANLIGHTSMODE_ON,	: 照準、照明ともにON

BcdGetEngineConfig

スキャンエンジンの構成を取得します。

BOOL BcdGetEngineConfig(ENGINECONFIG * pEngineConfig)

復帰値

BOOL : [TRUE]の場合、正常復帰

パラメータ

pEngineConfig : [ENGINECONFIG]へのポインタ。([ENGINECONFIG]参照)

BcdSetScanImageMode

スキャナーモードの使用を決定します。(スキャンモードあるいはイメージモード)

Void BcdSetScanImageMode(int nMode)

復帰値

Void

パラメータ

nMode

MODE_SCAN	=0x00	: スキャンモード
MODE_IMAGE	=0x01	: イメージモード

BcdTriggerEnable

スキャントリガーを有効化/無効化します。

Void BcdTriggerEnable (BOOL bTrigger)

復帰値

Void

パラメータ

bTrigger : [TURE]の場合、スキャントリガーが有効です。

BcdIsTriggerEnabled

.....

スキャントリガー状態を取得します。

BOOL BcdIsTriggerEnabled (Void)

復帰値

BOOL:[TRUE]の場合、スキャントリガーは有効です。

パラメータ

Void

3.1.3 その他の関数

BcdGetModelName

.....

デバイスモデル名を取得します。

LPWSTR BcdGetModelName(Void)

復帰値

LPWSTR : Model Name.

パラメータ

Void

BcdGetModelSerialNumber

.....

シリアル番号を取得します。

LPWSTR BcdGetModelSerialNumber(Void)

復帰値

LPWSTR : シリアル番号

パラメータ

Void

3.2 関数と関数ポインタ

各関数は関数ポインタを持っています。

下表は各関数と関数ポインタの説明です。

BcdOpen	typedef HANDLE (WINAPI *PFN_BcdOpen)(void);
BcdClose	typedef void (WINAPI *PFN_BcdClose)(void);
BcdEnable	typedef void (WINAPI *PFN_BcdEnable)(BOOL);
BcdIsEnabled	typedef BOOL (WINAPI *PFN_BcdIsEnabled)(void);
BcdSetHWND	typedef void (WINAPI *PFN_BcdSetHWND)(HWND);
BcdSetTerminator	typedef void (WINAPI *PFN_BcdSetTerminator)(BYTE);
BcdSetResultType	typedef void (WINAPI *PFN_BcdSetResultType)(DWORD);
BcdGetResultType	typedef BYTE (WINAPI *PFN_BcdGetResultType)(void);
BcdEnableBeep	typedef void (WINAPI *PFN_BcdEnableBeep)(BOOL);
BcdSetSuccessSoundFileName	typedef void (WINAPI *PFN_BcdSetSuccessSoundFileName)(TCHAR*);
BcdSetFailSoundFileName	typedef void (WINAPI *PFN_BcdSetFailSoundFileName)(TCHAR*);
BcdSetSuccessSound	typedef void (WINAPI *PFN_BcdSetSuccessSound)(LPWSTR);
BcdSetFailSound	typedef void (WINAPI *PFN_BcdSetFailSound)(LPWSTR);
BcdEnableVibrator	typedef void (WINAPI *PFN_BcdEnableVibrator)(BOOL);
BcdSetVibratorInterval	typedef void (WINAPI *PFN_BcdSetVibratorInterval)(DWORD, DWORD);
BcdEnableLED	typedef void (WINAPI *PFN_BcdEnableLED)(BOOL);
BcdEnablePrefix	typedef void (WINAPI *PFN_BcdEnablePrefix)(BOOL);
BcdSetPrefix	typedef BOOL (WINAPI *PFN_BcdSetPrefix)(TCHAR*);
BcdGetPrefix	typedef void (WINAPI *PFN_BcdGetPrefix)(TCHAR*);
BcdEnablePostfix	typedef void (WINAPI *PFN_BcdEnablePostfix)(BOOL);
BcdSetPostfix	typedef BOOL (WINAPI *PFN_BcdSetPostfix)(TCHAR*);
BcdGetPostfix	typedef void (WINAPI *PFN_BcdGetPostfix)(TCHAR*);
BcdSetTrigTimeout	typedef void (WINAPI *PFN_BcdSetTrigTimeout)(DWORD);
BcdEanbleAutoScan	typedef void (WINAPI *PFN_BcdEanbleAutoScan)(BOOL);
BcdSetAutoScanInterval	typedef void (WINAPI *PFN_BcdSetAutoScanInterval)(DWORD);
BcdSetConfig	typedef void (WINAPI *PFN_BcdSetConfig)(PBCD_CONFIG);
BcdGetConfig	typedef void (WINAPI *PFN_BcdGetConfig)(PBCD_CONFIG);
BcdGetBarTypeString	typedef LPWSTR (WINAPI *PFN_BcdGetBarTypeString)(BAR_TYPE);
BcdGetEnableLength	typedef BOOL (WINAPI *PFN_BcdGetEnableLength)(BAR_TYPE);
BcdGetBarLength	typedef void (WINAPI *PFN_BcdGetBarLength)(BAR_TYPE, int*, int*);
BcdGetDefaultBarLength	typedef void (WINAPI *PFN_BcdGetDefaultBarLength)(BAR_TYPE, int*, int*);
BcdSetBarLength	typedef BOOL (WINAPI *PFN_BcdSetBarLength)(BAR_TYPE, int, int);
BcdTransmitBarID	typedef void (WINAPI *PFN_BcdTransmitBarID)(BOOL);
BcdGetBarID	typedef BYTE (WINAPI *PFN_BcdGetBarID)(BAR_TYPE);
BcdSetBarID	typedef void (WINAPI *PFN_BcdSetBarID)(BAR_TYPE, BYTE);
BcdSetDefault	typedef void (WINAPI *PFN_BcdSetDefault)(void);
BcdEnableBarcode	typedef void (WINAPI *PFN_BcdEnableBarcode)(BAR_TYPE, BOOL);
BcdEnableALLBarcode	typedef BOOL (WINAPI *PFN_BcdEnableALLBarcode)(BOOL);
BcdIsBarcodeEnabled	typedef BOOL (WINAPI *PFN_BcdIsBarcodeEnabled)(BAR_TYPE);
BcdSetBarFlag	typedef void (WINAPI *PFN_BcdSetBarFlag)(BAR_TYPE, int, BOOL);
BcdGetBarFlagArrayPtr	typedef PBAR_FLAG (WINAPI *PFN_BcdGetBarFlagArrayPtr)(BAR_TYPE);
BcdGetOCRFlag	typedef OCR_FLAG (WINAPI *PFN_BcdGetOCRFlag)(void);
BcdSetOCRFlag	typedef void (WINAPI *PFN_BcdSetOCRFlag)(OCR_FLAG);
BcdTrigTurnOn	typedef void (WINAPI *PFN_BcdTrigTurnOn)(BOOL);
BcdGetResult	typedef PSCAN_RESULT (WINAPI *PFN_BcdGetResult)(void);
BcdGetString	typedef LPWSTR (WINAPI *PFN_BcdGetString)(void);
BcdGetBarType	typedef BAR_TYPE (WINAPI *PFN_BcdGetBarType)(void);
BcdGetLength	typedef UINT (WINAPI *PFN_BcdGetLength)(void);
BcdGetVersion	typedef void (WINAPI *PFN_BcdGetVersion)(TCHAR*);

BcdGetDecoderVersion	typedef void (WINAPI *PFN_BcdGetDecoderVersion)(TCHAR*);
BcdGetApiVersion	typedef void (WINAPI *PFN_BcdGetApiVersion)(TCHAR*);
BcdGetScaDriverVersion	typedef void (WINAPI *PFN_BcdGetScaDriverVersion)(TCHAR*);
BcdSetSCANPCLK	typedef void (WINAPI *PFN_BcdSetSCANPCLK)(DWORD);
BcdIs24MHzPCLK	typedef BOOL (WINAPI *PFN_BcdIs24MHzPCLK)(void);
BcdEnableCenterWindow	typedef void (WINAPI *PFN_BcdEnableCenterWindow)(BOOL);
BcdSetDecodeMode	typedef void (WINAPI *PFN_BcdSetDecodeMode)(BYTE);
BcdSetDelayDecoding	typedef void (WINAPI *PFN_BcdSetDelayDecoding)(DWORD);
BcdCreateDIBSection	typedef HBITMAP (WINAPI *PFN_BcdCreateDIBSection)(BmpFormat*);
BcdDestroyDIBSection	typedef void (WINAPI *PFN_BcdDestroyDIBSection)(HBITMAP);
BcdGetCaptureImage	typedef BOOL (WINAPI *PFN_BcdGetCaptureImage)(PBYTE, DWORD*);
BcdImageStreamInit	typedef BOOL (WINAPI *PFN_BcdImageStreamInit)(void);
BcdImageStreamStart	typedef BOOL (WINAPI *PFN_BcdImageStreamStart)(void);
BcdImageStreamRead	typedef BOOL (WINAPI *PFN_BcdImageStreamRead)(unsigned char*, DWORD*);
BcdImageStreamStop	typedef BOOL (WINAPI *PFN_BcdImageStreamStop)(void);
BcdAimerOn	typedef BOOL (WINAPI *PFN_BcdAimerOn)(BOOL);
BcdIlluminationOn	typedef BOOL (WINAPI *PFN_BcdIlluminationOn)(BOOL);
BcdSetScanLightsMode	typedef BOOL (WINAPI *PFN_BcdSetScanLightsMode)(SCANLIGHTSMODE);
BcdGetEngineConfig	typedef BOOL (WINAPI *PFN_BcdGetEngineConfig)(ENGINECONFIG*);
BcdSetScanImageMode	typedef void (WINAPI *PFN_BcdSetScanImageMode)(int);
BcdTriggerEnable	typedef void (WINAPI *PFN_BcdTriggerEnable)(BOOL);
BcdIsTriggerEnabled	typedef BOOL (WINAPI *PFN_BcdIsTriggerEnabled)(void);
BcdGetModelName	typedef LPWSTR (WINAPI *PFN_BcdGetModelName)(void);
BcdGetModelSerialNumber	typedef LPWSTR (WINAPI *PFN_BcdGetModelSerialNumber)(void);

3.3 スキャナ-APIの使い方

アプリケーションは関数ポインタを使ってプログラミングできます。

下記コードは関数ポインタの使い方のサンプルです。

```
//      Include scanner's heder file.
#include "../BCDapi.h"

//      Load library and get function's process address.
HINSTANCE hInstance = NULL;
PFN_BcdEnable pBcdEnable = NULL;
hInstance=LoadLibrary(_T("BcdCore.dll"));
if(hInstance != NULL)
    pBcdEnable = (PFN_BcdEnable)GetProcAddress(hInstance, L"BcdEnable");

// IF you want use to scanner.
pBcdEnable(TRUE);
FreeLibrary(hInstance);
```

3.4 構造体と定義の構成

下記の記述がSDKを使う上での構造体と定義です。

3.4.1 スキャナ-構造体と定義

About of scanner configurations.

```
//      define for user message
#define WM_SCANNED    WM_USER+0x7777  // Transmit message after scan.
#define WM_IMAGE      WM_USER+0x7778  // Transmit message after capture.

//      define for terminate : Terminate code is add the last character of scan value.
#define TERMINATOR_NONE    0x00  //      None.
#define TERMINATOR_CRLF    0x01  //      CR + LF
#define TERMINATOR_CR      0x02  //      CR
#define TERMINATOR_LF      0x03  //      LF
#define TERMINATOR_SPACE   0x04  //      SPACE
#define TERMINATOR_TAB      0x05  //      TAB
#define TERMINATOR_STXETX   0x06  //      STX + ETX

//      define for scan result type : How to transmit after scan.
#define RESULT_USERMSG      0x00  //      Each application get the WM_SCANNED.
#define RESULT_KBDMSG      0x01  //      Use to KBDMSG.
#define RESULT_COPYPASTE   0x02  //      Use to COPYPASTE.
#define RESULT_EVENT        0x03  //      NOT USE.
#define RESULT_CALLBACK     0x04  //      NOT USE.

//      scan status : WM_SCANNED message s LPARAM value.
typedef enum _STATUS_BCD
{
    ERROR_NO_ERROR = 0,    //      Scan success.
    ERROR_NO_READ,        //      Scan fail.
} ERROR_BSCANNER;

//      scanner order : CODE_ID of scan driver IOCTL.
typedef enum
```

```
{  
    ORDER_DEBUG00 = 0,  
    ORDER_DEBUG01,  
    ORDER_DEBUG02,  
    ORDER_PARAM_INIT,  
    ORDER_HWTRIGGERON,  
    ORDER_HWTRIGGEROFF,  
    ORDER_SETHWND,  
    ORDER_SCANNER_ENABLE,  
    ORDER_SCANNER_DISABLE,  
    ORDER_BEEP_ENABLE,  
    ORDER_BEEP_DISABLE,  
    ORDER_VIBRATE_ENABLE,  
    ORDER_VIBRATE_DISABLE,  
    ORDER_LED_ENABLE,  
    ORDER_LED_DISABLE,  
    ORDER_GETBARTYPE,  
    ORDER_GETBARSTRING,  
    ORDER_GETRESULT,  
    ORDER_AUTOSCAN_ON,  
    ORDER_AUTOSCAN_OFF,  
    ORDER_IENABLED,  
    ORDER_TRIGGER_TIMEOUT,  
    ORDER_SCANNER_DEFAULT,  
    ORDER_SCANNER_GETPARAM,  
    ORDER_SCANNER_AUTOSCANINTERVAL,  
    ORDER_SETRESULTTYPE,  
    ORDER_SETTERMINATOR,  
    ORDER_SCANNER_GETTYPESTATUS,  
    ORDER_SETCONFIG,  
    ORDER_GETCONFIG,  
    ORDER_ENABLEPREFIX,  
    ORDER_ENABLEPOSTFIX,  
    ORDER_SETPREFIX,  
    ORDER_SETPOSTFIX,  
    ORDER_GETPREFIX,  
    ORDER_GETPOSTFIX,  
}
```

ORDER_SETSCANKEY,
ORDER_SETSFILENAME,
ORDER_SETFFILENAME,
ORDER_SETVIBRATETIME_SUCCESS,
ORDER_SETVIBRATETIME_FAIL,
ORDER_GETSETTING,
ORDER_BARTYPE_DISABLE,
ORDER_BARTYPE_ENABLE,
ORDER_SET_CURBARCODE,
ORDER_SET_BARID,
ORDER_SET_BARLENGTH_MAX,
ORDER_SET_BARLENGTH_MIN,
ORDER_SET_BARFLAG_INDEX,
ORDER_SET_BARFLAG_BFLAG,
ORDER_GETBARFLAG,
ORDER_DETAIL_SET,
ORDER_VIBRATEOPERATION,
ORDER_TRANSMIT_ID,
ORDER_GETSCANNERVERSION,
ORDER_ISSCANPCLK,
ORDER_SETSCANPCLK,
ORDER_GETCAPTURE,
ORDER_STOPSCAN,
ORDER_GET_DECODERREVISION,
ORDER_GET_APIREVISION,
ORDER_CENTERWINDOW_EANBLE,
ORDER_CENTERWINDOW_DISABLE,
ORDER_SET_DECODEMODE,
ORDER_SET_DELAYDECODING,
ORDER_IS_TURNON,
ORDER_SWITCH_POWER,
ORDER_OEM_INIT,
ORDER_IMAGESTREAM_INIT,
ORDER_IMAGESTREAM_START,
ORDER_IMAGESTREAM_READ,
ORDER_IMAGESTREAM_STOP,
ORDER_SETEXPOSURESETTINGS,

```
ORDER_AIMER_ON,  
ORDER_AIMER_OFF,  
ORDER_ILLUMINATION_ON,  
ORDER_ILLUMINATION_OFF,  
ORDER_SETSCANLIGHTSMODE,  
ORDER_GETENGINECONFIG,  
ORDER_GET_SCANDRVREVISION,  
ORDER_SETSCANIMAGEMODE,  
ORDER_GETDEFAULTSETTING,  
ORDER_GETOCRFLAG,  
ORDER_SETOCRFLAG,  
ORDER_SET_TRIGGERENABLE,  
ORDER_SET_TRIGGERDISABLE,  
ORDER_SET_ISTRIGGERENABLE,  
} Order_t;
```

```
//      barcode type  
typedef enum _BAR_TYPE  
{  
    BAR_SYM_AZTEC = 0,  
    BAR_SYM_MESA,  
    BAR_SYM_CODABAR,  
    BAR_SYM_CODE11,  
    BAR_SYM_CODE128,  
    BAR_SYM_CODE39,  
    BAR_SYM_CODE49,  
    BAR_SYM_CODE93,  
    BAR_SYM_COMPOSITE,  
    BAR_SYM_DATAMATRIX,  
    BAR_SYM_EAN8,  
    BAR_SYM_EAN13,  
    BAR_SYM_INT25,  
    BAR_SYM_MAXICODE,  
    BAR_SYM_MICROPDF,  
    BAR_SYM_OCR,  
    BAR_SYM_PDF417,  
    BAR_SYM_POSTNET,
```

```
BAR_SYM_QR,
BAR_SYM_RSS,
BAR_SYM_UPCA,
BAR_SYM_UPCE0,
BAR_SYM_BPO,
BAR_SYM_CANPOST,
BAR_SYM_AUSPOST,
BAR_SYM_IATA25,
BAR_SYM_CODABLOCK,
BAR_SYM_JAPOST,
BAR_SYM_PLANET,
BAR_SYM_DUTCHPOST,
BAR_SYM_MSI,
BAR_SYM_TLCODE39,
BAR_SYM_TRIOPTIC,
BAR_SYM_CODE32,
BAR_SYM_MATRIX25,
BAR_SYM_PLESSEY,
BAR_SYM_CHINAPOST,
BAR_SYM_KOREAPOST,
BAR_SYM_TELEPEN,
BAR_SYM_CODE16K,
BAR_SYM_POSICODE,
BAR_SYM_GS1_128,
BAR_SYM_USPS4CB,
BAR_SYM_IDTAG,
BAR_SYM_ISBT,
BAR_SYM_STRT25,
BAR_SYM_COUPONCODE,
BAR_NUM_SYMBOLOGIES
BAR_SYM_UPCE1,          //      DUMMY value
}BAR_TYPE;

//      Scan result s structure.
typedef struct _SCAN_RESULT
{
    TCHAR          szScanValue[MAX_PATH]; // Scan Value.
```



```

    TCHAR      szScanType[MAX_PATH];    // Scan Type(String).
    TCHAR      tcCodeID;                 // Scan Type ID.
    BAR_TYPE    barType;                 // Scan Type(BAR_TYPE).
    TCHAR      tcSymLetter;              // Symbol Letter.
    TCHAR      tcSymModifier;            // Symbol Modify Letter
    WORD        unScanLength;            // Scan Length.
    TCHAR      szScanMaxValue[MAX_PATH*15];    //      MAX:3900 character
} SCAN_RESULT, *PSCAN_RESULT;

```

// Each barcode type s detail flag

```
typedef struct _BAR_FLAG
```

```

{
    TCHAR  wszFlagName[MAX_PATH];    // detail option subject
    BOOL    bFlag;                  // detail option subject flag
    BYTE    btDIntex;               // detail option index
    BYTE    btDetail;               // just flag for programming(ignore)
}BAR_FLAG, *PBAR_FLAG;

```

// OCR s detail flag

```
typedef struct _OCR_FLAG
```

```

{
    TCHAR  szOCRTemplate[MAX_PATH];    // A null-terminated string that
indicates one or more template patterns for the OCR decode.
    TCHAR  szOCRGroupG[MAX_PATH];      // A null-terminated string that defines
the set of characters matching group "g" in a template.
    TCHAR  szOCRGroupH[MAX_PATH];      // A null-terminated string that defines
the set of characters matching group "h" in a template.
    TCHAR  szOCRCheckChar[MAX_PATH];   // A null-terminated string that defines
the legal characters for checksum computation in a decoded message.
}OCR_FLAG, *POCR_FLAG;

```

// structure for scanner configure

```
typedef struct _BCD_CONFIG
```

```

{
    BOOL        config_scannerenable;    //スキャナ-有効状態
    BOOL        config_bautoscan;    //自動スキャン状態
    DWORD       config_dwtriggertimeout;    // トリガ-タイムアウト時間

```

```

    DWORD    config_dwautoscantriggertime;    //自動スキャン トリガ - 間隔時間.
    BYTE      config_btresulttype;            //結果タイプ 値.
    BYTE      config_btTerminator;            //ターミネータ-値
    BYTE      config_btscankey;               //使用しない.
    BOOL      config_bprefix;                 // プレフィックス使用状態.
    BOOL      config_bpostfix;                // ホストフィックス使用状態
    TCHAR     config_szprefix[MAX_PATH];      // プレフィックス文字列
    TCHAR     config_szpostfix[MAX_PATH];     // ホストフィックス文字列.
    BOOL      config_bbeep;                   // ビープ音使用状態.
    BOOL      config_bvibrator;               // バイブレータ-使用状態.
    BOOL      config_bLED;                    // LED 使用状態.
    TCHAR     config_szscansuccessfile[MAX_PATH]; // スキャン成功時のWAVファイル名.
    TCHAR     config_szscanfailfile[MAX_PATH]; // スキャン失敗時のWAVファイル名
    DWORD     config_dwvibsuccesstime;        // スキャン成功時バイブレータ - 駆動時間.
    DWORD     config_dwvibfailtime;           // スキャン失敗時バイブレータ-駆動時間.
    BAR_TYPE  config_BTcurbartype;            //現在使用中バーコード タイプ
    BYTE      config_btcurindex;              //現在使用中インデックス
    BOOL      config_bBarID;                  // バールコードID使用状態.
    BOOL      config_bCenterWindow;           // [center window]使用状態.
    BYTE      config_btDecodeMode;            // 現在のデコードモード
    DWORD     config_dwDelayDecoding;         // デコード 前遅延時間
    DWORD     config_dwMaxExposure;           // イメージキャプチャにおける最大露光値.
    DWORD     config_dwMaxGain;               // イメージキャプチャにおける最大ゲイン値
    DWORD     config_dwTargetWhite;           // イメージキャプチャにおけるターゲットホワイト値.
    DWORD     config_dwTargetWhiteWindow;     //イメージキャプチャにおけるターゲットホワイトウィンドウ値.
    int       config_nScanImageMode;          // スキャナーモード状態(スキャンモード, イメージモード)
    BOOL      config_bImageAimer;             // 照準の使用状態
    BOOL      config_bImageIllumination;      // 照明の使用状態
} BCD_CONFIG, *PBCD_CONFIG;

// OCR detail string flag
typedef struct _OCR_FLAG
{
    TCHAR     szOCRTemplate[MAX_PATH];
    TCHAR     szOCRGroupG[MAX_PATH];
    TCHAR     szOCRGroupH[MAX_PATH];
    TCHAR     szOCRCheckChar[MAX_PATH];
}OCR_FLAG, *POCR_FLAG

```

3.4.2 エンジン構造体と定義

About of engine configurations.

```
//      define for mode scan or image : Select the decoding mode or image mode.
#define  MODE_SCAN      0x00
#define  MODE_IMAGE     0x01

//      define for Decode Mode : Select the engine s decode mode.
#define  STANDARD        0x01    //      This decode mode is standard
#define  ADVANCED_LINEAR 0x02    //      This decode mode is fit of 1D barcode.
#define  QUICK_OMNI      0x04    //      This decode mode is fast more than standard.

//      order for image action : Select the capture start and sop.
enum
{
    IMAGE_GET_START = 0,
    IMAGE_GET_STOP,
};

//      Scan Light Mode : Select the light mode.
typedef enum SCANLIGHTSMODE SCANLIGHTSMODE;
enum SCANLIGHTSMODE
{
    SCANLIGHTSMODE_OFF= 0,          // Aimers and illumination off.
    SCANLIGHTSMODE_ILLUM_ONLY_ON,   // Illumination only on.
    SCANLIGHTSMODE_AIMER_ONLY_ON,   // Aimers only on.
    SCANLIGHTSMODE_ON,              // Both aimers and illumination on
};

typedef struct ENGINECONFIG ENGINECONFIG;
struct ENGINECONFIG
{
    DWORD dwIlluminationType;    // Writeable 0 = Green, Non Zero = Red
    DWORD dwLedControlMode;      // Read Only at this time : status of interlaced mode or concurrent mode.
    DWORD dwPixelFrequency;      // Read Only at this time : status of 12Mh or 24Mh pixel frequency.
```

```

DWORD dwEngineId;          // Read Only : Engine ID Number.
                             /*
                             0~2 BIT : Focus position of the lens(SF : 001, SR : 010)
                             3~4 BIT : Reserved.
                             5~6 BIT : Type of illumination populated(future use).
                             7~8 BIT : Reserved.
                             9~11 BIT : Aimer type(5000 LED : 000, 5100 Bright Aimer : 001, 5300 Laser : 010)
                             12~14 BIT : Imager type(future use).
                             15 BIT : Reserved.
                             */

DWORD dwFirmwareCksum;     // Read Only : FirmwareCksum
DWORD dwFirmwareVersion;   // Read Only : FirmwareVersion
DWORD dwAimerCenterX;      // Read Only : AimerCenterX
DWORD dwAimerCenterY;      // Read Only : AimerCenterY
DWORD dwEngineSensorID;    // Read Only : EngineSensorID


DWORD dwEngineID;          // EngineIDs values.
                             /*
                             TYPE_NONE = 0, // No imager hardware
                             TYPE_IT4200 = 1,
                             TYPE_IT4000 = 5,
                             TYPE_IT4100 = 6,
                             TYPE_IT4300 = 7,
                             TYPE_IT5000VGA = 10,
                             TYPE_IT5000VGA_P = 11
                             */

DWORD dwImagerRows;        // Number of rows for a given imager.
DWORD dwImagerCols;        // Number of columns for a given imager.
DWORD dwBitsPerPixel;      // Typically this is 8 for byte pixels.
DWORD dwRotation;          // RIGHT_SIDE_UP = 0, ROTATED_RIGHT, UPSIDE_DOWN, ROTATED_LEFT.
DWORD dwAimerXoffset;      // This value represents the X coordinate for the center of the aimer pattern.
DWORD dwAimerYoffset;      // This value represents the Y coordinate for the center of the aimer pattern.
DWORD dwYDepth;            // This value represents the Y Depth
};

```

3.5 メッセージ

下記の記述はSDKで使用するメッセージです。

```
//      define for USER Message

#define          WM_SCANNED                      WM_USER+0x7777
#define          WM_IMAGE                        WM_USER+0x7778
```

(1) WM_SCANNED

スキャン結果タイプが[RESULT_USERMSG]の場合、アプリケーションはスキャン後に[WM_SCANNED]を取得します。

(2) WM_IMAGE

スキャナーモードがイメージモードの場合、アプリケーションはトリガーON/OFF後に[WM_IMAGE]を取得します。
[WM_IMAGE]メッセージの[LPARAM]はトリガーONの時、[IMAGE_GET_START]です。
[WM_IMAGE]メッセージの[LPARAM]はトリガーOFFの時、[IMAGE_GET_STOP]です。

下記記述は[WM_SCANNED]と[WM_IMAGE]メッセージを使用したサンプルコードです。

```
BOOL CCaptureDiagDlg::PreTranslateMessage(MSG* pMsg)
{
//      about scanner
if(pMsg->message == WM_SCANNED)
{
    if(pMsg->lParam == ERROR_NO_ERROR)
    {
        //      Success get scan result.
    }
    else
    {
        //      Fail get scan result.
    }
}

//      about image
if(pMsg->message == WM_IMAGE)
{
    if(pMsg->lParam == IMAGE_GET_START)
    {
        //      Image Capture Start
    }
    else //  pMsg->lParam == IMAGE_GET_STOP
    {
        //      Image Capture Stop
    }
}
return CDialog::PreTranslateMessage(pMsg);
}
```

4 BCD.NET.dll

4.1 BCD.NET.dllとは

[BCD.NET.dll]は、.NETユーザーに対して、DllImportを通して[BcdCore.dll]のいくつかの関数を使用できるようにしたものです。この関数は、まさにインポートしただけであり、[BcdCore.dll]の関数と同じです。
[BCD.NET.dll]はスキャナを使う上で最小の関数のみ対応しています。

4.2 BCD.NET.dll の構成

1. スキャンデータを[OnScanned]関数に転送します。[OnScanned]関数はスキャナイベントハンドラーです。
2. コンストラクターの中でスキャナを制御するために必要な関数を呼び出します。
コンストラクター中に :
 BcdOpen();
 BcdEnable(true);
 BcdSetResultType(0);
 BcdSetHWND(scanMsgWnd.Hwnd);
3. スキャナを制御するための主な関数の再定義(DllImportを使って)
 BcdEnable(bool bEnable) : public void Enable(bool bEnable)
 BcdSetHWND(IntPtr hWnd) : public void SetHWND(IntPtr hWnd)
 BcdEnableBeep(bool bEnable) : public void EnableBeep(bool bEnable)
 BcdEnableVibrator(bool enable) : public void EnableVibrator(bool enable)
 BcdEnableAutoScan(bool bEnable) : public void EnableAutoScan(bool bEnable)
 BcdEnableBarcode(int nBarType, bool bEnable) : public void EnableBarcode(int nBarType, bool bEnable)
 BcdTriggerTurnOn(bool bTurnOn) : public void TriggerTurnOn(bool bTurnOn)
 BcdSetScanTimeOut(uint uTimeOut) : public void SetScanTimeOut(uint uTimeOut)
 BcdSetAutoScanInterval(uint dwInterval) : public void SetAutoScanInterval(uint dwInterval)
 BcdSetSuccessSound(string strSFileName) : public void SetSuccessSound(string strSFileName)
 BcdSetFailSound(string strFFFileName) : public void SetFailSound(string strFFFileName)
 BcdGetBarTypeString(uint nBarType) : public unsafe string GetBarTypeString(uint nBarType)
 Bcdstring GetModelName() : public unsafe string GetModelName()
 Bcdstring GetModelSerialNumber() : public unsafe string GetModelSerialNumber()

4.3 BCD.NET.dll 使用サンプル

下記のコードは付属サンプルプログラム(tScanner)から抽出したものです。

```
...
using BCD.net;
...
namespace tScanner
{
    public partial class Form1 : Form
    {
        public CBScanner BScanner;
        public Form1()
        {
            InitializeComponent();
            // for scanner
            this.BScanner = new BCD.net.CBScanner();
            BScanner.Enable(true);
            this.BScanner.OnScanned += new BCD.net.ScannerEventHandler(this.FormScanTest_OnScanned);
            this.BScanner.EnableAutoScan(false);
            this.BScanner.EnableBeep(true);
            this.BScanner.EnableVibrator(false);
        }

        public void FormScanTest_OnScanned(string strScanData, uint nLen, int barType, long errCode)
        {
            if (errCode == 0)
            {
                textBox1.Text = strScanData;
                textBox2.Text = this.BScanner.GetBarTypeString((uint)barType);
            }
            else
            {
                textBox1.Text = "SCAN FAIL";
                textBox2.Text = "SCAN FAIL";
            }
        }

        private void checkBox1_CheckStateChanged(object sender, EventArgs e)
        {
            this.BScanner.EnableAutoScan(checkBox1.Checked);
        }

        private void checkBox2_CheckStateChanged(object sender, EventArgs e)
        {
            this.BScanner.EnableBeep(checkBox2.Checked);
        }

        private void checkBox3_CheckStateChanged(object sender, EventArgs e)
        {
            this.BScanner.EnableVibrator(checkBox3.Checked);
        }

        private void button1_Click(object sender, EventArgs e)
```

```
{
    this.BScanner.TriggerTurnOn(true);
}

private void button2_Click(object sender, EventArgs e)
{
    this.BScanner.EnableAutoScan(false);
    checkBox1.Checked = false;
    this.BScanner.TriggerTurnOn(false);
}
}
```

[BCD.NET.dll]はすべてのスキャナ-関数に対応していません。

[BCD.NET.dll]はスキャナ-を使う上で最小の関数に対応しています。

もしその他の関数を使用したい場合は、[DllImport]を通して[BcdApi.h]の中のすべての関数を使用できます。