

Programmer's guide

M3 OX10 SDK Version 1.3 Manual for Microsoft® .NET CF

Abstract

This document describes the methods and properties of the M3 MOBILE OX10 SDK version 1.3. OX10 SDK supports M3 OX10 unless written separately.

Modules covered:

1D Scanner, 2D Imager, Camera, RFID (LF/HF), UHF Gun, WLAN, Bluetooth, GPS, System

Copyright and Agreement

WARNING: All contents of this SDK manual are protected by the copyright laws and all rights are reserved. Unauthorized distribution or copying is strictly prohibited.

M3 Mobile does not guarantee the quality and performance of the programs written in unsupported programming language. For supported development tools and languages, please refer to Development Tool and Requirements section.

Development Tools and Requirements

Supported Development Tools

- Visual Studio 2005/2008 – Visual C++, Visual C#, Visual Basic .NET

Development System Requirements

- Pentium – 1 Gigahertz (GHz) processor or higher
- Microsoft Windows 98 / ME / 2000 / XP / 2003 / 7
- ActiveSync, Windows Mobile Device Center (WMDC)

Development Platform Requirements

- "M3 Plus Platform SDK" and "Windows Mobile 5.0 SDK for Pocket PC" must be installed on your computer to be able to develop software using this SDK.
- M3Plus Platform SDK : [DOWNLOAD](#)
- Windows Mobile 5.0 SDK for Pocket PC : [LINK](#)

Release History

M3 OX10 SDK Version 1.3 Date: July 2016

- Added 45N Module of Wireless LAN to supported model

M3 OX10 SDK Version 1.2 Date: March 2016

- Added UHF Gun SDK
- Remove the POS SDK and LR Scanner SDK

UNI SDK Version 1.1.1 Date: November 2013

- Added M3 OX10 CE to supported model
- System SDK
 - DLL names have been changed
 - SystemNet.DLL => M3SystemNet.DLL

UNI SDK Version 1.1.0 Date: June 2013

- Scanner (1D)
 - Added M3 OX10 WM to supported model
 - Modified to include Symbol H/W decoder
 - Minor bug fixes
- Imager (2D)
 - Added M3 OX10 WM to supported model
 - IQ setting save error fixed
- Camera (Windows CE only)
 - M3 SMART CE has been included
 - GetBrightness function added
- RFID (LF/HF)
 - Improved port open procedure
 - SendCommandData function added
- System SDK
 - Added M3 POS, MM3 to supported model
 - Gyro sensor control added for M3 SMART

UNI SDK Version 1.0.0 Date: January 2012

- First release of combined SDK called UNI SDK

Contents

1	Introduction	0
2	Samples	1
3	Tutorial.....	2
3.1	SCANNER (1D)	4
3.2	IMAGER (2D)	6
3.3	CAMERA (CE)	8
3.4	CAMERA (WM)	10
3.5	RFID (LF/HF)	12
3.6	RFID Gun SDK.....	14
3.7	WLAN	17
3.8	BLUETOOTH.....	19
3.9	GPS	30
3.10	RFID (HF)	36
3.11	SYSTEM SDK	41
4	References.....	45
4.1	SCANNER (1D)	46
4.1.1	CLOSE	53
4.1.2	GetCODABAR.....	53
4.1.3	GetCODE11	54
4.1.4	GetCODE128.....	55
4.1.5	GetCODE25.....	56
4.1.6	GetCODE39.....	57
4.1.7	GetCODE93.....	58
4.1.8	GetDeviceType	59
4.1.9	GetEAN13	59
4.1.10	GetEAN8	60
4.1.11	GetEngineType	61
4.1.12	GetGS1	61
4.1.13	GetKOREAPOST	62
4.1.14	GetMSI.....	63
4.1.15	GetOption	63
4.1.16	GetScanData.....	64
4.1.17	GetSymbology.....	65
4.1.18	GetTELEPEN.....	66
4.1.19	GetUPCA	67
4.1.20	GetUPCE	68
4.1.21	GetVersionInfo.....	68
4.1.22	Open	69

4.1.23	Read	70
4.1.24	ReadCancel	70
4.1.25	RegHotKey.....	71
4.1.26	SetCODABAR	72
4.1.27	SetCODE11	73
4.1.28	SetCODE128	73
4.1.29	SetCODE25	74
4.1.30	SetCODE39	75
4.1.31	SetCODE93	76
4.1.32	SetEAN13.....	77
4.1.33	SetEAN8.....	78
4.1.34	SetGS1	78
4.1.35	SetKOREAPOST.....	79
4.1.36	SetMSI	80
4.1.37	SetOption.....	81
4.1.38	SetSymbology	82
4.1.39	SetSymbologyAll	83
4.1.40	SetSymbologyDefault	83
4.1.41	SetTELEPEN	84
4.1.42	SetUPCA.....	84
4.1.43	SetUPCE.....	85
4.1.44	UnRegHotKey	86
4.2	IMAGER (2D)	88
4.2.1	CAMCapture	102
4.2.2	CAMGetOption	102
4.2.3	CAMInit.....	103
4.2.4	CAMPreviewStart	104
4.2.5	CAMPreviewStop	104
4.2.6	CAMSetOption.....	105
4.2.7	CAMUnInit.....	106
4.2.8	Close	106
4.2.9	GetAZTEC	107
4.2.10	GetCenteringWindow.....	108
4.2.11	GetCHINAPOST	108
4.2.12	GetCODABAR	109
4.2.13	GetCODABLOCK	110
4.2.14	GetCODE11	111
4.2.15	GetCODE128	111
4.2.16	GetCODE16K	112
4.2.17	GetCODE39.....	113

4.2.18	GetCODE49.....	114
4.2.19	GetCODE93.....	114
4.2.20	GetCOMPOSITE	115
4.2.21	GetDATAMATRIX.....	116
4.2.22	GetDecOption.....	117
4.2.23	GetDeviceType	117
4.2.24	GetEAN13	118
4.2.25	GetEAN8	118
4.2.26	GetIATA25.....	119
4.2.27	GetImagerMode.....	120
4.2.28	GetINT25.....	120
4.2.29	GetKOREAPOST	121
4.2.30	GetMATRIX25	122
4.2.31	GetMAXICODE	123
4.2.32	GetMICROPDF	123
4.2.33	GetMSI.....	124
4.2.34	GetOCR	125
4.2.35	GetOption	126
4.2.36	GetPDF417.....	127
4.2.37	GetPLANET.....	127
4.2.38	GetPLESSEY	128
4.2.39	GetPOSICODE.....	129
4.2.40	GetPOSTNET.....	129
4.2.41	GetQR	130
4.2.42	GetRSS	131
4.2.43	GetScanData.....	132
4.2.44	GetSTRT25	132
4.2.45	GetSymbology.....	133
4.2.46	GetTELEPEN.....	135
4.2.47	GetUPCA	136
4.2.48	GetUPCE	136
4.2.49	GetVersionInfo	137
4.2.50	IQGetBarcodeData.....	138
4.2.51	IQGetOption	138
4.2.52	IQImagingStart.....	139
4.2.53	IQImagingStop.....	140
4.2.54	IQInit.....	140
4.2.55	IQSetOption.....	141
4.2.56	IQUnInit	142
4.2.57	Open	142

4.2.58	Read	143
4.2.59	ReadCancel	143
4.2.60	RegHotKey.....	144
4.2.61	SetAZTEC	145
4.2.62	SetCenteringWindow	146
4.2.63	SetCHINAPOST	146
4.2.64	SetCODABAR	147
4.2.65	SetCODABLOCK	148
4.2.66	SetCODE11	149
4.2.67	SetCODE128	149
4.2.68	SetCODE16K	150
4.2.69	SetCODE39	151
4.2.70	SetCODE49	152
4.2.71	SetCODE93	152
4.2.72	SetCOMPOSITE.....	153
4.2.73	SetDATAMATRIX	154
4.2.74	SetDecOption	155
4.2.75	SetEAN13.....	155
4.2.76	SetEAN8.....	156
4.2.77	SetIATA25	157
4.2.78	SetINT25.....	157
4.2.79	SetKOREAPOST.....	158
4.2.80	SetMATRIX25	159
4.2.81	SetMAXICODE.....	159
4.2.82	SetMICROPDF.....	160
4.2.83	SetMSI	161
4.2.84	SetOCR.....	162
4.2.85	SetOption.....	162
4.2.86	SetPDF417	163
4.2.87	SetPLANET	164
4.2.88	SetPLESSEY	165
4.2.89	SetPOSICODE	165
4.2.90	SetPOSTNET	166
4.2.91	SetQR.....	167
4.2.92	SetRSS	168
4.2.93	SetSTR25.....	168
4.2.94	SetSymbology	169
4.2.95	SetSymbologyAll.....	171
4.2.96	SetSymbologyDefault.....	172
4.2.97	SetTELEPEN.....	172

4.2.98	SetUPCA.....	173
4.2.99	SetUPCE.....	174
4.2.100	UnRegHotKey.....	174
4.3	CAMERA (CE)	176
4.3.1	Open.....	180
4.3.2	Close	180
4.3.3	PreviewStart.....	181
4.3.4	PreviewStop.....	181
4.3.5	Capture.....	182
4.3.6	GetLastSaveFilePath.....	183
4.3.7	AutoFocus.....	183
4.3.8	EnableAutoAF.....	184
4.3.9	Zoom.....	184
4.3.10	SetCameraOption	185
4.3.11	GetCameraOption.....	186
4.3.12	Brightness	187
4.3.13	FlashOn.....	187
4.3.14	FlashOff	188
4.3.15	RawData.....	188
4.3.16	InsertExifInformation.....	189
4.3.17	UseGPSExifData	190
4.3.18	GetScannerType	190
4.3.19	GetVersion.....	191
4.4	CAMERA (WM)	192
4.4.1	Open.....	196
4.4.2	Close	196
4.4.3	SetPreviewWindow	197
4.4.4	PreviewStart.....	198
4.4.5	PreviewStop.....	198
4.4.6	GetRegCapturePath	199
4.4.7	SetRegCapturePathEx.....	200
4.4.8	Capture.....	201
4.4.9	CaptureEx.....	202
4.4.10	VideoStart	202
4.4.11	VideoStop	203
4.4.12	VideoStopEx	204
4.4.13	GetFlashState.....	204
4.4.14	AlwaysFlash	205
4.4.15	CaptureFlash.....	205
4.4.16	AutoFocus.....	206

4.4.17	SetAfMode	207
4.4.18	Zoom.....	207
4.4.19	SetResolution.....	208
4.4.20	GetResolution	208
4.4.21	SetQuality	209
4.4.22	GetQuality.....	210
4.4.23	SetBrightness.....	210
4.4.24	GetBrightness	211
4.4.25	SetWhiteBalance.....	211
4.4.26	GetWhiteBalance	212
4.4.27	SetContrast	212
4.4.28	GetContrast	213
4.4.29	SetSharpness	214
4.4.30	GetSharpness.....	214
4.4.31	SetHistoEqual.....	215
4.4.32	GetHistoEqual	215
4.4.33	RegisterPreview	216
4.4.34	InsertExifInformation.....	217
4.4.35	UseGPSExifData	217
4.4.36	GetModelType.....	218
4.4.37	GetScannerType	219
4.4.38	GetVersion.....	220
4.5	RFID (LF/HF)	221
4.5.1	Open.....	224
4.5.2	Close	225
4.5.3	PowerSupply.....	225
4.5.4	GetRadioType.....	226
4.5.5	SelectTag	227
4.5.6	HighSelectTag	227
4.5.7	LoginTag.....	228
4.5.8	ReadBlock.....	229
4.5.9	WriteBlock	230
4.5.10	ReadMultiBlock.....	230
4.5.11	WriteMultiBlock	231
4.5.12	SetAntenna	232
4.5.13	GetVersion.....	233
4.5.14	EnableMultiTag	234
4.5.15	SendReadMultiTag.....	235
4.5.16	SendTransferCommand.....	235
4.5.17	SendContinuousRead	236

4.5.18	SendCommand.....	237
4.5.19	SendCommandGetData	237
4.5.20	GetData.....	238
4.5.21	CheckResult	239
4.5.22	SoundPlay	240
4.5.23	SetTagType	241
4.5.24	GetTagType.....	241
4.5.25	GetTagTypeToString.....	242
4.5.26	TagItInventory	242
4.5.27	TagItReadBlock	243
4.5.28	TagItWriteBlock	244
4.6	UHF GUN READER.....	245
4.6.1	GetError	251
4.6.2	IsReady	251
4.6.3	Init	252
4.6.4	Close	253
4.6.5	Inventory	254
4.6.6	InventoryStop	254
4.6.7	Read	255
4.6.8	Write	256
4.6.9	Lock.....	257
4.6.10	Kill.....	258
4.6.11	GetData.....	259
4.6.12	SetRegionFrequency	260
4.6.13	GetRegionalFrequency	261
4.6.14	ReadBattery	261
4.6.15	ReadBatteryStatus	262
4.6.16	Version.....	262
4.7	WLAN	264
4.7.1	ActivateConfig	267
4.7.2	Close	267
4.7.3	ConnectAP	268
4.7.4	ConnectAPEX	269
4.7.5	DeleteConfig.....	270
4.7.6	ExportConfig.....	271
4.7.7	ImportConfig.....	271
4.7.8	Init	272
4.7.9	GetAllConfigName	273
4.7.10	GetCurrentAPInfo	273
4.7.11	GetBssidList	274

4.7.12	GetPowerStatus.....	275
4.7.13	PowerOn	275
4.7.14	PowerOff	276
4.8	BLUETOOTH.....	277
4.8.1	Close	292
4.8.2	Connect	292
4.8.3	CreateConnection	293
4.8.4	CreateConnectionEx	293
4.8.5	DeleteConnection	294
4.8.6	Disconnect	295
4.8.7	FindConnectionClose.....	295
4.8.8	FindDeviceClose	296
4.8.9	FindFirstConnection	297
4.8.10	FindFirstConnectionEx	297
4.8.11	FindFirstDevice	298
4.8.12	FindFirstDeviceEx	299
4.8.13	FindFirstService	300
4.8.14	FindFirstServiceEx	301
4.8.15	FindLocalDevice	302
4.8.16	FindNextConnection	302
4.8.17	FindNextConnectionEx	303
4.8.18	FindNextDevice	304
4.8.19	FindNextService	304
4.8.20	FindNextServiceEx	305
4.8.21	FindServiceClose	306
4.8.22	GetBluetoothState	307
4.8.23	GetSCOConnectionState	307
4.8.24	GetSecurityMode.....	308
4.8.25	Open	309
4.8.26	PerformAction.....	309
4.8.27	SetAuthenticationCallback	310
4.8.28	SetBluetoothState	311
4.8.29	SetConnectionCallback.....	312
4.8.30	SetIncomingPIN	313
4.8.31	SetOutgoingPIN	314
4.8.32	SetSCOConnectionState.....	314
4.8.33	SetSecurityMode	315
4.9	GPS	317
4.9.1	AGPSDown	320
4.9.2	AGPSRun	320

4.9.3	Close	321
4.9.4	EnableStaticMode	322
4.9.5	ModuleRestart.....	322
4.9.6	Open.....	323
4.10	SYSTEM	325
4.10.1	BacklightOn.....	328
4.10.2	Cleanboot	328
4.10.3	GetBacklightlevel.....	329
4.10.4	GetBacklightTimeOut	330
4.10.5	GetBatteryLifePercent.....	331
4.10.6	GetBatteryState.....	332
4.10.7	GetBluetooth	332
4.10.8	GetCpuClock	333
4.10.9	GetDeviceInfo	334
4.10.10	GetGSensorValue	335
4.10.11	GetOSVersionInfo	336
4.10.12	GetPhoneVolumeLevel	336
4.10.13	GetPowerTimeOut	338
4.10.14	GetSerialNumber.....	338
4.10.15	GetVolumeLevel	339
4.10.16	GetWlan.....	339
4.10.17	KeypadLock	340
4.10.18	Reboot	341
4.10.19	SetBacklightlevel	341
4.10.20	SetBacklightTimeOut	342
4.10.21	SetBluetooth	343
4.10.22	SetCpuClock.....	343
4.10.23	SetGSensorValue.....	344
4.10.24	SetPhoneVolumeLevel	345
4.10.25	SetPowerTimeOut.....	345
4.10.26	SetSleepMode	346
4.10.27	SetVolumeLevel.....	347
4.10.28	SetWlan	347
4.10.29	Vibrate.....	348
4.10.30	VolumeMute	348
4.10.31	GetGuid	349
5	Demo Manual	351
5.1	WLAN	351
5.2	BLUETOOTH.....	354
5.3	POS	355

5.4 SYSTEM357

1 Introduction

This document is a reference guide for the software developer's kit (SDK) for M3 OX10 Devices. This handheld terminal may include up to 6 different modules depending on the specification of the device. For module list and brief description of each module, see below table.

Module Name	Description
SCANNER	Read 1D barcodes depend on the scanner module option.
IMAGER	Read 1D/2D barcodes depend on the scanner module option.
CAMERA	Used to take a picture of an object.
RFID	Read data from tags or write to the tags through Reader.
UHF Gun	Read data from tags or write to the tags through Reader.
WLAN	Enable network connection using Wi-Fi. Summit WLAN module is used.
BLUETOOTH	Mainly used to connect to a Bluetooth head set for phone calls or printer.
GPS	Gathers information on current location through satellite.
SYSTEM	System status can be control or read.

This guide document provides comprehensive SDK manual by providing Tutorials, Samples and References including function lists.

2 Samples

This chapter is illustrate the demo programs included in the DLL, SDK and current version of the SDK as well as the development tool used to write the sample program.

Module	Demo Application	Sample Source	Version	Date
SCANNER	ScanTestNet.exe Scanner.dll ScannetNet.dll	SCANNER_TEST	1.0.1 1.2.3 1.0.3 (1005)	2013-10-14 2013-09-25 2013-10-14
IMAGER	ImagerTestNet.exe Imager.dll ImagerNet.dll	IMAGER_TEST	1.0.1 1.0.4 1.0.3	2013-10-14 2013-10-25 2013-10-14
CAMERA_CE	CamNetTest.exe CAM.dll CamNet.dll	CAMERA_TEST_NET	1.2.0 1.2.0 1.1.0	2013-10-10 2013-09-09 2012-09-20
CAMERA_WM	CamTestNet.exe CAM.dll CamNet.dll	CAMERA_TEST	1.2.0 1.1.1 (1001) 1.0.0	2013-08-07 2013-04-26 2012-01-13
RFID	RfidTestNet.exe RFID.dll RFIDNet.dll	RFID_TEST_NET	1.0.4 1.0.4 1.1.0	2013-03-18 2013-03-18 2012-09-20
UHF GUN READER	UHF_Gun_Net_Test.exe RFID_UHF.dll RFID_UHF_Net.dll	UHF_Gun_NET_TEST	1.0.1	2014-04-24
WLAN	WlanTestNet.exe WLAN.dll WLANNET.dll	WLAN_TEST	1.0.1 1.1.5 1.0.1	2013-09-09 2013-10-10 2012-09-12
BLUETOOTH	BluetoothTestNet.exe BLUETOOTH.dll BluetoothNet.dll	BLUETOOTH_TEST	1.0.1 1.0.0 1.0.0	
GPS	GpsTestNet.exe GPS.dll GpsNet.dll	GPS_TEST	1.0.2 1.0.1 1.0.1	
SYSTEM	SystemTestNet.exe M3System.dll M3SystemNet.dll	SYSTEM_TEST	1.1.3 1.1.6 1.1.3	2013-09-06 2013-10-10 2013-10-04

3 Tutorial

This chapter describes the basic usage of M3 Mobile SDK functions in a step-by-step manner. In this tutorial section, the following topics are treated.

Section	Topic
SCANNER	Initialization ScanRead Get Window Message for getting Scan Data Close Scanner
IMAGER	Initialization ScanRead Get Window Message for getting Scan Data Close Scanner
CAMERA_CE	Open Close Capture Still Shot Preview Insert EXIF Information
CAMERA_WM	Open Close Capture Still Preview Insert EXIF Information
RFID	Open Antenna On/Off Tag Select Data Read Data Write
UHF GUN READER	Init UHF Start Inventory Get Data Stop Inventory Close UHF
WLAN	Initialization Searching AP Connect Config Delete Config Change Config Configuration Import/Export Close Wlan
BLUETOOTH	Initialization Device Find Service Find Connection Management Connect Disconnect Close
GPS	Open Close Receive Message from GPS Module AGPS
Printer	Init Printer Close Printer About Spool Add Command isReady Printer. Get Status

RFID

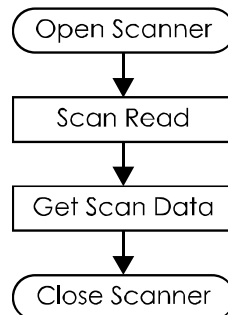
Open Rfid
Close Rfid
Start or Stop scanning Tag

3.1 SCANNER (1D)

Supported PDA:

Brand	Restriction
M3 OX10	No restriction

Basic scanner flow chart:



Tip !!

- I. Scanner communicates through serial so that other programs cannot use scanner while scanner is being run. For example, it occurs an error when other program access scanner while ScanEmul is running.
- II. Combined version of Imager.dll can be available for all 1D Scanner regardless of model type, Scanner Engine and OS.
- III. Its Virtual Keys are VK_F22 for WinCE and VK_F14 for Windows Mobile.
- IV. Scan data can be simply obtained by adding Scanner event with APIs related HotKey.

Open Scanner

```
private Scanner m_Scanner;
public const int m_nHotKeyCE = 133; // VK_F22(WinCE)
public const int m_nHotKeyWM = 125; // VK_F14(WM)
:
m_Scanner = new Scanner();
m_Scanner.ScannerDataEvent += new ScannerNet.ScannerDataDelegate(OnScanRead);
// Get Device Type
m_DeviceType = m_Scanner.GetDeviceType();
// OS Type
if ((m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3SMART_CE) || (m_DeviceType ==
SCAN_DEVICE_TYPE.DEVICE_M3GREEN) || (m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3T) || (m_DeviceType ==
SCAN_DEVICE_TYPE.DEVICE_M3POS))
    m_bWinCE = true;
//Scanner Open
m_Scanner.Open();
if (m_bWinCE == true)
    m_Scanner.RegHotKey(1, m_nHotKeyCE, m_bSyncMode);
```

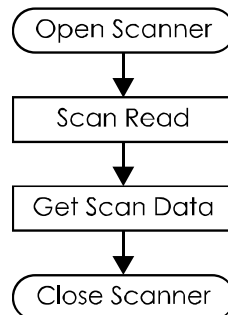
<pre>else m_Scanner.RegHotKey(1, m_nHotKeyWM, m_bSyncMode);</pre>
Scan Read
<pre>private void BN_SCAN_Click(object sender, EventArgs e) { m_Scanner.Read(); }</pre>
Get ScanData
<pre>m_Scanner.ScannerDataEvent += new ScannerNet.ScannerDataDelegate(OnScanRead); : public void OnScanRead(object sender, ScannerDataArgs e) { if (e.ScanData != "") { LB_TYPE.Text = e.ScanType; TB_DATA.Text = e.ScanData; } }</pre>
Close Scanner
<pre>m_Scanner.UnRegHotKey(1); for (int i = 0; i < 3; i++) { m_bResult = m_Scanner.Close(); Thread.Sleep(300); if (m_bResult == true) break; }</pre>

3.2 IMAGER (2D)

Supported PDA:

Brand	Restriction
M3 OX10	No restriction

Basic imager flow chart:



Tip !!

- I. Imager Driver and Camera Driver use the same Quick Capture Interface of CPU. This interface cannot be used at the same time. Because of unloading imager driver while running the camera program so that Imager cannot be used.
- II. Combined version of Imager.dll can be available for all 2D Scanner, except the Long-Range Scanner, regardless of model type and OS.
- III. Its Virtual Keys are VK_F22 for WinCE and VK_F14 for Windows Mobile.
- IV. Scan data can be simply obtained by adding Scanner event with APIs related HotKey.

Open Scanner

```
public Imager m_Imager;
public const int m_nHotKeyCE = 133; // VK_F22(WinCE)
public const int m_nHotKeyWM = 125; // VK_F14(WM)
        :
m_Imager = new Imager();
m_Imager.ImagerDataEvent += new ImagerNet.ImagerDataDelegate(OnScanRead);
// Get Device Type
m_DeviceType = m_Imager.GetDeviceType();
// OS Type
if ((m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3SMART_CE) || (m_DeviceType ==
SCAN_DEVICE_TYPE.DEVICE_M3GREEN) || (m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3T) || (m_DeviceType ==
SCAN_DEVICE_TYPE.DEVICE_M3POS))
    m_bWinCE = true;
//Imager Open
m_Imager.Open();
if (m_bWinCE == true)
```

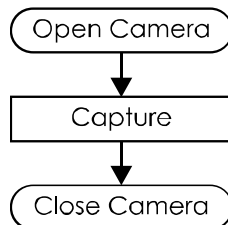
<pre>m_Imager.RegHotKey(1, m_nHotKeyCE, m_bSyncMode); else m_Imager.RegHotKey(1, m_nHotKeyWM, m_bSyncMode); m_Imager.SetSymbologyAll();</pre>
Scan Read
<pre>private void BN_SCAN_Click(object sender, EventArgs e) { m_Imager.Read(); }</pre>
Get ScanData
<pre>m_Imager.ImagerDataEvent += new ImagerNet.ImagerDataDelegate(OnScanRead); : public void OnScanRead(object sender, ImagerDataArgs e) { if (e.ScanData != "") { LB_TYPE.Text = e.ScanType; TB_DATA.Text = e.ScanData; } }</pre>
Close Scanner
<pre>m_Imager.UnRegHotKey(1); for (int i = 0; i < 3; i++) { m_bResult = m_Imager.Close(); Thread.Sleep(300); if (m_bResult == true) break; }</pre>

3.3 CAMERA (CE)

Supported PDA:

Brand	Restriction
M3 OX10	No restriction

Basic camera flow chart:



Tip !!

- I. Its Virtual Key is VK_23.
- II. File name can be editable when capturing pictures. File is stored with file name that users input, otherwise it will be stored by date as default.
- III. CapStatusEvent function is for current status of capturing pictures and name of picture can be obtained through CAM_GetLastSaveFilePath function when capturing is done completely.
- IV. EXIF information can be inserted when enabling option for EXIF information. GPS information can be included in EXIF information and for that, GPS supported device is required. After enabling option for GPS information, GPS information can be confirmed when receiving the grid information from the satellite.
- V. Self Auto-Focus function can be used in devices that AF function is available such as M3 T. It recognizes changes in the preview screen automatically and perform focusing automatically.

Open Camera
<pre>using CamNet; Cam m_Cam = new Cam(); Cam.MODEL_TYPE model = m_Cam.Open(this.Handle, pictureBox_Preview.Handle); if (model == Cam.MODEL_TYPE.MODEL_UNKNOWN) { MessageBox.Show("Camera can not open"); return; }</pre>
Capture Still
<pre>String strSavePath = ""; m_Cam.Capture(strSavePath);</pre>
Preview
<pre>bool bPreview = true;</pre>

```
if(bPreview)
{
    m_Cam.PreviewStart();
}
else
{
    m_Cam.PreviewStop();
}
```

Insert Exif Information

```
string pathTmp;
m_Cam.GetLastSaveFilePath(out pathTmp);
int cnt = 0;
for (int i = 0; i < pathTmp.Length; i++)
    if (pathTmp[i] == '\\')
    {
        cnt = i;
        break;
    }
pathTmp = pathTmp.Substring(0, cnt);
Cam.ExifInfo info = new Cam.ExifInfo();
char delimiter = '\\';
string FilePath = pathTmp.ToString();
string[] tmp = pathTmp.ToString().Split(delimiter);
info.TitleName = tmp[tmp.Length - 1];
info.Make = "M3 Mobile Co.Ltd";
info.Model = "M3Green";
m_Cam.InsertExifInformation(FilePath, info);
```

Close Camra

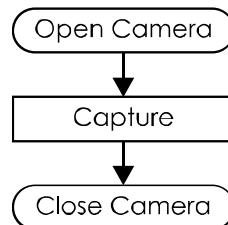
```
M_Cam.Close();
```

3.4 CAMERA (WM)

Supported PDA:

Brand	Restriction
M3 OX10	No restriction

Basic camera flow chart:



Tip !!

- I. Combined version of Camera.dll for Windows Mobile OS.
- II. Its Virtual Key is VK_13.
- III. File name can be editable when capturing pictures. File is stored with file name that users input, otherwise it will be stored by date as default.
- IV. EXIF information can be inserted when enabling option for EXIF information. GPS information can be included in EXIF information and for that, GPS supported device is required. After enabling option for GPS information, GPS information can be confirmed when receiving the grid information from the satellite.
- V. Self Auto-Focus function can be used in devices that AF function is available such as M3 SKY and M3 ORANGE. It recognizes changes in the preview screen automatically and perform focusing automatically.

Open Camera
<pre>using CamNet; Cam m_Cam = new Cam(); m_Cam.Open(this.Handle, Cam.CAMERA_MODE.STILL_MODE, Cam.VIDEO_TYPE.VIDEO_WMV);</pre>
Capture Still
<pre>char[] strOutFile = new char[260]; m_Cam.Capture("Flash Disk\\Camera\\", strOutFile, true);</pre>
Preview
<pre>void PreviewStart(bool bStart) { if(bStart) { m_Cam.PreviewStart(); } }</pre>


```
}  
else  
{  
    m_Cam.PreviewStop();  
}  
}
```

Insert Exif Information

```
string pathTmp;  
m_Cam.GetLastSaveFilePath(out pathTmp);  
int cnt = 0;  
for (int i = 0; i < pathTmp.Length; i++)  
    if (pathTmp[i] == '\\0')  
    {  
        cnt = i;  
        break;  
    }  
pathTmp = pathTmp.Substring(0, cnt);  
Cam.ExifInfo info = new Cam.ExifInfo();  
char delimiter = '\\';  
string FilePath = pathTmp.ToString();  
string[] tmp = pathTmp.ToString().Split(delimiter);  
info.TitleName = tmp[tmp.Length - 1];  
info.Make = "M3 Mobile Co.Ltd";  
info.Model = "M3SKY";  
m_Cam.InsertExifInformation(FilePath, info);
```

Close Camra

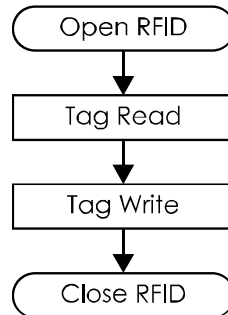
```
m_Cam.Close();
```

3.5 RFID (LF/HF)

Supported PDA:

Brand	Restriction
M3 OX10	HF RFID modules is installed. UHF is not supported

Basic RFID flow chart:



Tip !!

- I. Power when using RFID can be reduced by Antenna On/Off.
- II. Read/Write speed can be improved when setting the tag type to read.
- III. RFID_SendContinuousRead and RFID_GetData functions are suitable if using Serial Number only. There is RFID_SelectTag function that has same function.
- IV. The latest Firmware version of RFID module is 1.2.5 and FW upgrade is not supported through module itself.
- V. RFID Close cuts power with RFID_PowerSupply function and closes program.

Open RFID

```
using RFIDNet;
RFIDCommon RfidCommon = new RFIDCommon();
RFIDCommon.RFID_TYPE type = RfidCommon.Open();
if(type == RFIDCommon.RFID_TYPE .RFID_LF)
{
    // LF RFID
}
else if(type == RFIDCommon.RFID_TYPE.RFID_HF)
{
    // HF RFID
}
else
{
    return;
}
```

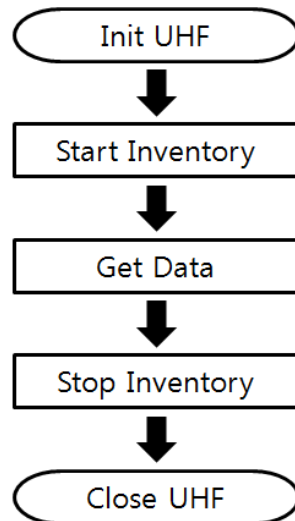
Antenna On/Off
<pre>if(bAntennaOn == TRUE) { RfidCommon.SetAntenna(TRUE); } else { RfidCommon .SetAntenna(FALSE); }</pre>
Tag Select
<pre>StringBuilder strSerial = new StringBuilder(260); RfidCommon.SelectTag(strSerial);</pre>
Data Read
<pre>StringBuilder strSerial = new StringBuilder(); RfidCommon.SelectTag(strSerial); const int nMaxData = 1024; StringBuilder strData = new StringBuilder(nMaxData); StringBuilder strOutData = new StringBuilder(nMaxData); RfidCommon.ReadBlock("01", strData);</pre>
Data Write
<pre>ngBuilder(); RfidCommon.SelectTag(strSerial); const int nMaxData = 1024; StringBuilder strData = new StringBuilder(nMaxData); StringBuilder strOutData = new StringBuilder(nMaxData); m_Rfid.WriteBlock("01", "00112233", strOutData);</pre>

3.6 RFID Gun SDK

Supported PDA:

Brand	Restriction
M3 OX10	Need to have UGR Gun

Basic UHF GUN READER flowchart is shown below.



< UHF RFID flow chart >

Tip!!

- I. It can be used with both Windows Mobile and Windows CE platforms.
- II. Initialize will return fail in the following conditions:
 - A. PDA detached from gun reader
 - B. RFID / SCAN switch is at SCAN position
 - C. Gun reader's battery is detached or empty
 - D. PDA in gun reader is synced with PC
- III. Power status can be obtained using delegate ReceivedPower. When turning off, UHF RFID must be closed. When turning on, UHF RFID must be initialized.
- IV. UHF RFID uses the same virtual key as scanner; VK_H14 for WM and VK_F22 for CE.

Init UHF
UHFNet.MessageClass.PowerFunc += new ReceivedPower(OnReceivedPower); bool Open()

```

{
    RFIDUHF.RFID_STATUS status = new RFIDUHF.RFID_STATUS();
    String strstatus = null;

    status = UHFNet.Init();
    if (status != RFIDUHF.RFID_STATUS.RFID_STATUS_OK)
    {
        strstatus = String.Format("RFID Init Error-{0:G}", status);
        return false;
    }
    label_Result.Text = "Open UHF";
    return true;
}

public void OnReceivedPower(bool a_bPowerOn)
{
    if (a_bPowerOn == true)
    {
        if (m_bOpen == false)
        {
            if (Open())
            {
                m_bOpen = true;
            }
        }
    }
    else
    {
        if (m_bOpen == true)
        {
            Close();
            m_bOpen = false;
        }
    }
}
}

```

Inventory Start

```

void Inventory()
{
    UHFNet.Inventory();
}

```

Get Data

```

UHFNet.MessageClass.InventoryFunc += new ReceivedInventory(OnReceivedInventory);

private void OnReceivedInventory()

```

```

{
    int nTaglenth = 0;
    String strTagData = null;
    StringBuilder strData = new StringBuilder(260);

    nTaglenth = UHFNet.GetData(strData);

    if (checkBox_CRC.Checked == true)
    {
        strTagData = strData.ToString().Substring(0, nTaglenth);
    }
    else if (checkBox_CRC.Checked == false)
    {
        strTagData = strData.ToString().Substring(0, nTaglenth - 2);
    }
}

```

Inventory Stop

```

void InventoryStop()
{
    UHFNet.InventoryStop();
}

```

Close UHF

```

bool Close()
{
    RFIDUHF.RFID_STATUS status = new RFIDUHF.RFID_STATUS();
    String strstatus = null;

    status = UHFNet.Close();
    if (status != RFIDUHF.RFID_STATUS.RFID_STATUS_OK)
    {
        strstatus = String.Format("{0:G}", status);

        return false;
    }
    label_Result.Text = "Close UHF";
    return true;
}

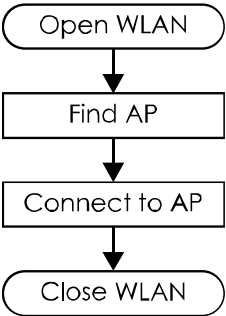
```

3.7 WLAN

Supported PDA:

Brand	Restriction
M3 OX10	WLAN must be SUMMIT

Basic WLAN flow chart:



Tip !!

- I. WLANTest.exe functions as same as SCU (Summit Client Utility) and they are synchronized.
- II. Combined version of WLAN.dll can be used in all M3 Summit devices regardless of model type or OS.

Init Wlan
<pre>WLANNet.WLANNet Wlan = new WLANNet.WLANNet(); if(Wlan.Init() == false) { MessageBox.Show("Fail To WLAN_Init()"); }</pre>
Searching AP
<pre>WLAN_SSID[] ssidlist = new WLAN_SSID[40]; Cursor.Current = Cursors.WaitCursor; do { Wlan.GetBssidList(ref ssidlist); } while (ssidlist[0].SSIDName == "");</pre>
Connect Config
<pre>bool result=false; // Security result= Wlan.ConnectAP("SSID"); if(result==false) MessageBox.Show("Fail to ConnectAP()"); // Open</pre>

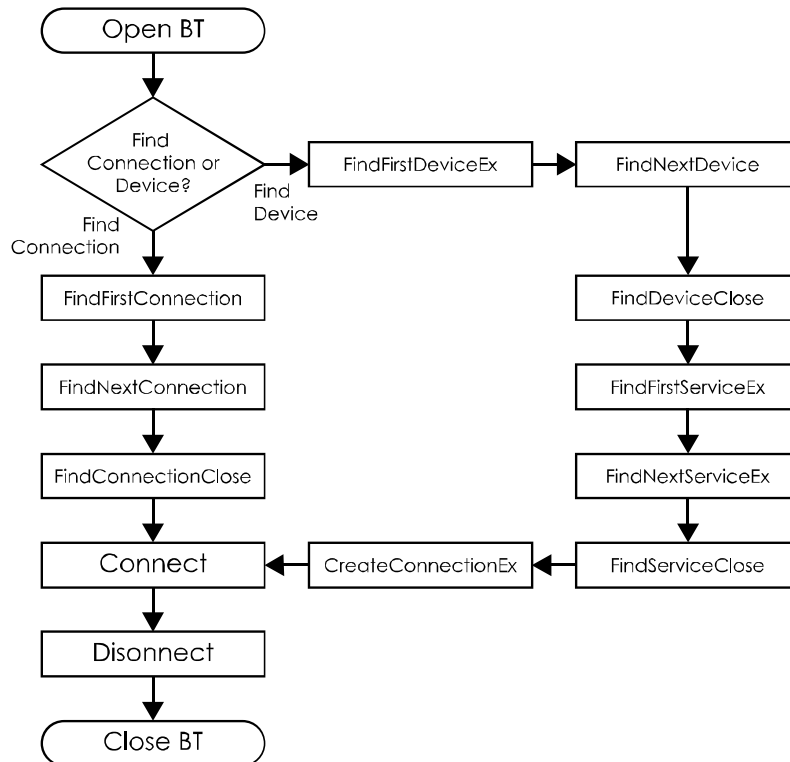
<pre> result=Wlan.ConnectAPEX("SSID", "1234", 1, 2); if(result==false) MessageBox.Show("Fail to ConnectAPEX()"); </pre>
Delete Config
<pre> bool result= Wlan.DeleteConfig("SSID"); if(!result) MessageBox.Show("Activate config can not removed!!!"); </pre>
Change Config
<pre> bool result=Wlan.ActivateConfig("SSID"); if(!result) MessageBox.Show("Fail To ActivateConfig()"); </pre>
Export Configuration
<pre> if (FileDialog.ShowDialog() == DialogResult.OK) m_filepath.Text = FileDialog.FileName; if (Wlan.ExportConfig(m_filepath.Text)) m_result.Text="Export Success!!!"; else m_result.Text = "Export Fail!!!"; </pre>
WLAN Power On
<pre> bool result = Wlan.PowerOn(); if(!result) MessageBox.Show("Fail To PowerOn()"); </pre>
WLAN Power Off
<pre> bool result = Wlan.PowerOff(); if(!result) MessageBox.Show("Fail To PowerOff()"); </pre>
Import Configuration
<pre> if (FileDialog2.ShowDialog() == DialogResult.OK) m_filepath.Text = FileDialog2.FileName; if (Wlan.ImportConfig(m_filepath.Text)) m_result.Text="Import Success!!!"; else m_result.Text = "Import Fail!!!"; </pre>
Close WLAN
<pre> bool result=Wlan.Close(); if(!result) MessageBox.Show("Fail To Close ()"); </pre>

3.8 BLUETOOTH

Supported PDA:

Brand	Restriction
M3 OX10	BTExplorer version 2.1.1 and Build version 27460 or higher

Basic Bluetooth flow chart:



Tip !!

- I. Bluetooth SDK can be applied to upper level from StonestreetOne BTExplorer Version 2.1.1 and Build Version 27460.
- II. Bluetooth communicates through Serial with COM port 9.
- III. BT can provide following services;
 - AVC Audio/Video Control Transport Protocol Sample Application
 - AVR Audio/Video Remove Control Profile Sample Application
 - BTC Generic Bluetooth COM Profile Sample Application
 - DUN Dial-Up Networking Profile Sample Application
 - FTP OBEX File Transfer Profile Sample Application
 - GAV Generic Audio/Video Profile Sample Application
 - HDS Headset Profile Sample Application
 - OBP OBEX Object Push Profile Sample Application
 - PAN Personal Area Networking Profile Sample Application
 - SDP Sample SDP Application using Bluetopia
 - SPP Sample SPP Application using Bluetopia

Initialization

```
//Bluetooth Open
private void button_Open_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    if (Bluetooth.Open())
    {
        label_State_View.Text = "Open Success";
        g_bBluetoothOpen = true;
        button_Open.Enabled = false;
        button_Device_Find.Enabled = true;
        button_Service_Find.Enabled = false;
        button_Create_Connection.Enabled = false;
        button_Conn_Find.Enabled = true;
        button_Connect.Enabled = false;
        button_Disconnect.Enabled = false;
        button_Delete_Connection.Enabled = false;
        button_Close.Enabled = true;
        button_All_Of_Delete_Device.Enabled = true;
        button_Local_Info.Enabled = true;
    }
    else
    {
        label_State_View.Text = "Open Fail";
    }
}
```

Device Find

```
//Bluetooth Device Find
private void button_Device_Find_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    m_DeviceItem.Clear();
    if (g_bBluetoothOpen == true)
    {
        m_ListType = ListType.DeviceFind;
        BT_Device_Find DeviceFind = new BT_Device_Find();
        BluetoothNet.BT_Device_Info_t DeviceInfo = new BluetoothNet.BT_Device_Info_t();
        BluetoothNet.BT_Device_Query_Ex_t DeviceQuery = new BluetoothNet.BT_Device_Query_Ex_t();
        BluetoothNet.BT_Device_Info_t DeviceInfo_temp;
        String strName;
        String strAddr;

        m_ConnectionInfo.Size = (ushort)(Marshal.SizeOf(typeof(BluetoothNet.BT_Connection_Info_Ex_t)));
        m_ConnectionInfo.ProfileType = BluetoothNet.BT_Profile_Type.BT_PROFILE_SPP;
    }
}
```

```

DeviceQuery.Size = (ushort)(Marshal.SizeOf(typeof(BlueToothNet.BT_Device_Query_Ex_t)));
DeviceQuery.DeviceAttributes = (UInt16)BlueToothNet.BT_DEVICE_ALL;

DeviceQuery.DeviceType = (m_ConnectionInfo.ProfileType == BlueToothNet.BT_Profile_Type.BT_PROFILE_SPP ?
BlueToothNet.BT_DeviceType_t.bdAll : BlueToothNet.BT_DeviceType_t.bdHID);

DeviceQuery.InquiryTimeout = 10;
Cursor.Current = Cursors.WaitCursor;

hResult = Bluetooth.FindFirstDeviceEx(ref DeviceFind, ref DeviceInfo, ref DeviceQuery);
if (BlueToothNet.BT_ERROR_SUCCESS == hResult)
{
    Cursor.Current = Cursors.Default;
    do
    {
        DeviceInfo_temp = DeviceInfo;
        m_DeviceItem.Insert(0, DeviceInfo_temp);

        hResult = Bluetooth.FindNextDevice(DeviceFind, ref DeviceInfo);
    } while (BlueToothNet.BT_ERROR_SUCCESS == hResult);
}

Bluetooth.FindDeviceClose(DeviceFind);
int ImportDevicecount = m_DeviceItem.Count;
int i = 0;
for (i = ImportDevicecount - 1; i >= 0; i--)
{
    DeviceInfo = m_DeviceItem[i];
    strName = String.Format("{0:G}", Encoding.Default.GetString(DeviceInfo.Name, 0,
(BlueToothNet.MAX_NAME_LENGTH + 1)));

    strAddr = String.Format("{0:X2}:{1:X2}:{2:X2}:{3:X2}:{4:X2}:{5:X2}",
DeviceInfo.BD_ADDR.BD_ADDR5,
DeviceInfo.BD_ADDR.BD_ADDR4,
DeviceInfo.BD_ADDR.BD_ADDR3,
DeviceInfo.BD_ADDR.BD_ADDR2,
DeviceInfo.BD_ADDR.BD_ADDR1,
DeviceInfo.BD_ADDR.BD_ADDR0);

    ListViewItem list_View = new ListViewItem(strName);
    list_View.SubItems.Add(strAddr);
    listView_Info.Items.Insert(0, list_View);
}

label_State_View.Text = "Device Find";
g_bAllDelDevice = false;
button_Open.Enabled = false;
button_Device_Find.Enabled = false;
button_Service_Find.Enabled = true;
button_Create_Connection.Enabled = false;
button_Conn_Find.Enabled = true;
button_Connect.Enabled = false;
button_Disconnect.Enabled = false;
button_Delete_Connection.Enabled = false;

```

```

        button_Close.Enabled = true;

        button_All_Of_Delete_Device.Enabled = true;

        button_Local_Info.Enabled = true;

    }
}

```

Service Find

```

//Bluetooth Service Find
private void button_Service_Find_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    if (g_bBluetoothOpen == true)
    {
        BT_Service_Find ServiceFind = new BT_Service_Find();
        BluetoothNet.BT_Service_Info_Ex_t ServiceInfo = new BluetoothNet.BT_Service_Info_Ex_t();
        BluetoothNet.BT_Service_Query_t ServiceQuery = new BluetoothNet.BT_Service_Query_t();
        BluetoothNet.SDP_UUID_Entry_t Service = new BluetoothNet.SDP_UUID_Entry_t();
        String strName;
        m_ListType = ListType.ServiceFind;
        ServiceInfo.Size = (ushort)(Marshal.SizeOf(typeof(BluetoothNet.BT_Service_Info_Ex_t)));
        ServiceQuery.BD_ADDR = m_ConnectionInfo.BD_ADDR;
        ServiceQuery.NumberServiceUUID = 1;
        Service.SDP_Data_Element_Type = BluetoothNet.SDP_Data_Element_Type_t.deUUID_16;
        switch (m_ConnectionInfo.ProfileType)
        {
            case BluetoothNet.BT_Profile_Type.BT_PROFILE_SPP:
                if (128 == serviceUUID)
                {
                    Service.SDP_Data_Element_Type = BluetoothNet.SDP_Data_Element_Type_t.deUUID_128;
                    Bluetooth.ASSIGN_UUID_128(ref (Service.value.UUID_128),
                        0x85, 0x60, 0xCA, 0x18,
                        0x62, 0x3F,
                        0x4A, 0x77,
                        0x9F, 0x5E, 0x3C, 0x52, 0x66, 0x68, 0x05, 0x1A);
                }
                else
                {
                    Bluetooth.ASSIGN_UUID_16(ref (Service.value.UUID_16), 0x11, 0x01);
                }
                break;
            case BluetoothNet.BT_Profile_Type.BT_PROFILE_HID_HOST:
            case BluetoothNet.BT_Profile_Type.BT_PROFILE_HID_DEVICE:
                Bluetooth.ASSIGN_UUID_16(ref (Service.value.UUID_16), 0x11, 0x24);
                break;
        }
        Cursor.Current = Cursors.WaitCursor;
    }
}

```

```

ServiceQuery.Service = Marshal.AllocCoTaskMem(Marshal.SizeOf(Service));
Marshal.StructureToPtr(Service, ServiceQuery.Service, false);
hResult = Bluetooth.FindFirstServiceEx(ref ServiceFind, ref ServiceInfo, ref ServiceQuery);
if (BluetoothNet.BT_ERROR_SUCCESS == hResult)
{
    Cursor.Current = Cursors.Default;
    do
    {
        m_ConnectionInfo.ProfileType = ServiceInfo.ProfileType;
        m_ConnectionInfo.MajorVersion = ServiceInfo.MajorVersion;
        m_ConnectionInfo.MinorVersion = ServiceInfo.MinorVersion;
        switch (m_ConnectionInfo.ProfileType)
        {
            case BluetoothNet.BT_Profile_Type.BT_PROFILE_UNKNOWN:
m_ConnectionInfo._profileinformation.RemoteSerialPortProfileInfo.RFCOMMServerPort =
ServiceInfo._profileinformation.RemoteSerialPortProfileInfo.RFCOMMServerPort;
m_ConnectionInfo._profileinformation.RemoteSerialPortProfileInfo.UseActiveSync =
ServiceInfo._profileinformation.RemoteSerialPortProfileInfo.UseActiveSync;

                break;

            case BluetoothNet.BT_Profile_Type.BT_PROFILE_SPP:
m_ConnectionInfo._profileinformation.RemoteSerialPortProfileInfo.RFCOMMServerPort =
ServiceInfo._profileinformation.RemoteSerialPortProfileInfo.RFCOMMServerPort;
m_ConnectionInfo._profileinformation.RemoteSerialPortProfileInfo.UseActiveSync =
ServiceInfo._profileinformation.RemoteSerialPortProfileInfo.UseActiveSync;

                break;

            case BluetoothNet.BT_Profile_Type.BT_PROFILE_HID_HOST:
            case BluetoothNet.BT_Profile_Type.BT_PROFILE_HID_DEVICE:
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.DeviceAutomaticReconnect =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.DeviceAutomaticReconnect;
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.DeviceNormallyConnectable =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.DeviceNormallyConnectable;
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.DeviceSubclass =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.DeviceSubclass;
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.L2CAPControlChannel =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.L2CAPControlChannel;
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.L2CAPInterruptChannel =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.L2CAPInterruptChannel;
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.VirtualCableSupported =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.VirtualCableSupported;

                break;
        }

        strName = String.Format("{0:G}", Encoding.Default.GetString(ServiceInfo.ServiceName, 0, 249));
        ListViewItem list_View = new ListViewItem(strName);
        listView_Info.Items.Insert(0, list_View);

        hResult = Bluetooth.FindNextServiceEx(ServiceFind, ref ServiceInfo);
    } while (BluetoothNet.BT_ERROR_SUCCESS == hResult);
    if (BluetoothNet.BT_ERROR_NO_MORE != hResult)
    {
        label_State_View.Text = "Failed to find next service";
    }
}

```

```

    }
    Bluetooth.FindServiceClose(ServiceFind);
}
label_State_View.Text = "Service Find";
button_Open.Enabled = false;
button_Device_Find.Enabled = true;
button_Service_Find.Enabled = false;
button_Create_Connection.Enabled = true;
button_Conn_Find.Enabled = true;
button_Connect.Enabled = false;
button_Disconnect.Enabled = false;
button_Delete_Connection.Enabled = false;
button_Close.Enabled = true;
button_All_Of_Delete_Device.Enabled = false;
button_Local_Info.Enabled = true;
}
else if (g_bAllDelDevice == true)
{
    label_State_View.Text = "Please, 'Device Find' is re-click.";
}
}

```

Connection Management

```

// Bluetooth Connection Management - (1) Create Connection
private void button_Create_Connection_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    if (g_bBluetoothOpen == true)
    {
        m_ConnectionInfo.ConnectionAttributes = BluetoothNet.BT_CONNECTION_REMEMBERED |
BluetoothNet.BT_CONNECTION_ACTIVE;
        m_ConnectionInfo.LocalCOMPort = 9;
        m_ConnectionInfo.ProfileType = BluetoothNet.BT_Profile_Type.BT_PROFILE_SPP;
        hResult = Bluetooth.CreateConnectionEx(ref m_ConnectionInfo);
        if (BluetoothNet.BT_ERROR_SUCCESS == hResult)
        {
            label_State_View.Text = "Create Connection";
        }
        else
        {
            label_State_View.Text = "Create Connection Error";
        }
        button_Open.Enabled = false;
        button_Device_Find.Enabled = true;
        button_Service_Find.Enabled = true;
        button_Create_Connection.Enabled = false;
    }
}

```

```

        button_Conn_Find.Enabled = true;
        button_Connect.Enabled = false;
        button_Disconnect.Enabled = false;
        button_Delete_Connection.Enabled = false;
        button_Close.Enabled = true;
        button_All_Of_Delete_Device.Enabled = false;
        button_Local_Info.Enabled = true;
    }
}

// Bluetooth Connection Management - (2) Delete Connection
private void button_Delete_Connection_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    if (g_bBluetoothOpen == true)
    {
        hResult = Bluetooth.DeleteConnection(m_ConnectionInfo.ConnectionID);
        if (BlueToothNet.BT_ERROR_SUCCESS == hResult)
        {
            label_State_View.Text = "Delete Connection";
        }
        else
        {
            label_State_View.Text = "Delete Connection Error";
        }
        button_Open.Enabled = false;
        button_Device_Find.Enabled = true;
        button_Service_Find.Enabled = false;
        button_Create_Connection.Enabled = false;
        button_Conn_Find.Enabled = true;
        button_Connect.Enabled = false;
        button_Disconnect.Enabled = false;
        button_Delete_Connection.Enabled = false;
        button_Close.Enabled = true;
        button_All_Of_Delete_Device.Enabled = true;
        button_Local_Info.Enabled = true;
    }
}

// Bluetooth Connection Management - (3) Connection Find
private void button_Conn_Find_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    m_DeviceItem.Clear();
    m_ConnectionItem.Clear();
    if (g_bBluetoothOpen == true)
    {
        m_ListType = ListType.ConnectionFind;
    }
}

```

```

String strID;
String strAddr;
BT_Connection_Find ConnectFind = new BT_Connection_Find();
BluetoothNet.BT_Connection_Info_t ConnectionInfo = new BluetoothNet.BT_Connection_Info_t();
BluetoothNet.BT_Connection_Query_t ConnectQuery = new BluetoothNet.BT_Connection_Query_t();
ConnectQuery.ConnectionAttributes = BluetoothNet.BT_CONNECTION_REMEMBERED;
Cursor.Current = Cursors.WaitCursor;
hResult = Bluetooth.FindFirstConnection(ref ConnectFind, ref ConnectionInfo, ref ConnectQuery);
if (BluetoothNet.BT_ERROR_NO_MORE == hResult)
{
    Cursor.Current = Cursors.Default;
    label_State_View.Text = "Have Not Connection";
    return;
}
else if (BluetoothNet.BT_ERROR_SUCCESS == hResult)
{
    Cursor.Current = Cursors.Default;
    do
    {
        BluetoothNet.BT_Connection_Info_t ConnectInfo_temp;
        ConnectInfo_temp = ConnectionInfo;
        m_ConnectionItem.Insert(0, ConnectInfo_temp);
        hResult = Bluetooth.FindNextConnection(ConnectFind, ref ConnectionInfo);
    } while (BluetoothNet.BT_ERROR_SUCCESS == hResult);
}
Bluetooth.FindConnectionClose(ConnectFind);
int ImportDeviceCount = m_ConnectionItem.Count;
int i = 0;
for (i = m_ConnectionItem.Count - 1; i >= 0; i--)
{
    ConnectionInfo = m_ConnectionItem[i];
    strID = String.Format("{0:}", ConnectionInfo.ConnectionID);
    strAddr = String.Format("{0:X2}:{1:X2}:{2:X2}:{3:X2}:{4:X2}:{5:X2}",
        ConnectionInfo.BD_ADDR.BD_ADDR5,
        ConnectionInfo.BD_ADDR.BD_ADDR4,
        ConnectionInfo.BD_ADDR.BD_ADDR3,
        ConnectionInfo.BD_ADDR.BD_ADDR2,
        ConnectionInfo.BD_ADDR.BD_ADDR1,
        ConnectionInfo.BD_ADDR.BD_ADDR0);
    ListViewItem list_View = new ListViewItem(strID);
    list_View.SubItems.Add(strAddr);
    listView_Info.Items.Insert(0, list_View);
}
label_State_View.Text = "Connection Find";
button_Open.Enabled = false;
button_Device_Find.Enabled = false;

```



```

        button_Service_Find.Enabled = false;
        button_Create_Connection.Enabled = false;
        button_Conn_Find.Enabled = false;
        button_Connect.Enabled = true;
        button_Disconnect.Enabled = false;
        button_Delete_Connection.Enabled = true;
        button_Close.Enabled = true;
        button_All_Of_Delete_Device.Enabled = false;
        button_Local_Info.Enabled = true;
    }
}

```

Connect

```

// Bluetooth Connect
private void button_Connect_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    if (g_bBluetoothOpen == true)
    {
        hResult = Bluetooth.Connect(m_ConnectionInfo.ConnectionID);
        CreateFile("COM9:", GENERIC_READ | GENERIC_WRITE, 0, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
        if (BluetoothNet.BT_ERROR_SUCCESS == hResult)
        {
            label_State_View.Text = "Connect";
        }
        else
        {
            label_State_View.Text = "Connect Error";
        }
        button_Open.Enabled = false;
        button_Device_Find.Enabled = true;
        button_Service_Find.Enabled = false;
        button_Create_Connection.Enabled = false;
        button_Conn_Find.Enabled = false;
        button_Connect.Enabled = false;
        button_Disconnect.Enabled = true;
        button_Delete_Connection.Enabled = false;
        button_Close.Enabled = true;
        button_All_Of_Delete_Device.Enabled = false;
        button_Local_Info.Enabled = true;
    }
}

```

Disconnect

```

//Bluetooth Disconnect
private void button_Disconnect_Click(object sender, EventArgs e)

```

```

{
    listView_Info.Items.Clear();
    if (g_bBluetoothOpen == true)
    {
        hResult = Bluetooth.Disconnect(m_ConnectionInfo.ConnectionID);
        if (BluetoothNet.BT_ERROR_SUCCESS == hResult)
        {
            label_State_View.Text = "Disconnect";
        }
        else
        {
            label_State_View.Text = "Disconnect Error";
        }
        button_Open.Enabled = false;
        button_Device_Find.Enabled = true;
        button_Service_Find.Enabled = false;
        button_Create_Connection.Enabled = false;
        button_Conn_Find.Enabled = true;
        button_Connect.Enabled = false;
        button_Disconnect.Enabled = false;
        button_Delete_Connection.Enabled = true;
        button_Close.Enabled = true;
        button_All_Of_Delete_Device.Enabled = false;
        button_Local_Info.Enabled = true;
    }
}

```

Close

```

// Bluetooth Close
private void button_Close_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    if (g_bBluetoothOpen == true)
    {
        Bluetooth.Close();
        label_State_View.Text = "Close Success";
        button_Open.Enabled = true;
        button_Device_Find.Enabled = false;
        button_Service_Find.Enabled = false;
        button_Create_Connection.Enabled = false;
        button_Conn_Find.Enabled = false;
        button_Connect.Enabled = false;
        button_Disconnect.Enabled = false;
        button_Delete_Connection.Enabled = false;
        button_Close.Enabled = false;
        button_All_Of_Delete_Device.Enabled = false;
    }
}

```

```
button_Local_Info.Enabled = false;
```

```
}
```

```
g_bBluetoothOpen = false;
```

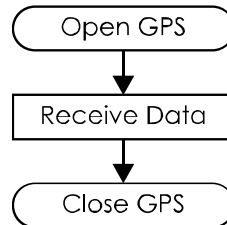
```
}
```

3.9 GPS

Supported PDA:

Brand	Restriction
M3 OX10	SirF III or UBLOX GPS must be installed

Basic GPS flow chart:



Tip !!

- I. GPS SDK supports SirF III and UBLOX module.
- II. Device that UBLOX module is installed supports AGPS SDK.
- III. The GPS can be used in stand-alone (conventional) or Assisted GPS (A-GPS) modes. A Stand-alone GPS receiver must download data from GPS satellites. It can take several minutes to get a fix. By using GPS Location servers, A-GPS dramatically improves the performance of the TTFF (Time To First Fix) of GPS receivers by providing them with data that they would ordinarily have to download from the GPS satellites. With the A-GPS data, GPS receivers can operate faster and more reliably.
- IV. GPS communicates through Serial with different COM port according to device's OS, M3 Mobile PDA with Windows Mobile OS uses COM 2 and Windows CE OS uses COM 3.

Open & Close

```
//Bluetooth Open
private void button_Open_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    if (Bluetooth.Open())
    {
        label_State_View.Text = "Open Success";
        g_bBluetoothOpen = true;
        button_Open.Enabled = false;
        button_Device_Find.Enabled = true;
        button_Service_Find.Enabled = false;
        button_Create_Connection.Enabled = false;
        button_Conn_Find.Enabled = true;
        button_Connect.Enabled = false;
        button_Disconnect.Enabled = false;
        button_Delete_Connection.Enabled = false;
    }
}
```

```

        button_Close.Enabled = true;

        button_All_Of_Delete_Device.Enabled = true;

        button_Local_Info.Enabled = true;

    }
    else
    {
        label_State_View.Text = "Open Fail";
    }
}

```

Receive Message from GPS Module

```

//Bluetooth Device Find
private void button_Device_Find_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    m_DeviceItem.Clear();
    if (g_bBluetoothOpen == true)
    {
        m_ListType = ListType.DeviceFind;
        BT_Device_Find DeviceFind = new BT_Device_Find();
        BluetoothNet.BT_Device_Info_t DeviceInfo = new BluetoothNet.BT_Device_Info_t();
        BluetoothNet.BT_Device_Query_Ex_t DeviceQuery = new BluetoothNet.BT_Device_Query_Ex_t();
        BluetoothNet.BT_Device_Info_t DeviceInfo_temp;
        String strName;
        String strAddr;
        m_ConnectionInfo.Size = (ushort)(Marshal.SizeOf(typeof(BluetoothNet.BT_Connection_Info_Ex_t)));
        m_ConnectionInfo.ProfileType = BluetoothNet.BT_Profile_Type.BT_PROFILE_SPP;
        DeviceQuery.Size = (ushort)(Marshal.SizeOf(typeof(BluetoothNet.BT_Device_Query_Ex_t)));
        DeviceQuery.DeviceAttributes = (UInt16)BluetoothNet.BT_DEVICE_ALL;
        DeviceQuery.DeviceType = (m_ConnectionInfo.ProfileType == BluetoothNet.BT_Profile_Type.BT_PROFILE_SPP ?
BluetoothNet.BT_DeviceType_t.bdAll : BluetoothNet.BT_DeviceType_t.bdHID);
        DeviceQuery.InquiryTimeout = 10;
        Cursor.Current = Cursors.WaitCursor;
        hResult = Bluetooth.FindFirstDeviceEx(ref DeviceFind, ref DeviceInfo, ref DeviceQuery);
        if (BluetoothNet.BT_ERROR_SUCCESS == hResult)
        {
            Cursor.Current = Cursors.Default;
            do
            {
                DeviceInfo_temp = DeviceInfo;
                m_DeviceItem.Insert(0, DeviceInfo_temp);
                hResult = Bluetooth.FindNextDevice(DeviceFind, ref DeviceInfo);
            } while (BluetoothNet.BT_ERROR_SUCCESS == hResult);
        }
        Bluetooth.FindDeviceClose(DeviceFind);
        int ImportDevicecount = m_DeviceItem.Count;
    }
}

```

```

int i = 0;
for (i = ImportDevicecount - 1; i >= 0; i--)
{
    DeviceInfo = m_DeviceItem[i];
    strName = String.Format("{0:G}", Encoding.Default.GetString(DeviceInfo.Name, 0,
(BluetoothNet.MAX_NAME_LENGTH + 1)));
    strAddr = String.Format("{0:X2}:{1:X2}:{2:X2}:{3:X2}:{4:X2}:{5:X2}",
    DeviceInfo.BD_ADDR.BD_ADDR5,
    DeviceInfo.BD_ADDR.BD_ADDR4,
    DeviceInfo.BD_ADDR.BD_ADDR3,
    DeviceInfo.BD_ADDR.BD_ADDR2,
    DeviceInfo.BD_ADDR.BD_ADDR1,
    DeviceInfo.BD_ADDR.BD_ADDR0);
    ListViewItem list_View = new ListViewItem(strName);
    list_View.SubItems.Add(strAddr);
    listView_Info.Items.Insert(0, list_View);
}
label_State_View.Text = "Device Find";
g_bAllDelDevice = false;
button_Open.Enabled = false;
button_Device_Find.Enabled = false;
button_Service_Find.Enabled = true;
button_Create_Connection.Enabled = false;
button_Conn_Find.Enabled = true;
button_Connect.Enabled = false;
button_Disconnect.Enabled = false;
button_Delete_Connection.Enabled = false;
button_Close.Enabled = true;
button_All_Of_Delete_Device.Enabled = true;
button_Local_Info.Enabled = true;
}
}

```

AGPS

```

//Bluetooth Service Find
private void button_Service_Find_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    if (g_bBluetoothOpen == true)
    {
        BT_Service_Find ServiceFind = new BT_Service_Find();
        BluetoothNet.BT_Service_Info_Ex_t ServiceInfo = new BluetoothNet.BT_Service_Info_Ex_t();
        BluetoothNet.BT_Service_Query_t ServiceQuery = new BluetoothNet.BT_Service_Query_t();
        BluetoothNet.SDP_UUID_Entry_t Service = new BluetoothNet.SDP_UUID_Entry_t();
        String strName;
    }
}

```

```

m_ListType = ListType.ServiceFind;
ServiceInfo.Size = (ushort)(Marshal.SizeOf(typeof(BluetoothNet.BT_Service_Info_Ex_t)));
ServiceQuery.BD_ADDR = m_ConnectionInfo.BD_ADDR;
ServiceQuery.NumberServiceUUID = 1;
Service.SDP_Data_Element_Type = BluetoothNet.SDP_Data_Element_Type_t.deUUID_16;
switch (m_ConnectionInfo.ProfileType)
{
    case BluetoothNet.BT_Profile_Type.BT_PROFILE_SPP:
        if (128 == serviceUUID)
        {
            Service.SDP_Data_Element_Type = BluetoothNet.SDP_Data_Element_Type_t.deUUID_128;
            Bluetooth.ASSIGN_UUID_128(ref (Service.value.UUID_128),
                0x85, 0x60, 0xCA, 0x18,
                0x62, 0x3F,
                0x4A, 0x77,
                0x9F, 0x5E, 0x3C, 0x52, 0x66, 0x68, 0x05, 0x1A);
        }
        else
        {
            Bluetooth.ASSIGN_UUID_16(ref (Service.value.UUID_16), 0x11, 0x01);
        }
        break;
    case BluetoothNet.BT_Profile_Type.BT_PROFILE_HID_HOST:
    case BluetoothNet.BT_Profile_Type.BT_PROFILE_HID_DEVICE:
        Bluetooth.ASSIGN_UUID_16(ref (Service.value.UUID_16), 0x11, 0x24);
        break;
}
Cursor.Current = Cursors.WaitCursor;
ServiceQuery.Service = Marshal.AllocCoTaskMem(Marshal.SizeOf(Service));
Marshal.StructureToPtr(Service, ServiceQuery.Service, false);
hResult = Bluetooth.FindFirstServiceEx(ref ServiceFind, ref ServiceInfo, ref ServiceQuery);
if (BluetoothNet.BT_ERROR_SUCCESS == hResult)
{
    Cursor.Current = Cursors.Default;
    do
    {
        {
            m_ConnectionInfo.ProfileType = ServiceInfo.ProfileType;
            m_ConnectionInfo.MajorVersion = ServiceInfo.MajorVersion;
            m_ConnectionInfo.MinorVersion = ServiceInfo.MinorVersion;
            switch (m_ConnectionInfo.ProfileType)
            {
                case BluetoothNet.BT_Profile_Type.BT_PROFILE_UNKNOWN:
m_ConnectionInfo._profileinformation.RemoteSerialPortProfileInfo.RFCOMMServerPort =
ServiceInfo._profileinformation.RemoteSerialPortProfileInfo.RFCOMMServerPort;
m_ConnectionInfo._profileinformation.RemoteSerialPortProfileInfo.UseActiveSync =
ServiceInfo._profileinformation.RemoteSerialPortProfileInfo.UseActiveSync;

```

```

        break;

        case BluetoothNet.BT_Profile_Type.BT_PROFILE_SPP:
m_ConnectionInfo._profileinformation.RemoteSerialPortProfileInfo.RFCommServerPort =
ServiceInfo._profileinformation.RemoteSerialPortProfileInfo.RFCommServerPort;
m_ConnectionInfo._profileinformation.RemoteSerialPortProfileInfo.UseActiveSync =
ServiceInfo._profileinformation.RemoteSerialPortProfileInfo.UseActiveSync;

        break;

        case BluetoothNet.BT_Profile_Type.BT_PROFILE_HID_HOST:

        case BluetoothNet.BT_Profile_Type.BT_PROFILE_HID_DEVICE:
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.DeviceAutomaticReconnect =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.DeviceAutomaticReconnect;
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.DeviceNormallyConnectable =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.DeviceNormallyConnectable;
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.DeviceSubclass =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.DeviceSubclass;
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.L2CAPControlChannel =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.L2CAPControlChannel;
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.L2CAPInterruptChannel =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.L2CAPInterruptChannel;
m_ConnectionInfo._profileinformation.RemoteHIDProfileInfo.VirtualCableSupported =
ServiceInfo._profileinformation.RemoteHIDProfileInfo.VirtualCableSupported;

        break;
    }

    strName = String.Format("{0:G}", Encoding.Default.GetString(ServiceInfo.ServiceName, 0, 249));
    ListViewItem list_View = new ListViewItem(strName);
    listView_Info.Items.Insert(0, list_View);
    hResult = Bluetooth.FindNextServiceEx(ServiceFind, ref ServiceInfo);
} while (BluetoothNet.BT_ERROR_SUCCESS == hResult);
if (BluetoothNet.BT_ERROR_NO_MORE != hResult)
{
    label_State_View.Text = "Failed to find next service";
}
Bluetooth.FindServiceClose(ServiceFind);
}
label_State_View.Text = "Service Find";
button_Open.Enabled = false;
button_Device_Find.Enabled = true;
button_Service_Find.Enabled = false;
button_Create_Connection.Enabled = true;
button_Conn_Find.Enabled = true;
button_Connect.Enabled = false;
button_Disconnect.Enabled = false;
button_Delete_Connection.Enabled = false;
button_Close.Enabled = true;
button_All_Of_Delete_Device.Enabled = false;
button_Local_Info.Enabled = true;
}
else if (g_bAllDelDevice == true)

```

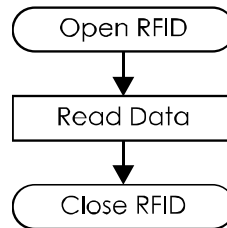


```
{  
    label_State_View.Text = "Please, 'Device Find' is re-click.";  
}  
}
```

3.10 RFID (HF)

Supported PDA: M3 POS only

Please refer to below flow chart for RFID.



Open RFID

```
private void btn_rfid_open_Click(object sender, EventArgs e)
{
    if (!g_OnOffStatus)
    {
        POS_RFID.RFID_Open();
        btn_rfid_open.Enabled = false;
        btn_rfid_close.Enabled = true;
        label_rfid_status.Text = "RFID Open Success!!";
        POS_RFID.RFID_ReadMode(true, m_hWnd);
        Thread.Sleep(500);
        g_OnOffStatus = true;
    }
    else
        label_rfid_status.Text = "RFID Open Fail!!";
}
```

RFID Data

```
public long OnDisplayCard(IntPtr wParam, IntPtr lParam)
{
    //IntPtr wParam = new IntPtr(1000);
    uint i;
    byte[] Tmp = new byte[64];
    string str;
    tRfMsg rfMsg = (tRfMsg)Marshal.PtrToStructure(wParam, typeof(tRfMsg));
    textBox_rfid_flag.Text = "";
    textBox_rfid_colum.Text = "";
    if ((rfMsg.CardLowStatus & POSNet.POS.RFCARDMASK) == POSNet.POS.RF_TIMEOUT)
    {
        POS_RFID.RFID_ReadMode(false, IntPtr.Zero);
        POS_RFID.RFID_Close();
        textBox_rfid_status.Text = "";
        textBox_rfid_count.Text = "";
        textBox_rfid_type.Text = "";
        textBox_rfid_info.Text = "";
    }
}
```

```

        textBox_rfid_serial.Text = "";
        textBox_rfid_colum.Text = "";
        textBox_rfid_flag.Text = "";
        label_rfid_status.Text = "No Reader Response";
        return 0;
    }

    str = String.Format("{0}/{0}", rfMsg.CardOKProcesscount, rfMsg.CardTransTotalCount);
    textBox_rfid_count.Text = str;
    str = String.Format("{0:X}", (byte)rfMsg.CardLowStatus);
    textBox_rfid_status.Text = str;
    // Card Type
    str = "";
    if (rfMsg.CardLowStatus == 0x00)
    {
        if (rfMsg.CardType == POSNet.POS.TYPE_ISO14443_A)
        {
            if (rfMsg.CardISO == POSNet.POS.ISO14443_3)
                str = "ISO14443-3 A";
            else
                str = "ISO14443-4 B";
            POS_RFID.MSR_SetPlaySound(true);
        }
        else if (rfMsg.CardType == POSNet.POS.TYPE_ISO14443_B)
        {
            if (rfMsg.CardISO == POSNet.POS.ISO14443_3)
                str = "ISO14443-3 B";
            else
                str = "ISO14443-4 B";
            POS_RFID.MSR_SetPlaySound(true);
        }
        else if (rfMsg.CardType == POSNet.POS.TYPE_ISO15693)
        {
            str = "ISO15693";
            POS_RFID.MSR_SetPlaySound(true);
        }
        else
        {
            str = "";
        }
    }
    else if (((rfMsg.CardType & POSNet.POS.CARDTYPEMASK) == POSNet.POS.TYPE_ISO15693) && rfMsg.CardLowStatus == 0xE7)
    {
        str = "ISO15693";
    }
    else

```

```

{
    str = "";
}
textBox_rfid_type.Text = str;
str = "";
if (rfMsg.CardType == POSNet.POS.TYPE_ISO15693)
{
    str = String.Format("{0:X2}", (byte)rfMsg.ColPos);
    textBox_rfid_column.Text = str;
    str = String.Format("{0:X2}", (byte)rfMsg.Flag);
    textBox_rfid_flag.Text = str;
}
str = "";
if (rfMsg.CardLowStatus == 0x00)
{
    if (rfMsg.CardInfo == POSNet.POS.MIFARE_1K)
        str = "MIFARE 1K";
    else if (rfMsg.CardInfo == POSNet.POS.MIFARE_4K)
        str = "MIFARE 4K";
    else if (rfMsg.CardInfo == POSNet.POS.MIFARE_UL)
        str = "MIFARE Ultralight";
    else if (rfMsg.CardInfo == POSNet.POS.PAYPASS)
        str = "PayPass";
    else if (rfMsg.CardInfo == POSNet.POS.PAYWAVE)
        str = "PayWave";
    else if (rfMsg.CardInfo == POSNet.POS.TMONEY)
        str = "T-Money";
    else if (rfMsg.CardInfo == POSNet.POS.RF_ETC)
        str = "UnKnown";
    else str = "";
}
else
{
    str = "";
}
textBox_rfid_info.Text = str;
//
// UID
str = "";
if (((rfMsg.CardType & POSNet.POS.CARDTYPEMASK) == POSNet.POS.TYPE_ISO14443_B) && (rfMsg.CardLowStatus == 0x00))
{
    for (i = 0; i < 4; i++)
    {
        Tmp[4 - i - 1] = (byte)rfMsg.UID[i];
    }
}

```

```

        for (i = 0; i < 4; i++)
        {
            string temp;
            temp = String.Format("{0:X2}", Tmp[i]);
            str = str + temp;
        }
    }

    else if (((rfMsg.CardType & POSNet.POS.CARDTYPEMASK) == POSNet.POS.TYPE_ISO14443_A) && (rfMsg.CardLowStatus
== 0x00))
    {
        for (i = 0; i < rfMsg.UIDLength; i++)
        {
            Tmp[rfMsg.UIDLength - i - 1] = (byte)rfMsg.UID[i];
        }
        for (i = 0; i < rfMsg.UIDLength; i++)
        {
            string temp;
            temp = String.Format("{0:X2} ", Tmp[i]);
            str = str + temp;
        }
    }

    else if (((rfMsg.CardType & POSNet.POS.CARDTYPEMASK) == POSNet.POS.TYPE_ISO15693) &&
((rfMsg.CardLowStatus == 0x00) || (rfMsg.CardLowStatus == 0xE7)))
    {
        for (i = 0; i < rfMsg.UIDLength; i++)
        {
            Tmp[rfMsg.UIDLength - i - 1] = (byte)rfMsg.UID[i];
        }
        for (i = 0; i < rfMsg.UIDLength; i++)
        {
            string temp;
            temp = String.Format("{0:X2} ", Tmp[i]);
            str = str + temp;
        }
    }

    else
    {
        string temp = "";
        str = temp;
    }

    textBox_rfid_serial.Text = str;
    // serial/ UID
    rfMsg.CardDisplayStatus = 0;
    Thread.Sleep(500);
    return (long)1;
}

```

Close RFID

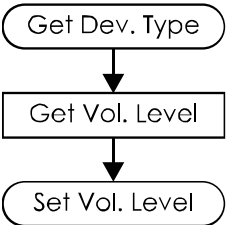
```
private void btn_rfid_close_Click(object sender, EventArgs e)
{
    if (g_OnOffStatus)
    {
        if (POS_RFID.RFID_Close())
        {
            label_rfid_status.Text = "RFID Close Success!!";
            g_OnOffStatus = false;
        }
        else
            label_rfid_status.Text = "RFID Close Fail!!";
    }
}
```

3.11 SYSTEM SDK

Supported PDA:

Brand	Restriction
M3 OX10	No restriction

Basic system SDK flow chart:



Get Device Type
<pre>m_System = new SystemNet.SystemNet(); DeviceInfo = m_System.GetDeviceInfo(); if (g_DeviceInfo == 5) { TB_MAINVOLUME.Maximum = 10; TB_MAINVOLUME.Minimum = 0; } else { TB_MAINVOLUME.Maximum = 5; TB_MAINVOLUME.Minimum = 0; }</pre>
Get Volume Level
<pre>if ((DeviceInfo == 5) (DeviceInfo == 3)) { LB_PHONEVOLUME.Enabled = false; TB_PHONEVOLUME.Enabled = false; } else { nPhoneVolumeLevel_Cur = m_System.GetPhoneVolumeLevel(); switch (nPhoneVolumeLevel_Cur) { case 0: nPhoneVolumeLevel_Cur = 5; break;</pre>

```

        case 1:
            nPhoneVolumeLevel_Cur = 4;
            break;

        case 2:
            nPhoneVolumeLevel_Cur = 3;
            break;

        case 3:
            nPhoneVolumeLevel_Cur = 2;
            break;

        case 4:
            nPhoneVolumeLevel_Cur = 1;
            break;

        case 5:
            nPhoneVolumeLevel_Cur = 0;
            break;

        default:
            nPhoneVolumeLevel_Cur = 3;
            break;
    }

    TB_PHONEVOLUME.Value = nPhoneVolumeLevel_Cur;
}

```

Set Volume Level

```

if (DeviceInfo == 5)
{
    switch (nPos)
    {
        case 0:
            nPos = 10;
            break;

        case 1:
            nPos = 9;
            break;

        case 2:
            nPos = 8;
            break;

        case 3:
            nPos = 7;
            break;

        case 4:
            nPos = 6;
            break;

        case 5:
            nPos = 5;
            break;

        case 6:

```



```
        nPos = 4;
        break;
    case 7:
        nPos = 3;
        break;
    case 8:
        nPos = 2;
        break;
    case 9:
        nPos = 1;
        break;
    case 10:
        nPos = 0;
        break;
    default:
        nPos = 3;
        break;
    }
}
else
{
    switch (nPos)
    {
        case 0:
            nPos = 5;
            break;
        case 1:
            nPos = 4;
            break;
        case 2:
            nPos = 3;
            break;
        case 3:
            nPos = 2;
            break;
        case 4:
            nPos = 1;
            break;
        case 5:
            nPos = 0;
            break;
        default:
            nPos = 3;
            break;
    }
}
```

```
m_System.SetVolumeLevel(nPos);
```

```
}
```

4 References

This chapter provides references of the modules included in M3 products. APIs are described using C/C++. Applications are written in VS2005.

Below table describes required files and supported M3 products.

Module	Required Files	Supported brand
SCANNER	Scanner.h Scanner.lib	All brands with 1D scanner
IMAGER	Imager.h Imager.lib	M3 T, M3 SMART WM, M3 SKY, MM3, M3 OX10 * Please see IMAGER tutorial for restrictions
LRSCANNER	LRScanner.h LRScanner.lib	MM3 only
CAMERA CE	Cam.h Cam.lib	All M3 CE units * Please see CAMERA (CE) tutorial for more details
CAMERA WM	Cam.h Cam.lib	All M3 WM units * Please see CAMERA (WM) tutorial for more details
RFID (LF / HF)	RFID.h RFID.lib	M3 SKY (LF / HF), M3 ORANGE (HF), M3 OX10 (HF)
UHF Gun		
WLAN	WLAN.h WLAN.lib	All brands with SUMMIT WLAN
BLUETOOTH	BLUETOOTH.h BLUETOOTH.lib	All brands with BT * Please see BLUETOOTH tutorial for more details
GPS	GPS.h GPS.lib	All brands with GPS *Please see GPS tutorial for more details
SYSTEM	System.h M3System.lib	All brands except M3 GREEN *Please see SYSTEM SDK tutorial for more details

4.1 SCANNER (1D)

Status Return value

Please refer to the below table for the status value definition.

Status Value Definition	Code	Meaning
TRUE	1	Success
FALSE	0	General Error

Event
public event ScannerDataDelegate ScannerDataEvent;
Enum
<pre>public enum SCAN_ENGINE_TYPE { OPTICON = 0, SYMBOL, INTERMEC, SYMBOL_HW, CINO , UNKNOWN, NOTYET }; public enum SCAN_DEVICE_TYPE { DEVICE_M3SKY = 0, DEVICE_M3SKYSAM, DEVICE_MM3, DEVICE_M3ORANGE, DEVICE_M3SMART_CE, DEVICE_M3SMART_WM, DEVICE_M3GREEN, DEVICE_M3T, DEVICE_M3POS, DEVICE_M3ORANGEPLUS, DEVICE_M3ORANGEPLUS_CE }; public enum SCAN_SOUND { SOUND_DEFAULT = 0, SOUND_BEEP, SOUND_NONE };</pre>

```
public enum SCAN_CODE39_FORMAT
```

```
{  
    CODE39_STANDARD = 0,  
    CODE39_CODE32,  
    CODE39_PZN,  
    CODE39_TRIOPTIC,  
    CODE39_FULLASCII  
};
```

Structure

```
public struct DECODER
```

```
{  
    // SYMBOLOGY 14  
    public bool bUPCA;  
    public bool bUPCE;  
    public bool bEAN13;  
    public bool bEAN8;  
    public bool bCODE39;  
    public bool bCODE128;  
    public bool bCODE93;  
    public bool bCODE11;  
    public bool bCODE25;  
    public bool bCODABAR;  
    public bool bKOREAPOST;  
    public bool bMSI;  
    public bool bGS1;  
    public bool bTELEPEN;  
  
    public DECODER(bool bUPCA, bool bUPCE, bool bEAN13, bool bEAN8, bool bCODE39, bool bCODE128, bool  
bCODE93, bool bCODE11, bool bCODE25, bool bCODABAR, bool bKOREAPOST, bool bMSI, bool bGS1, bool bTELEPEN);  
};
```

```
public struct DECODER_PARAMS
```

```
{  
    public bool bWindeScan;  
    public bool bHighFilter;  
    public bool bContinueMode;  
    public bool bXmitAimID;  
    public bool bVibrate;  
    public int nSound;  
    public int nTimeout;  
    public int nSecurityLevel;  
  
    public DECODER_PARAMS(bool bWindeScan, bool bHighFilter, bool bContinueMode, bool bXmitAimID, bool  
bVibrate, int nSound, int nTimeout, int nSecurityLevel);  
};
```

```
public struct UPCA_PARAMS
```

```
{  
    public bool bEnable;
```

```

    public bool bXNum;
    public bool bXCD;
    public bool bUPCA_AS_EAN13;
    public bool bAddOn;
    public UPCA_PARAMS(bool bEnable, bool bXNum, bool bXCD, bool bUPCA_AS_EAN13, bool bAddOn);
};

public struct UPCE_PARAMS
{
    public bool bEnable;
    public bool bXNum;
    public bool bXCD;
    public int nConvert;
    public UPCE_PARAMS(bool bEnable, bool bXNum, bool bXCD, int nConvert);
};

public struct EAN13_PARAMS
{
    public bool bEnable;
    public bool bBOOKLAND;
    public bool bXCD;
    public bool bAddOn;
    public EAN13_PARAMS(bool bEnable, bool bBOOKLAND, bool bXCD, bool bAddOn);
};

public struct EAN8_PARAMS
{
    public bool bEnable;
    public bool bXCD;
    public bool bEAN8_AS_EAN13;
    public EAN8_PARAMS(bool bEnable, bool bXCD, bool bEAN8_AS_EAN13);
};

public struct CODE39_PARAMS
{
    public bool bEnable;
    public int nFormat;
    public bool bCDV;
    public bool bXCD;
    public int nMinLen;
    public int nMaxLen;
    public CODE39_PARAMS(bool bEnable, int nFormat, bool bCDV, bool bXCD, int nMinLen, int nMaxLen);
};

public struct CODE128_PARAMS
{
    public bool bEnable;
    public bool bUCCEAN128;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string strFNC1_ASCII;
    public int nMinLen;

```

```

    public int      nMaxLen;
    public CODE128_PARAMS(bool bEnable, bool bUCCEAN128, string strFNC1_ASCII, int nMinLen, int nMaxLen);
};

public struct CODE93_PARAMS
{
    public bool bEnable;
    public int nMinLen;
    public int nMaxLen;
    public CODE93_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct KOREAPOST_PARAMS
{
    public bool bEnable;
    public KOREAPOST_PARAMS(bool bEnable);
};

public struct CODE11_PARAMS
{
    public bool bEnable;
    public bool bXCD;
    public int nCDV;
    public int nMinLen;
    public int nMaxLen;
    public CODE11_PARAMS(bool bEnable, bool bXCD, int nCDV, int nMinLen, int nMaxLen);
};

public struct CODE25_PARAMS
{
    public bool bl2OF5;
    public bool bs2OF5;
    public bool blTF14;
    public bool bMATRIX2OF5;
    public bool bCHINAPOST;
    public bool bINDUSTRY;
    public bool blATA;
    public bool bCDV;
    public bool bXCD;
    public int nMinLen;
    public int nMaxLen;
    public CODE25_PARAMS(bool bl2OF5, bool bs2OF5, bool blTF14, bool bMATRIX2OF5, bool bCHINAPOST, bool
bINDUSTRY, bool blATA, bool bCDV, bool bXCD, int nMinLen, int nMaxLen);
};

public struct CODABAR_PARAMS
{
    public bool bEnable;
    public bool bXSS;
    public int nMinLen;
    public int nMaxLen;

```

```
    public CODABAR_PARAMS(bool bEnable, bool bXSS, int nMinLen, int nMaxLen);
};

public struct MSI_PARAMS
{
    public bool bEnable;
    public bool bXCD;
    public int nCDV;
    public int nMinLen;
    public int nMaxLen;
    public MSI_PARAMS(bool bEnable, bool bXCD, int nCDV, int nMinLen, int nMaxLen);
};

public struct GS1_PARAMS
{
    public bool bGS1;
    public bool bGS1LIM;
    public bool bGS1EXP;
    public GS1_PARAMS(bool bGS1, bool bGS1LIM, bool bGS1EXP);
};

public struct TELEPEN_PARAMS
{
    public bool bEnable;
    public bool bNumeric;

    public TELEPEN_PARAMS(bool bEnable, bool bNumeric);
};
```


Functions for 1D Scanner

Name	Description
Close	Closes an open scanner
GetCODABAR	Gets the option of CODABAR Barcode
GetCODE11	Gets the option of CODE11 Barcode
GetCODE128	Gets the option of CODE128 Barcode
GetCODE25	Gets the option of CODE25 Barcode
GetCODE39	Gets the option of CODE39 Barcode
GetCODE93	Gets the option of CODE93 Barcode
GetDeviceType	Gets the type of device
GetEAN13	Gets the option of EAN-13 Barcode
GetEAN8	Gets the option of EAN-8 Barcode
GetEngineType	Gets the type of scanner engine
GetGS1	Gets the option of GS1 Barcode
GetKOREAPOST	Gets the option of KOREAPOST Barcode
GetMSI	Gets the option of MSI Barcode
GetOption	Gets the option of Scanner
GetScanData	Gets the ScanData which is read by scanner
GetSymbology	Gets Enable/Disable of Symbologies
GetTELEPEN	Gets the option of TELEPEN Barcode
GetUPCA	Gets the option of UPC-A Barcode
GetUPCE	Gets the option of UPC-E Barcode
GetVersionInfo	Gets the information of scanner engine and dll version
Open	Opens a scanner
Read	Starts the beaming of scanner
ReadCancel	Stops the beaming of scanner
RegHotKey	Registers Scanner Button as a hot key
SetCODABAR	Sets the option of CODABAR Barcode
SetCODE11	Sets the option of CODE11 Barcode
SetCODE128	Sets the option of CODE128 Barcode
SetCODE25	Sets the option of CODE25 Barcode
SetCODE39	Sets the option of CODE39 Barcode

<u>SetCODE93</u>	Sets the option of CODE93 Barcode
<u>SetEAN13</u>	Sets the option of EAN-13 Barcode
<u>SetEAN8</u>	Sets the option of EAN-8 Barcode
<u>SetGS1</u>	Sets the option of GS1 Barcode
<u>SetKOREAPOST</u>	Sets the option of KOREAPOST Barcode
<u>SetMSI</u>	Sets the option of MSI Barcode
<u>SetOption</u>	Sets the option of Scanner
<u>SetSymbology</u>	Sets the Enable/Disable of Symbologies
<u>SetSymbologyAll</u>	Enables all of Symbologies
<u>SetSymbologyDefault</u>	Initializes all option of scanner
<u>SetTELEPEN</u>	Sets the option of TELEPEN Barcode
<u>SetUPCA</u>	Sets the option of UPC-A Barcode
<u>SetUPCE</u>	Sets the option of UPC-E Barcode
<u>UnRegHotKey</u>	Free Scanner Button as a hot key

4.1.1 CLOSE

Description

Closes an open scanner.

Syntax

```
public bool Close();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Open

For C++

Library : Scanner.lib

Function : BOOL SCAN_Close()

Example

```
for (int i = 0; i < 3; i++)  
{  
    m_bResult = m_Scanner.Close();  
    Thread.Sleep(300);  
    if (m_bResult == true)  
        break;  
}
```

4.1.2 GetCODABAR

Description

Gets the option of CODABAR Barcode.

Syntax

```
public bool GetCODABAR(out CODABAR_PARAMS pCodabar);
```

Parameters

pCodabar

Pointer to a CODABAR_PARAMS structure to be filled in with the CODABAR common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODABAR

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetCODABAR(PCODABAR_PARAMS pCodabar)

Example

```
private CODABAR_PARAMS m_Codabar = new CODABAR_PARAMS();
m_Scanner.GetCODABAR(out m_Codabar);
CB_ENABLE.Checked = m_Codabar.bEnable;
CB_XMITSS.Checked = m_Codabar.bXSS;
TB_MINLEN.Text = m_Codabar.nMinLen.ToString();
TB_MAXLEN.Text = m_Codabar.nMaxLen.ToString();
```

4.1.3 GetCODE11

Description

Gets the option of CODE11 Barcode.

Syntax

```
public bool GetCODE11(out CODE11_PARAMS pCode11);
```

Parameters

pCode11

Pointer to a CODE11_PARAMS structure to be filled in with the CODE11 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE11

For .C++

Library : Scanner.lib

Function : BOOL GetCODE11(PCODE11_PARAMS pCode11)

Example

```

private CODE11_PARAMS m_Code11 = new CODE11_PARAMS();
m_Scanner.GetCODE11(out m_Code11);
CB_ENABLE.Checked = m_Code11.bEnable;
CB_XCD.Checked = m_Code11.bXCD;
if (m_Code11.nCDV == 1)
    RD_CDV1.Checked = true;
else if (m_Code11.nCDV == 2)
    RD_CDV2.Checked = true;
else
    RD_NOTCONVERT.Checked = true;
TB_MINLEN.Text = m_Code11.nMinLen.ToString();
TB_MAXLEN.Text = m_Code11.nMaxLen.ToString();

```

4.1.4 GetCODE128

Description

Gets the option of CODE128 Barcode.

Syntax

```
public bool GetCODE128(out CODE128_PARAMS pCode128);
```

Parameters

pCode128

Pointer to a CODE128_PARAMS structure to be filled in with the CODE128 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE128

For C++

Library : Scanner.dll

Function : BOOL SCAN_GetCODE128(PCODE128_PARAMS pCode128)

Example

```

private CODE128_PARAMS m_Code128 = new CODE128_PARAMS();
m_Scanner.GetCODE128(out m_Code128);
CB_ENABLE.Checked = m_Code128.bEnable;

```

```
CB_UCCEAN128.Checked = m_Code128.bUCCEAN128;  
TB_ASCII.Text = m_Code128.strFNC1_ASCII;  
TB_MINLEN.Text = m_Code128.nMinLen.ToString();  
TB_MAXLEN.Text = m_Code128.nMaxLen.ToString();
```

4.1.5 GetCODE25

Description

Gets the option of CODE25 Barcode.

Syntax

```
public bool GetCODE25(out CODE25_PARAMS pCode25);
```

Parameters

pCode25

Pointer to a CODE25_PARAMS structure to be filled in with the CODE25 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE25

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetCODE25(PCODE25_PARAMS pCode25)

Example

```
private CODE25_PARAMS m_Code25 = new CODE25_PARAMS();  
m_Scanner.GetCODE25(out m_Code25);  
CB_ENABLE.Checked = m_Code25.bI2OF5;  
CB_STANDARD_2OF5.Checked = m_Code25.bS2OF5;  
CB_ITF14.Checked = m_Code25.bITF14;  
CB_MATRIX_2OF5.Checked = m_Code25.bMATRIX2OF5;  
CB_CHINAPOST.Checked = m_Code25.bCHINAPOST;  
CB_CODE25_INDUSTRY.Checked = m_Code25.bINDUSTRY;  
CB_CODE25_IATA.Checked = m_Code25.bIATA;  
CB_CDV.Checked = m_Code25.bCDV;  
CB_XCD.Checked = m_Code25.bXCD;
```

```
TB_MINLEN.Text = m_Code25.nMinLen.ToString();  
TB_MAXLEN.Text = m_Code25.nMaxLen.ToString();
```

4.1.6 GetCODE39

Description

Gets the option of CODE39 Barcode.

Syntax

```
public bool GetCODE39(out CODE39_PARAMS pCode39);
```

Parameters

pCode39

Pointer to a CODE39_PARAMS structure to be filled in with the CODE39 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE39

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetCODE39(PCODE39_PARAMS pCode39)

Example

```
private CODE39_PARAMS m_Code39 = new CODE39_PARAMS();  
m_Scanner.GetCODE39(out m_Code39);  
CB_ENABLE.Checked = m_Code39.bEnable;  
CB_CDV.Checked = m_Code39.bCDV;  
CB_XCD.Checked = m_Code39.bXCD;  
switch(m_Code39.nFormat)  
{  
    case 0:  
        RD_STANDARD.Checked = true;  
        break;  
    case 1:  
        RD_CODE32.Checked = true;  
        break;
```

```

case 2:
    RD_PZN.Checked = true;
    break;
case 3:
    RD_TRIOPTIC.Checked = true;
    break;
case 4:
    RD_FULLASCII.Checked = true;
    break;
}
TB_MINLEN.Text = m_Code39.nMinLen.ToString();
TB_MAXLEN.Text = m_Code39.nMaxLen.ToString();

```

4.1.7 GetCODE93

Description

Gets the option of CODE93 Barcode.

Syntax

```
public bool GetCODE93(out CODE93_PARAMS pCode93);
```

Parameters

pCode93

Pointer to a CODE93_PARAMS structure to be filled in with the CODE93 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE39

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetCODE93(PCODE93_PARAMS pCode93)

Example

```

private CODE93_PARAMS m_Code93 = new CODE93_PARAMS();
m_Scanner.GetCODE93(out m_Code93);
CB_ENABLE.Checked = m_Code93.bEnable;

```



```
TB_MINLEN.Text = m_Code93.nMinLen.ToString();  
TB_MAXLEN.Text = m_Code93.nMaxLen.ToString();
```

4.1.8 GetDeviceType

Description

Gets the type of device.

Syntax

```
public SCAN_DEVICE_TYPE GetDeviceType();
```

Parameters

None

Return Value

The return value is SCAN_DEVICE_TYPE.

Remarks

None

See Also

GetEngineType

For C++

Library : Scanner.lib

Function : SCAN_DEVICE_TYPE SCAN_GetDeviceType()

Example

None

4.1.9 GetEAN13

Description

Gets the option of EAN-13 Barcode.

Syntax

```
public bool GetEAN13(out EAN13_PARAMS pEan13);
```

Parameters

pEan13

Pointer to a EAN13_PARAMS structure to be filled in with the EAN-13 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetEAN13

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetEAN13(PEAN13_PARAMS pEan13)

Example

```
private EAN13_PARAMS m_Ean13 = new EAN13_PARAMS();  
m_Scanner.GetEAN13(out m_Ean13);  
CB_ENABLE.Checked = m_Ean13.bEnable;  
CB_BOOKLAND.Checked = m_Ean13.bBOOKLAND;  
CB_XCD.Checked = m_Ean13.bXCD;  
CB_SUPP.Checked = m_Ean13.bAddOn;
```

4.1.10 GetEAN8

Description

Gets the option of EAN-8 Barcode.

Syntax

```
public bool GetEAN8(out EAN8_PARAMS pEan8);
```

Parameters

pEan8

Pointer to a EAN8_PARAMS structure to be filled in with the EAN-8 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetEAN8

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetEAN8(PEAN8_PARAMS pEan8)

Example

```
private EAN8_PARAMS m_Ean8 = new EAN8_PARAMS();  
m_Scanner.GetEAN8(out m_Ean8);  
CB_ENABLE.Checked = m_Ean8.bEnable;
```

```
CB_XCD.Checked = m_Ean8.bXCD;
```

```
CB_CONVERT.Checked = m_Ean8.bEAN8_AS_EAN13;
```

4.1.11 GetEngineType

Description

Gets the type of scanner engine.

Syntax

```
public SCAN_ENGINE_TYPE GetEngineType();
```

Parameters

None

Return Value

The return value is SCAN_ENGINE_TYPE.

Remarks

None

See Also

GetDeviceType

For C++

Library : Scanner.lib

Function : SCAN_ENGINE_TYPE SCAN_GetEngineType()

Example

None

4.1.12 GetGS1

Description

Gets the option of GS1 Barcode.

Syntax

```
public bool GetGS1(out GS1_PARAMS pGs1);
```

Parameters

pGs1

Pointer to a GS1_PARAMS structure to be filled in with the GS1 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetGS1

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetGS1PGS1_PARAMS pGs1)

Example

```
private GS1_PARAMS m_Gs1 = new GS1_PARAMS();  
m_Scanner.GetGS1(out m_Gs1);  
CB_ENABLE.Checked = m_Gs1.bGS1;  
CB_GS1LIM.Checked = m_Gs1.bGS1LIM;  
CB_GS1EXP.Checked = m_Gs1.bGS1EXP;
```

4.1.13 GetKOREAPOST

Description

Gets the option of KOREAPOST Barcode.

Syntax

```
public bool GetKOREAPOST(out KOREAPOST_PARAMS pKoreaPost);
```

Parameters

pKoreaPost

Pointer to a KOREAPOST_PARAMS structure to be filled in with the KOREAPOST common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetKOREAPOST

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetKOREAPOST(PKOREAPOST_PARAMS pKoreaPost)

Example

```
private KOREAPOST_PARAMS m_KoreaPost = new KOREAPOST_PARAMS();  
m_Scanner.GetKOREAPOST(out m_KoreaPost);  
CB_ENABLE.Checked = m_KoreaPost.bEnable;
```

4.1.14 GetMSI

Description

Gets the option of MSI Barcode.

Syntax

```
public bool GetMSI(out MSI_PARAMS pMsi);
```

Parameters

pMsi

Pointer to a MSI_PARAMS structure to be filled in with the MSI common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetMSI

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetMSI(PMSI_PARAMS pMsi)

Example

```
private MSI_PARAMS m_Msi = new MSI_PARAMS();
m_Scanner.GetMSI(out m_Msi);
CB_ENABLE.Checked = m_Msi.bEnable;
CB_XCD.Checked = m_Msi.bXCD;
if (m_Msi.nCDV == 1)
    RD_MOD1010.Checked = true;
else if (m_Msi.nCDV == 2)
    RD_MOD1011.Checked = true;
else
    RD_MOD10.Checked = true;
TB_MINLEN.Text = m_Msi.nMinLen.ToString();
TB_MAXLEN.Text = m_Msi.nMaxLen.ToString();
```

4.1.15 GetOption

Description

Gets the option of Scanner.

Syntax

```
public bool GetOption(out DECODER_PARAMS pOption);
```

Parameters

pOption

Pointer to a DECODER_PARAMS structure to be filled in with the scanner parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetOption

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetOption(PDECODER_PARAMS pOption)

Example

```
private DECODER_PARAMS m_DecoderParams = new DECODER_PARAMS();
m_Scanner.GetOption(out m_DecoderParams);
CB_TIMEOUT.SelectedIndex = m_DecoderParams.nTimeOut - 1;
CB_SECURITYLEVEL.SelectedIndex = m_DecoderParams.nSecurityLevel - 1;
CB_VIBRATE.Checked = m_DecoderParams.bVibrate;
if (m_DecoderParams.nSound == 1)
    RD_BEEP.Checked = true;
else if (m_DecoderParams.nSound == 2)
    RD_NONE.Checked = true;
else
    RD_DEFAULT.Checked = true;
CB_AIMID.Checked = m_DecoderParams.bXmitAimID;
CB_CONTINUEMODE.Checked = m_DecoderParams.bContinueMode;
CB_WIDESCAN.Checked = m_DecoderParams.bWindeScan;
CB_HIGHFILTER.Checked = m_DecoderParams.bHighFilter;
```

4.1.16 GetScanData

Description

Gets the ScanData which is read by scanner.

Syntax

```
public bool GetScanData(StringBuilder szBarType, StringBuilder szBarData);
```

Parameters

szBarType

Pointer to barcode type.

szBarData

Pointer to barcode data

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetScanData(TCHAR *pszBarType, TCHAR *pszBarData)

Example

None

4.1.17 GetSymbology

Description

Gets Enable/Disable of Symbologies.

Syntax

```
public bool GetSymbology(out DECODER pSymbology);
```

Parameters

pSymbology

Pointer to a DECODER structure to be filled in with the symbologies enable or disable.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetSymbology

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetSymbology(PDECODER pSymbology)

Example

```
private DECODER m_DECODER = new DECODER();
m_Scanner.GetSymbology(out m_DECODER);
CB_UPCA.Checked = m_DECODER.bUPCA;
CB_UPCE.Checked = m_DECODER.bUPCE;
CB_EAN13.Checked = m_DECODER.bEAN13;
CB_EAN8.Checked = m_DECODER.bEAN8;
CB_CODE39.Checked = m_DECODER.bCODE39;
CB_CODE128.Checked = m_DECODER.bCODE128;
CB_CODE93.Checked = m_DECODER.bCODE93;
CB_CODE11.Checked = m_DECODER.bCODE11;
CB_CODE25.Checked = m_DECODER.bCODE25;
CB_CODABAR.Checked = m_DECODER.bCODABAR;
CB_KOREAPOST.Checked = m_DECODER.bKOREAPOST;
CB_MSI.Checked = m_DECODER.bMSI;
CB_GS1.Checked = m_DECODER.bGS1;
CB_TELEPEN.Checked = m_DECODER.bTELEPEN;
```

4.1.18 GetTELEPEN

Description

Gets the option of TELEPEN Barcode.

Syntax

```
public bool GetTELEPEN(out TELEPEN_PARAMS pTelepen);
```

Parameters

pTelepen

Pointer to a TELEPEN_PARAMS structure to be filled in with the TELEPEN common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetTELEPEN

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetTELEPEN(PTELEPEN_PARAMS pTelepen)

Example

```
private TELEPEN_PARAMS m_Telepen = new TELEPEN_PARAMS();  
m_Scanner.GetTELEPEN(out m_Telepen);  
CB_ENABLE.Checked = m_Telepen.bEnable;  
CB_NUMERIC.Checked = m_Telepen.bNumeric;
```

4.1.19 GetUPCA

Description

Gets the option of UPC-A Barcode.

Syntax

```
public bool GetUPCA(out UPCA_PARAMS pUpca);
```

Parameters

pUpca

Pointer to a UPCA_PARAMS structure to be filled in with the UPC-A common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetUPCA

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetUPCA(PUPCA_PARAMS pUpca)

Example

```
private UPCA_PARAMS m_Upca = new UPCA_PARAMS();  
m_Scanner.GetUPCA(out m_Upca);  
CB_ENABLE.Checked = m_Upca.bEnable;  
CB_XMIT_NUM.Checked = m_Upca.bXNum;  
CB_XCD.Checked = m_Upca.bXCD;  
CB_CONVERT.Checked = m_Upca.bUPCA_AS_EAN13;  
CB_SUPP.Checked = m_Upca.bAddOn;
```

4.1.20 GetUPCE

Description

Gets the option of UPC-E Barcode.

Syntax

```
public bool GetUPCE(out UPCE_PARAMS pUpce);
```

Parameters

pUpce

Pointer to a UPCE_PARAMS structure to be filled in with the UPC-E common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetUPCE

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetUPCE(PUPCE_PARAMS pUpce)

Example

```
private UPCE_PARAMS m_Upce = new UPCE_PARAMS();
m_Scanner.GetUPCE(out m_Upce);
CB_ENABLE.Checked = m_Upce.bEnable;
CB_XMIT_NUM.Checked = m_Upce.bXNum;
CB_XCD.Checked = m_Upce.bXCD;
if (m_Upce.nConvert == 1)
    RD_UPCEASUPCA.Checked = true;
else if (m_Upce.nConvert == 2)
    RD_UPCEASEAN13.Checked = true;
else
    RD_NOTCONVERT.Checked = true;
```

4.1.21 GetVersionInfo

Description

Gets the information of scanner engine and dll version.

Syntax

```
public string GetVersionInfo();
```

Parameters

pszVersion

Pointer to a TCHAR to be filled in with the version info.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : Scanner.lib

Function : BOOL SCAN_GetVersionInfo(TCHAR* pszVersion)

Example

None

4.1.22 Open

Description

Opens a scanner.

Syntax

```
public bool Open();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Close

For C++

Library : Scanner.lib

Function : BOOL SCAN_Open()

Example

None

4.1.23 Read

Description

Starts the beaming of scanner.

Syntax

```
public bool Read();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

ReadCancel

For C++

Library : Scanner.lib

Function : BOOL SCAN_Read()

Example

None

4.1.24 ReadCancel

Description

Stops the beaming of scanner.

Syntax

```
public bool ReadCancel();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Read

For C++

Library : Scanner.lib

Function : BOOL SCAN_ReadCancel()

Example

None

4.1.25 RegHotKey

Description

Registers Scanner Button as a hot key.

Syntax

```
public bool RegHotKey(int id, uint vk, bool SyncMode);
```

Parameters

id

Identifier of the hot key. No other hot key in the calling thread should have the same identifier. An application must specify a value in the range 0x0000 through 0xBFFF.

vk

The value of Virtual Key.

SyncMode

The state is either true = Sync Mode, false = ASync Mode.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

UnRegHotKey

For C++

None

Example

```
public const int m_nHotKeyCE = 133;    // VK_F22(WinCE)
public const int m_nHotKeyWM = 125;    // VK_F14(WM)
// Get Device Type
m_DeviceType = m_Scanner.GetDeviceType();
// OS Type
```

```

if ((m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3SMART_CE) || (m_DeviceType ==
SCAN_DEVICE_TYPE.DEVICE_M3GREEN) || (m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3T) ||
(m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3POS))

    m_bWinCE = true;

//Scanner Open
m_Scanner.Open();
if (m_bWinCE == true)

    m_Scanner.RegHotKey(1, m_nHotKeyCE, m_bSyncMode);
else

    m_Scanner.RegHotKey(1, m_nHotKeyWM, m_bSyncMode);

```

4.1.26 SetCODABAR

Description

Sets the option of CODABAR Barcode.

Syntax

```
public bool SetCODABAR(ref CODABAR_PARAMS pCodabar);
```

Parameters

pCodabar

Pointer to a CODABAR_PARAMS structure holding the CODABAR common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODABAR

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetCODABAR(PCODABAR_PARAMS pCodabar)

Example

```

private CODABAR_PARAMS m_Codabar = new CODABAR_PARAMS();
m_Codabar.bEnable = CB_ENABLE.Checked;
m_Codabar.bXSS = CB_XMITSS.Checked;
m_Codabar.nMinLen = Int32.Parse(TB_MINLEN.Text);
m_Codabar.nMaxLen = Int32.Parse(TB_MAXLEN.Text);
m_Scanner.SetCODABAR(ref m_Codabar);

```

4.1.27 SetCODE11

Description

Sets the option of CODE11 Barcode.

Syntax

```
public bool SetCODE11(ref CODE11_PARAMS pCode11);
```

Parameters

pCode11

Pointer to a CODE11_PARAMS structure holding the CODE11 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE11

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetCODE11(PCODE11_PARAMS pCode11)

Example

```
private CODE11_PARAMS m_Code11 = new CODE11_PARAMS();
m_Code11.bEnable = CB_ENABLE.Checked;
m_Code11.bXCD = CB_XCD.Checked;
if (RD_CDV1.Checked)
    m_Code11.nCDV = 1;
else if (RD_CDV2.Checked)
    m_Code11.nCDV = 2;
else
    m_Code11.nCDV = 0;
m_Code11.nMinLen = Int32.Parse(TB_MINLEN.Text);
m_Code11.nMaxLen = Int32.Parse(TB_MAXLEN.Text);
m_Scanner.SetCODE11(ref m_Code11);
```

4.1.28 SetCODE128

Description

Sets the option of CODE128 Barcode.

Syntax

```
public bool SetCODE128(ref CODE128_PARAMS pCode128);
```

Parameters

pCode128

Pointer to a CODE128_PARAMS structure holding the CODE128 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE128

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetCODE128(PCODE128_PARAMS pCode128)

Example

```
private CODE128_PARAMS m_Code128 = new CODE128_PARAMS();  
m_Code128.bEnable = CB_ENABLE.Checked;  
m_Code128.bUCCEAN128 = CB_UCCEAN128.Checked;  
m_Code128.strFNC1_ASCII = TB_ASCII.Text;  
m_Code128.nMinLen = Int32.Parse(TB_MINLEN.Text);  
m_Code128.nMaxLen = Int32.Parse(TB_MAXLEN.Text);  
m_Scanner.SetCODE128(ref m_Code128);
```

4.1.29 SetCODE25

Description

Sets the option of CODE25 Barcode.

Syntax

```
public bool SetCODE25(ref CODE25_PARAMS pCode25);
```

Parameters

pCode25

Pointer to a CODE25_PARAMS structure holding the CODE25 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE25

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetCODE25(PCODE25_PARAMS pCode25)

Example

```
private CODE25_PARAMS m_Code25 = new CODE25_PARAMS();
m_Code25.bI2OF5 = CB_ENABLE.Checked;
m_Code25.bS2OF5 = CB_STANDARD_2OF5.Checked;
m_Code25.bITF14 = CB_ITF14.Checked;
m_Code25.bMATRIX2OF5 = CB_MATRIX_2OF5.Checked;
m_Code25.bCHINAPOST = CB_CHINAPOST.Checked;
m_Code25.bINDUSTRY = CB_CODE25_INDUSTRY.Checked;
m_Code25.bIATA = CB_CODE25_IATA.Checked;
m_Code25.bCDV = CB_CDV.Checked;
m_Code25.bXCD = CB_XCD.Checked;
m_Code25.nMinLen = Int32.Parse(TB_MINLEN.Text);
m_Code25.nMaxLen = Int32.Parse(TB_MAXLEN.Text);
m_Scanner.SetCODE25(ref m_Code25);
```

4.1.30 SetCODE39

Description

Sets the option of CODE39 Barcode.

Syntax

```
public bool SetCODE39(ref CODE39_PARAMS pCode39);
```

Parameters

pCode39

Pointer to a CODE39_PARAMS structure holding the CODE39 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE39

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetCODE39(PCODE39_PARAMS pCode39)

Example

```
private CODE39_PARAMS m_Code39 = new CODE39_PARAMS();
m_Code39.bEnable = CB_ENABLE.Checked;
m_Code39.bCDV = CB_CDV.Checked;
m_Code39.bXCD = CB_XCD.Checked;
if (RD_CODE32.Checked == true)
    m_Code39.nFormat = 1;
else if (RD_PZN.Checked == true)
    m_Code39.nFormat = 2;
else if (RD_TRIOPTIC.Checked == true)
    m_Code39.nFormat = 3;
else if (RD_FULLASCII.Checked == true)
    m_Code39.nFormat = 4;
else
    m_Code39.nFormat = 0;
m_Code39.nMinLen = Int32.Parse(TB_MINLEN.Text);
m_Code39.nMaxLen = Int32.Parse(TB_MAXLEN.Text);
m_Scanner.SetCODE39(ref m_Code39);
```

4.1.31 SetCODE93

Description

Sets the option of CODE93 Barcode.

Syntax

```
public bool SetCODE93(ref CODE93_PARAMS pCode93);
```

Parameters

pCode93

Pointer to a CODE93_PARAMS structure holding the CODE93 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE93

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetCODE93(PCODE93_PARAMS pCode93)

Example

```
private CODE93_PARAMS m_Code93 = new CODE93_PARAMS();  
m_Code93.bEnable = CB_ENABLE.Checked;  
m_Code93.nMinLen = Int32.Parse(TB_MINLEN.Text);  
m_Code93.nMaxLen = Int32.Parse(TB_MAXLEN.Text);  
m_Scanner.SetCODE93(ref m_Code93);
```

4.1.32 SetEAN13

Description

Sets the option of EAN-13 Barcode.

Syntax

```
public bool SetEAN13(ref EAN13_PARAMS pEan13);
```

Parameters

pEan13

Pointer to a EAN13_PARAMS structure holding the EAN-13 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetEAN13

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetEAN13(EAN13_PARAMS pEan13)

Example

```
private EAN13_PARAMS m_Ean13 = new EAN13_PARAMS();  
m_Ean13.bEnable = CB_ENABLE.Checked;
```

```
m_Ean13.bBOOKLAND = CB_BOOKLAND.Checked;  
m_Ean13.bXCD = CB_XCD.Checked;  
m_Ean13.bAddOn = CB_SUPP.Checked;  
m_Scanner.SetEAN13(ref m_Ean13);
```

4.1.33 SetEAN8

Description

Sets the option of EAN-8 Barcode.

Syntax

```
public bool SetEAN8(ref EAN8_PARAMS pEan8);
```

Parameters

pEan8

Pointer to a EAN8_PARAMS structure holding the EAN9 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetEAN8

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetEAN8(EAN8_PARAMS pEan8)

Example

```
private EAN8_PARAMS m_Ean8 = new EAN8_PARAMS();  
m_Ean8.bEnable = CB_ENABLE.Checked;  
m_Ean8.bXCD = CB_XCD.Checked;  
m_Ean8.bEAN8_AS_EAN13 = CB_CONVERT.Checked;  
m_Scanner.SetEAN8(ref m_Ean8);
```

4.1.34 SetGS1

Description

Sets the option of GS1 Barcode.

Syntax

```
public bool SetGS1(ref GS1_PARAMS pGs1);
```

Parameters

pGs1

Pointer to a GS1_PARAMS structure holding the GS1 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetGS1

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetGS1PGS1_PARAMS pGs1)

Example

```
private GS1_PARAMS m_Gs1 = new GS1_PARAMS();  
m_Gs1.bGS1 = CB_ENABLE.Checked;  
m_Gs1.bGS1LIM = CB_GS1LIM.Checked;  
m_Gs1.bGS1EXP = CB_GS1EXP.Checked;  
m_Scanner.SetGS1(ref m_Gs1);
```

4.1.35 SetKOREAPOST

Description

Sets the option of KOREAPOST Barcode.

Syntax

```
public bool SetKOREAPOST(ref KOREAPOST_PARAMS pKoreaPost);
```

Parameters

pKoreaPost

Pointer to a KOREAPOST_PARAMS structure holding the KOREAPOST common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetKOREAPOST

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetKOREAPOST(PKOREAPOST_PARAMS pKoreaPost)

Example

```
private KOREAPOST_PARAMS m_KoreaPost = new KOREAPOST_PARAMS();  
m_KoreaPost.bEnable = CB_ENABLE.Checked;  
m_Scanner.SetKOREAPOST(ref m_KoreaPost);
```

4.1.36 SetMSI

Description

Sets the option of MSI Barcode.

Syntax

```
public bool SetMSI(ref MSI_PARAMS pMsi);
```

Parameters

pMsi

Pointer to a MSI_PARAMS structure holding the MSI common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetMSI

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetMSI(PMSI_PARAMS pMsi)

Example

```
private MSI_PARAMS m_Msi = new MSI_PARAMS();  
m_Msi.bEnable = CB_ENABLE.Checked;  
m_Msi.bXCD = CB_XCD.Checked;  
if (RD_MOD1010.Checked)  
    m_Msi.nCDV = 1;  
else if (RD_MOD1011.Checked)  
    m_Msi.nCDV = 2;  
else  
    m_Msi.nCDV = 0;
```

```
m_Msi.nMinLen = Int32.Parse(TB_MINLEN.Text);  
m_Msi.nMaxLen = Int32.Parse(TB_MAXLEN.Text);  
m_Scanner.SetMSI(ref m_Msi);
```

4.1.37 SetOption

Description

Sets the option of Scanner.

Syntax

```
public bool SetOption(ref DECODER_PARAMS pOption);
```

Parameters

pOption

Pointer to a DECODER_PARAMS structure holding scanner parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetOption

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetOption(PDECODER_PARAMS pOption)

Example

```
private DECODER_PARAMS m_DecoderParams = new DECODER_PARAMS();  
m_DecoderParams.nTimeOut = CB_TIMEOUT.SelectedIndex + 1;  
m_DecoderParams.nSecurityLevel = CB_SECURITYLEVEL.SelectedIndex + 1;  
if (RD_BEEP.Checked == true)  
    m_DecoderParams.nSound = 1;  
else if (RD_NONE.Checked == true)  
    m_DecoderParams.nSound = 2;  
else  
    m_DecoderParams.nSound = 0;  
m_DecoderParams.bVibrate = CB_VIBRATE.Checked;  
m_DecoderParams.bXmitAimID = CB_AIMID.Checked;  
m_DecoderParams.bContinueMode = CB_CONTINUEMODE.Checked;
```

```
m_DecoderParams.bWindeScan = CB_WIDESCAN.Checked;  
m_DecoderParams.bHighFilter = CB_HIGHFILTER.Checked;  
m_Scanner.SetOption(ref m_DecoderParams);
```

4.1.38 SetSymbology

Description

Sets Enable/Disable of Symbologies.

Syntax

```
public bool SetSymbology(ref DECODER pSymbology);
```

Parameters

pSymbology

Pointer to a DECODER structure holding the symbologies enable or disable.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetSymbology

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetSymbology(PDECODER pSymbology)

Example

```
private DECODER m_DECODER = new DECODER();  
m_DECODER.bUPCA = CB_UPCA.Checked;  
m_DECODER.bUPCE = CB_UPCE.Checked;  
m_DECODER.bEAN13 = CB_EAN13.Checked;  
m_DECODER.bEAN8 = CB_EAN8.Checked;  
m_DECODER.bCODE39 = CB_CODE39.Checked;  
m_DECODER.bCODE128 = CB_CODE128.Checked;  
m_DECODER.bCODE93 = CB_CODE93.Checked;  
m_DECODER.bCODE11 = CB_CODE11.Checked;  
m_DECODER.bCODE25 = CB_CODE25.Checked;  
m_DECODER.bCODABAR = CB_CODABAR.Checked;  
m_DECODER.bKOREAPOST = CB_KOREAPOST.Checked;
```



```
m_DECODER.bMSI = CB_MSI.Checked;  
m_DECODER.bGS1 = CB_GS1.Checked;  
m_DECODER.bTELEPEN = CB_TELEPEN.Checked;  
m_Scanner.SetSymbology(ref m_DECODER);
```

4.1.39 SetSymbologyAll

Description

Enables all of Symbologies.

Syntax

```
public bool SetSymbologyAll();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetSymbologyDefault

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetSymbologyAll()

Example

None

4.1.40 SetSymbologyDefault

Description

Initializes all option of scanner.

Syntax

```
public bool SetSymbologyDefault();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetSymbologyAll

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetSymbologyDefault()

Example

None

4.1.41 SetTELEPEN

Description

Sets the option of TELEPEN Barcode.

Syntax

```
public bool SetTELEPEN(ref TELEPEN_PARAMS pTelepen);
```

Parameters

pTelepen

Pointer to a TELEPEN_PARAMS structure holding the TELEPEN common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetTELEPEN

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetTELEPEN(PTELEPEN_PARAMS pTelepen)

Example

```
private TELEPEN_PARAMS m_Telepen = new TELEPEN_PARAMS();  
m_Telepen.bEnable = CB_ENABLE.Checked;  
m_Telepen.bNumeric = CB_NUMERIC.Checked;  
m_Scanner.SetTELEPEN(ref m_Telepen);
```

4.1.42 SetUPCA

Description

Sets the option of UPC-A Barcode.

Syntax

```
public bool SetUPCA(ref UPCA_PARAMS pUpca);
```

Parameters

pUpca

Pointer to a UPCA_PARAMS structure holding the UPC-A common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetUPCA

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetUPCA(PUPCA_PARAMS pUpca)

Example

```
private UPCA_PARAMS m_Upca = new UPCA_PARAMS();  
m_Upca.bEnable = CB_ENABLE.Checked;  
m_Upca.bXNum = CB_XMIT_NUM.Checked;  
m_Upca.bXCD = CB_XCD.Checked;  
m_Upca.bUPCA_AS_EAN13 = CB_CONVERT.Checked;  
m_Upca.bAddOn = CB_SUPP.Checked;  
m_Scanner.SetUPCA(ref m_Upca);
```

4.1.43 SetUPCE

Description

Sets the option of UPC-E Barcode.

Syntax

```
public bool SetUPCE(ref UPCE_PARAMS pUpce);
```

Parameters

pUpce

Pointer to a UPCE_PARAMS structure holding the UPC-E common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetUPCE

For C++

Library : Scanner.lib

Function : BOOL SCAN_SetUPCE(PUPCE_PARAMS pUpce)

Example

```
private UPCE_PARAMS m_Upce = new UPCE_PARAMS();
m_Upce.bEnable = CB_ENABLE.Checked;
m_Upce.bXNum = CB_XMIT_NUM.Checked;
m_Upce.Bxcd = CB_XCD.Checked;
if (RD_UPCEASUPCA.Checked)
    m_Upce.nConvert = 1;
else if (RD_UPCEASEAN13.Checked)
    m_Upce.nConvert = 2;
else
    m_Upce.nConvert = 0;
m_Scanner.SetUPCE(ref m_Upce);
```

4.1.44 UnRegHotKey

Description

Free Scanner Button as a hot key.

Syntax

```
public bool UnRegHotKey(int id);
```

Parameters

id

Identifier of the hot key to be freed.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

RegHotKey

For C++

None

Example

None

4.2 IMAGER (2D)

Status Return value

Please refer to the below table for the status value definition.

Status Value Definition	Code	Meaning
TRUE	1	Success
FALSE	0	General Error

Event
public event ImagerDataDelegate ImagerDataEvent;
Enum
<pre>public enum SCAN_DEVICE_TYPE { DEVICE_M3SKY = 0, DEVICE_M3SKYSAM, DEVICE_MM3, DEVICE_M3ORANGE, DEVICE_M3SMART_CE, DEVICE_M3SMART_WM, DEVICE_M3GREEN, DEVICE_M3T, DEVICE_M3POS, DEVICE_M3ORANGEPLUS, DEVICE_M3ORANGEPLUS_CE }; public enum SCAN_SOUND { SOUND_DEFAULT = 0, SOUND_BEEP, SOUND_NONE }; public enum CODE39_FORMAT { CODE39_STANDARD = 0, CODE39_CODE32, CODE39_TRIOPTIC }; public enum OCR_MODE { MODE_OCR_DISABLED = 0, MODE_OCR_A,</pre>

```

MODE_OCR_B,
MODE_OCR_MONEY,
MODE_OCR_MICR_UNSUPPORTED
};

public enum DECODEMODE
{
    DECODE_STANDARD = 0,
    DECODE_QUICK_OMNI,
    DECODE_LINEAR_PRIORITY
};

public enum CAM_RESOLUTION
{
    CAM_640X480 = 0,
    CAM_320X240,
    CAM_160X120
};

public enum CAM_FORMAT
{
    CAM_JPG = 0,
    CAM_BMP
};

public enum SAVE_MODE
{
    SAVE_DATE = 0,
    SAVE_CUSTOM_WITH_NUM,
    SAVE_CUSTOM
};

public enum IQ_TYPE
{
    IQ_AZTEC = 0,
    IQ_CODE39
};

```

Structure

```

public struct DECODER
{
    public bool bAZTEC;
    public bool bCODABAR;
    public bool bCODE11;
    public bool bCODE128;
    public bool bCODE39;
    public bool bCODE49;
    public bool bCODE93;
    public bool bCOMPOSITE;
    public bool bDATAMATRIX;
    public bool bEAN8;

```

```
public bool bEAN13;
public bool bINT25;
public bool bMAXICODE;
public bool bMICROPDF;
public bool bOCR;
public bool bPDF417;
public bool bPOSTNET;
public bool bQR;
public bool bRSS;
public bool bUPCA;
public bool bUPCE;
public bool bISBT;
public bool bBPO;
public bool bCANPOST;
public bool bAUSPOST;
public bool bIATA25;
public bool bCODABLOCK;
public bool bJAPOST;
public bool bPLANET;
public bool bDUTCHPOST;
public bool bMSI;
public bool bTLCODE39;
public bool bSTR25;
public bool bMATRIX25;
public bool bPLESSEY;
public bool bCHINAPOST;
public bool bKOREAPOST;
public bool bTELEPEN;
public bool bCODE16K;
public bool bPOSICODE;
public bool bCOUPONCODE;
public bool bUSPS4CB;
public bool bIDTAG;
public bool bLABEL;
public bool bGS1_128;
public bool bHANXIN;
public bool bGRIDMATRIX;
```

```
public DECODER(bool bAZTEC, bool bCODABAR, bool bCODE11, bool bCODE128, bool bCODE39, bool bCODE49, bool
bCODE93, bool bCOMPOSITE, bool bDATAMATRIX, bool bEAN8, bool bEAN13, bool bINT25, bool bMAXICODE, bool
bMICROPDF, bool bOCR, bool bPDF417, bool bPOSTNET, bool bQR, bool bRSS, bool bUPCA, bool bUPCE, bool bISBT, bool
bBPO, bool bCANPOST, bool bAUSPOST, bool bIATA25, bool bCODABLOCK, bool bJAPOST, bool bPLANET, bool
bDUTCHPOST, bool bMSI, bool bTLCODE39, bool bSTR25, bool bMATRIX25, bool bPLESSEY, bool bCHINAPOST, bool
bKOREAPOST, bool bTELEPEN, bool bCODE16K, bool bPOSICODE, bool bCOUPONCODE, bool bUSPS4CB, bool bIDTAG, bool
bLABEL, bool bGS1_128, bool bHANXIN, bool bGRIDMATRIX);
```

```
};
```

```
public struct DECODER_PARAMS
```

```
{
```

```
public bool bCentering;
```



```

    public bool bContinueMode;
    public bool bXmitAimID;
    public bool bHexMode;
    public bool bVibrate;
    public int nSound;
    public int nTimeOut;
    public int nLightMode;
    public DECODER_PARAMS(bool bCentering, bool bContinueMode, bool bXmitAimID, bool bHexMode, bool bVibrate, int
nSound, int nTimeOut, int nLightMode);
};
public struct DECOPTION_PARAMS
{
    public int nDecodeMode;
    public int nLinearRange;
    public int nPrintWeight;
    public int nMaxDecode;
    public int nMaxSearch;
    public bool bVideoReverse;
    public DECOPTION_PARAMS(int nDecodeMode, int nLinearRange, int nPrintWeight, int nMaxDecode, int nMaxSearch,
bool bVideoReverse);
};
public struct CAM_PARAMS
{
    public int nSaveMode;
    public int nSaveFormat;
    public int nJpegQuality;
    public int nResolution;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string szSaveFolder;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string szFileName;
    public CAM_PARAMS(int nSaveMode, int nSaveFormat, int nJpegQuality, int nResolution, string szSaveFolder, string
szFileName);
};
public struct IQ_PARAMS
{
    public int nSaveMode;
    public int nIQType;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string szSaveFolder;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string szFileName;
    public IQ_PARAMS(int nSaveMode, int nIQType, string szSaveFolder, string szFileName);
};
public struct AZTEC_PARAMS{
    public bool    bEnable;
    public int    nMinLen;

```

```

    public int  nMaxLen;
    public AZTEC_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct CODABAR_PARAMS{
    public bool  bEnable;
    public bool  bCDV;
    public bool  bXCD;
    public bool  bXSS;
    public int  nMinLen;
    public int  nMaxLen;
    public CODABAR_PARAMS(bool bEnable, bool bCDV, bool  bXCD, bool bXSS, int nMinLen, int nMaxLen);
};

public struct CODE11_PARAMS{
    public bool  bEnable;
    public bool  bCDV;
    public int  nMinLen;
    public int  nMaxLen;
    public CODE11_PARAMS(bool bEnable, bool bCDV, int nMinLen, int nMaxLen);
};

public struct CODE128_PARAMS{
    public bool  bEnable;
    public int  nMinLen;
    public int  nMaxLen;
    public CODE128_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct CODE39_PARAMS{
    public bool  bEnable;
    public int  nFormat;
    public bool  bCDV;
    public bool  bXCD;
    public bool  bFullASCII;
    public int  nMinLen;
    public int  nMaxLen;
    public CODE39_PARAMS(bool bEnable, int nFormat, bool bCDV, bool bXCD, bool bFullASCII, int nMinLen, int nMaxLen);
};

public struct CODE49_PARAMS{
    public bool  bEnable;
    public int  nMinLen;
    public int  nMaxLen;

    public CODE49_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct CODE93_PARAMS{
    public bool  bEnable;
    public int  nMinLen;
    public int  nMaxLen;

```

```

    public CODE93_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct COMPOSITE_PARAMS{
    public bool    bEnable;
    public bool    bComposite_Upc;
    public int     nMinLen;
    public int     nMaxLen;
    public COMPOSITE_PARAMS(bool bEnable, bool bComposite_Upc, int nMinLen, int nMaxLen);
};

public struct DATAMATRIX_PARAMS{
    public bool    bEnable;
    public int     nMinLen;
    public int     nMaxLen;

    public DATAMATRIX_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct EAN8_PARAMS{
    public bool    bEnable;
    public bool    bXCD;
    public bool    bAddOn;
    public EAN8_PARAMS(bool bEnable, bool bXCD, bool bAddOn);
};

public struct EAN13_PARAMS{
    public bool    bEnable;
    public bool    bXCD;
    public bool    bAddOn;

    public EAN13_PARAMS(bool bEnable, bool bXCD, bool bAddOn);
};

public struct INT25_PARAMS{
    public bool    bEnable;
    public bool    bCDV;
    public bool    bXCD;
    public int     nMinLen;
    public int     nMaxLen;

    public INT25_PARAMS(bool bEnable, bool bCDV, bool bXCD, int nMinLen, int nMaxLen);
};

public struct MAXICODE_PARAMS{
    public bool    bEnable;
    public int     nMinLen;
    public int     nMaxLen;
    public MAXICODE_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct MICROPDF_PARAMS{
    public bool    bEnable;

```

```

    public int  nMinLen;
    public int  nMaxLen;

    public MICROPDF_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct OCR_PARAMS{
    public bool      bEnable;
    public int nMode;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string szTemplate;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string szGroupG;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string szGroupH;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 64)]
    public string szCheckChar;
    public OCR_PARAMS(bool bEnable, int nMode, string szTemplate, string szGroupG, string szGroupH, string szCheckChar);
};

public struct PDF417_PARAMS{
    public bool  bEnable;
    public int  nMinLen;
    public int  nMaxLen;

    public PDF417_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct POSTNET_PARAMS{
    public bool  bEnable;
    public bool  bXCD;

    public POSTNET_PARAMS(bool bEnable, bool bXCD);
};

public struct QR_PARAMS{
    public bool  bEnable;
    public int  nMinLen;
    public int  nMaxLen;

    public QR_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct RSS_PARAMS{
    public bool  bEnable;
    public bool  bRssLim;
    public bool  bRssExp;
    public int  nMinLen;
    public int  nMaxLen;
    public RSS_PARAMS(bool bEnable, bool bRssLim, bool bRssExp, int nMinLen, int nMaxLen);
};

```

```

public struct UPCA_PARAMS{
    public bool    bEnable;
    public bool    bXCD;
    public bool    bXNum;
    public bool    bAddOn;

    public UPCA_PARAMS(bool bEnable, bool bXCD, bool bXNum, bool bAddOn);
};

public struct UPCE_PARAMS{
    public bool    bEnable;
    public bool    bXCD;
    public bool    bXNum;
    public bool    bAddOn;

    public UPCE_PARAMS(bool bEnable, bool bXCD, bool bXNum, bool bAddOn);
};

public struct IATA25_PARAMS{
    public bool    bEnable;
    public int     nMinLen;
    public int     nMaxLen;

    public IATA25_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct CODABLOCK_PARAMS{
    public bool    bEnable;
    public int     nMinLen;
    public int     nMaxLen;
    public CODABLOCK_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct PLANET_PARAMS{
    public bool    bEnable;
    public bool    bXCD;

    public PLANET_PARAMS(bool bEnable, bool bXCD);
};

public struct MSI_PARAMS{
    public bool    bEnable;
    public bool    bXCD;
    public int     nMinLen;
    public int     nMaxLen;
    public MSI_PARAMS(bool bEnable, bool bXCD, int nMinLen, int nMaxLen);
};

public struct STRT25_PARAMS{
    public bool    bEnable;
    public int     nMinLen;
    public int     nMaxLen;

```

```
    public STRT25_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct MATRIX25_PARAMS{
    public bool    bEnable;
    public int    nMinLen;
    public int    nMaxLen;

    public MATRIX25_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct PLESSEY_PARAMS{
    public bool    bEnable;
    public int    nMinLen;
    public int    nMaxLen;

    public PLESSEY_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct CHINAPOST_PARAMS{
    public bool    bEnable;
    public int    nMinLen;
    public int    nMaxLen;

    public CHINAPOST_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct KOREAPOST_PARAMS{
    public bool    bEnable;
    public int    nMinLen;
    public int    nMaxLen;

    public KOREAPOST_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};

public struct TELEPEN_PARAMS{
    public bool    bEnable;
    public bool    bNumeric;
    public int    nMinLen;
    public int    nMaxLen;

    public TELEPEN_PARAMS(bool bEnable, bool bNumeric, int nMinLen, int nMaxLen);
};

public struct CODE16K_PARAMS{
    public bool    bEnable;
    public int    nMinLen;
    public int    nMaxLen;

    public CODE16K_PARAMS(bool bEnable, int nMinLen, int nMaxLen);
};
```

```
public struct POSICODE_PARAMS{
    public bool    bEnable;
    public bool    bPosi_Lim1;
    public bool    bPosi_Lim2;
    public int     nMinLen;
    public int     nMaxLen;
    public POSICODE_PARAMS(bool bEnable, bool bPosi_Lim1, bool bPosi_Lim2, int nMinLen, int nMaxLen);
};

public struct RECT_PARAM
{
    public int left;
    public int top;
    public int right;
    public int bottom;
    public RECT_PARAM(int left, int top, int right, int bottom);
};
```

Functions for 2D Scanner

Name	Description
CAMCapture	Captures the preview of imaging
CAMGetOption	Gets the option of imaging
CAMInit	Initializes imaging device
CAMPreviewStart	Starts the preview of imaging
CAMPreviewStop	Stops the preview of imaging
CAMSetOption	Sets the option of imaging
CAMUninit	Uninitializes imaging device
Close	Closes an open imager
GetAZTEC	Gets the option of AZTEC Barcode
GetCenteringWindow	Gets Enable/Disable of centering window function
GetCHINAPOST	Gets the option of CHINAPOST Barcode
GetCODABAR	Gets the option of CODABAR Barcode
GetCODABLOCK	Gets the option of CODABLOCK Barcode
GetCODE11	Gets the option of CODE11 Barcode
GetCODE128	Gets the option of CODE128 Barcode
GetCODE16K	Gets the option of CODE16K Barcode
GetCODE39	Gets the option of CODE39 Barcode
GetCODE49	Gets the option of CODE49 Barcode
GetCODE93	Gets the option of CODE93 Barcode
GetCOMPOSITE	Gets the option of COMPOSITE Barcode
GetDATAMATRIX	Gets the option of DATAMATRIX Barcode
GetDecOption	Gets the option of Decoder
GetDeviceType	Gets the type of device
GetEAN13	Gets the option of EAN-13 Barcode
GetEAN8	Gets the option of EAN-8 Barcode
GetIATA25	Gets the option of IATA25 Barcode
GetImagerMode	Gets the Imager Mode
GetINT25	Gets the option of INT25 Barcode
GetKOREAPOST	Gets the option of KOREAPOST Barcode
GetMATRIX25	Gets the option of MATRIX25 Barcode

GetMAXICODE	Gets the option of MAXICODE Barcode
GetMICROPDF	Gets the option of MICROPDF Barcode
GetMSI	Gets the option of MSI Barcode
GetOCR	Gets the option of OCR Barcode
GetOption	Gets the option of imager
GetPDF417	Gets the option of PDF417 Barcode
GetPLANET	Gets the option of PLANET Barcode
GetPLESSEY	Gets the option of PLESSEY Barcode
GetPOSICODE	Gets the option of POSICODE Barcode
GetPOSTNET	Gets the option of POSTNET Barcode
GetQR	Gets the option of QR Barcode
GetRSS	Gets the option of RSS Barcode
GetScanData	Gets the ScanData which is read by imager
GetSTRT25	Gets the option of STRT25 Barcode
GetSymbology	Gets Enable/Disable of Symbologies
GetTELEPEN	Gets the option of TELEPEN Barcode
GetUPCA	Gets the option of UPC-A Barcode
GetUPCE	Gets the option of UPC-E Barcode
GetVersionInfo	Gets the information of imager driver and dll version
IQGetBarcodeData	Gets the ScanData which is read by imager
IQGetOption	Gets the option of IQ Imaging
IQImagingStart	Starts IQ Imaging
IQImagingStop	Stops IQ Imaging
IQInit	Initializes IQ imaging
IQSetOption	Sets the option of IQ Imaging
IQUnInit	Uninitializes IQ imaging
Open	Opens a imager
Read	Starts the beaming of imager
ReadCancel	Stops the beaming of imager
RegHotKey	Registers Scanner Button as a hot key
SetAZTEC	Sets the option of AZTEC Barcode
SetCenteringWindow	Enables centering window function

SetCHINAPOST	Sets the option of CHINAPOST Barcode
SetCODABAR	Sets the option of CODABAR Barcode
SetCODABLOCK	Sets the option of CODABLOCK Barcode
SetCODE11	Sets the option of CODE11 Barcode
SetCODE128	Sets the option of CODE128 Barcode
SetCODE16K	Sets the option of CODE16K Barcode
SetCODE39	Sets the option of CODE39 Barcode
SetCODE49	Sets the option of CODE49 Barcode
SetCODE93	Sets the option of CODE93 Barcode
SetCOMPOSITE	Sets the option of COMPOSITE Barcode
SetDATAMATRIX	Sets the option of DATAMATRIX Barcode
SetDecOption	Sets the option of Decoder
SetEAN13	Sets the option of EAN-13 Barcode
SetEAN8	Sets the option of EAN-8 Barcode
SetIATA25	Sets the option of IATA25 Barcode
SetINT25	Sets the option of INT25 Barcode
SetKOREAPOST	Sets the option of KOREAPOST Barcode
SetMATRIX25	Sets the option of MATRIX25 Barcode
SetMAXICODE	Sets the option of MAXICODE Barcode
SetMICROPDF	Sets the option of MICROPDF Barcode
SetMSI	Sets the option of MSI Barcode
SetOCR	Sets the option of OCR Barcode
SetOption	Sets the option of imager
SetPDF417	Sets the option of PDF417 Barcode
SetPLANET	Sets the option of PLANET Barcode
SetPLESSEY	Sets the option of PLESSEY Barcode
SetPOSICODE	Sets the option of POSICODE Barcode
SetPOSTNET	Sets the option of POSTNET Barcode
SetQR	Sets the option of QR Barcode
SetRSS	Sets the option of RSS Barcode
SetSTRT25	Sets the option of STRT25 Barcode
SetSymbology	Sets the Enable/Disable of Symbologies

SetSymbologyAll	Enables all of Symbologies
SetSymbologyDefault	Initializes all option of scanner
SetTELEPEN	Sets the option of TELEPEN Barcode
SetUPCA	Sets the option of UPC-A Barcode
SetUPCE	Sets the option of UPC-E Barcode
UnRegHotKey	UnRegHotKey Free Scanner Button as a hot key

4.2.1 CAMCapture

Description

Captures the preview of imaging.

Syntax

```
public bool CAMCapture();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

CAMPreviewStart, CAMPreviewStop

For C++

Library : Imager.lib

Function : BOOL IMAGER_CAMCapture()

Example

None

4.2.2 CAMGetOption

Description

Gets the option of imaging.

Syntax

```
public bool CAMGetOption(out CAM_PARAMS pCamOption);
```

Parameters

pCamOption

Pointer to a CAM_PARAMS structure to be filled in with the camera parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

CAMSetOption

For C++

Library : Imager.lib

Function : BOOL IMAGER_CAMGetOption(PCAM_PARAMS pCamOption)

Example

```
private CAM_PARAMS m_CamParams = new CAM_PARAMS();  
m_Imager.CAMGetOption(out m_CamParams);  
if (m_CamParams.nSaveFormat == 0)  
    RD_JPG.Checked = true;  
else  
    RD_BMP.Checked = true;  
NUD_JPEGQUALITY.Value = m_CamParams.nJpegQuality;  
CB_RESOLUTION.SelectedIndex = m_CamParams.nResolution;  
TB_SAVEFOLDER.Text = m_CamParams.szSaveFolder;  
CB_SAVEMODE.SelectedIndex = m_CamParams.nSaveMode;  
TB_FILENAME.Text = m_CamParams.szFileName;
```

4.2.3 CAMInit

Description

Initializes imaging device.

Syntax

```
public bool CAMInit(IntPtr hPictureWnd);
```

Parameters

hPictureWnd

Window that will show preview.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

CAMUnInit

For C++

Library : Imager.lib

Function : BOOL IMAGER_CAMInit(HWND hPictureWnd)

Example

None

4.2.4 CAMPreviewStart

Description

Starts the preview of imaging.

Syntax

```
public bool CAMPreviewStart();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

SeeAlso

CAMPreviewStop

For C++

Library : Imager.lib

Function : BOOL IMAGER_CAMPreviewStart()

Example

None

4.2.5 CAMPreviewStop

Description

Stops the preview of imaging.

Syntax

```
public bool CAMPreviewStop();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

CAMPreviewStart

For C++

Library : Imager.lib

Function : BOOL IMAGER_CAMPreviewStop()

Example

None

4.2.6 CAMSetOption

Description

Sets the option of imaging.

Syntax

```
public bool CAMSetOption(ref CAM_PARAMS pCamOption);
```

Parameters

pCamOption

Pointer to a CAM_PARAMS structure holding the camera parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

CAMGetOption

For C++

Library : Imager.lib

Function : BOOL IMAGER_CAMSetOption(PCAM_PARAMS pCamOption)

Example

```
private CAM_PARAMS m_CamParams = new CAM_PARAMS();  
if(RD_JPG.Checked == true)  
    m_CamParams.nSaveFormat = 0;  
else  
    m_CamParams.nSaveFormat = 1;  
m_CamParams.nJpegQuality = (int)NUD_JPEGQUALITY.Value;  
m_CamParams.nResolution = CB_RESOLUTION.SelectedIndex;  
m_CamParams.szSaveFolder = TB_SAVEFOLDER.Text;  
m_CamParams.nSaveMode = CB_SAVEMODE.SelectedIndex;  
m_CamParams.szFileName = TB_FILENAME.Text;  
m_Imager.CAMSetOption(ref m_CamParams);
```

4.2.7 CAMUnInit

Description

Uninitializes imaging device.

Syntax

```
public bool CAMUnInit();
```

Parameters

None.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

CAMInit

For C++

Library : Imager.lib

Function : BOOL IMAGER_CAMUnInit()

Example

None

4.2.8 Close

Description

Closes an open imager.

Syntax

```
public bool Close();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Open

For C++

Library : Imager.lib

Function : BOOL IMAGER_Close()

Example

```
for (int i = 0; i < 3; i++)
{
    m_bResult = m_Imager.Close();
    Thread.Sleep(300);
    if (m_bResult == true)
        break;
}
```

4.2.9 GetAZTEC

Description

Gets the option of AZTEC Barcode.

Syntax

```
public bool GetAZTEC(out AZTEC_PARAMS pAztec);
```

Parameters

pAztec

Pointer to a AZTEC_PARAMS structure to be filled in with the AZTEC common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetAZTEC

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetAZTEC(PAZTEC_PARAMS pAztec)

Example

```
public AZTEC_PARAMS m_Aztec = new AZTEC_PARAMS();
m_Imager.GetAZTEC(out m_Aztec);
CB_ENABLE.Checked = m_Aztec.bEnable;
TB_MINLEN.Text = m_Aztec.nMinLen.ToString();
TB_MAXLEN.Text = m_Aztec.nMaxLen.ToString();
```

4.2.10 GetCenteringWindow

Description

Gets Enable/Disable of centering window function.

Syntax

```
public bool GetCenteringWindow(out RECT_PARAM pRect);
```

Parameters

pRect

Defines the region of the image.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCenteringWindow

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCenteringWindow(RECT* pRect)

Example

```
private RECT_PARAM m_Rect = new RECT_PARAM();  
m_Imager.GetCenteringWindow(out m_Rect);  
NUD_TOP.Value = m_Rect.top;  
NUD_LEFT.Value = m_Rect.left;  
NUD_RIGHT.Value = m_Rect.right;  
NUD_BOTTOM.Value = m_Rect.bottom;
```

4.2.11 GetCHINAPOST

Description

Gets the option of CHINAPOST Barcode.

Syntax

```
public bool GetCHINAPOST(out CHINAPOST_PARAMS pChinaPost);
```

Parameters

pChinaPost

Pointer to a CHINAPOST_PARAMS structure to be filled in with the CHINAPOST common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCHINAPOST

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCHINAPOST(PCHINAPOST_PARAMS pChinaPost)

Example

```
public CHINAPOST_PARAMS m_Chinapost = new CHINAPOST_PARAMS();  
m_Imager.GetCHINAPOST(out m_Chinapost);  
CB_ENABLE.Checked = m_Chinapost.bEnable;  
TB_MINLEN.Text = m_Chinapost.nMinLen.ToString();  
TB_MAXLEN.Text = m_Chinapost.nMaxLen.ToString();
```

4.2.12 GetCODABAR

Description

Gets the option of CODABAR Barcode.

Syntax

```
public bool GetCODABAR(out CODABAR_PARAMS pCodabar);
```

Parameters

pCodabar

Pointer to a CODABAR_PARAMS structure to be filled in with the CODABAR common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODABAR

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCODABAR(PCODABAR_PARAMS pCodabar)

Example

```
private CODABAR_PARAMS m_Codabar = new CODABAR_PARAMS();
m_Imager.GetCODABAR(out m_Codabar);
CB_ENABLE.Checked = m_Codabar.bEnable;
CB_CDV.Checked = m_Codabar.bCDV;
CB_XCD.Checked = m_Codabar.bXCD;
CB_XMITSS.Checked = m_Codabar.bXSS;
TB_MINLEN.Text = m_Codabar.nMinLen.ToString();
TB_MAXLEN.Text = m_Codabar.nMaxLen.ToString();
```

4.2.13 GetCODABLOCK

Description

Gets the option of CODABLOCK Barcode.

Syntax

```
public bool GetCODABLOCK(out CODABLOCK_PARAMS pCodablock);
```

Parameters

pCodablock

Pointer to a CODABLOCK_PARAMS structure to be filled in with the CODABLOCK common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODABLOCK

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCODABLOCK(PCODABLOCK_PARAMS pCodablock);

Example

```
public CODABLOCK_PARAMS m_Codablock = new CODABLOCK_PARAMS();
m_Imager.GetCODABLOCK(out m_Codablock);
CB_ENABLE.Checked = m_Codablock.bEnable;
TB_MINLEN.Text = m_Codablock.nMinLen.ToString();
TB_MAXLEN.Text = m_Codablock.nMaxLen.ToString();
```

4.2.14 GetCODE11

Description

Gets the option of CODE11 Barcode.

Syntax

```
public bool GetCODE11(out CODE11_PARAMS pCode11);
```

Parameters

pCode11

Pointer to a CODE11_PARAMS structure to be filled in with the CODE11 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE11

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCODE11(PCODE11_PARAMS pCode11);

Example

```
private CODE11_PARAMS m_Code11 = new CODE11_PARAMS();  
m_Imager.GetCODE11(out m_Code11);  
CB_ENABLE.Checked = m_Code11.bEnable;  
CB_CDV.Checked = m_Code11.bCDV;  
TB_MINLEN.Text = m_Code11.nMinLen.ToString();  
TB_MAXLEN.Text = m_Code11.nMaxLen.ToString();
```

4.2.15 GetCODE128

Description

Gets the option of CODE128 Barcode

Syntax

```
public bool GetCODE128(out CODE128_PARAMS pCode128);
```

Parameters

pCode128

Pointer to a CODE128_PARAMS structure to be filled in with the CODE128 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE128

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCODE128(PCODE128_PARAMS pCode128);

Example

```
public CODE128_PARAMS m_Code128 = new CODE128_PARAMS();  
m_Imager.GetCODE128(out m_Code128);  
CB_ENABLE.Checked = m_Code128.bEnable;  
TB_MINLEN.Text = m_Code128.nMinLen.ToString();  
TB_MAXLEN.Text = m_Code128.nMaxLen.ToString();
```

4.2.16 GetCODE16K

Description

Gets the option of CODE16K Barcode.

Syntax

```
public bool GetCODE16K(out CODE16K_PARAMS pCode16k);
```

Parameters

pCode16k

Pointer to a CODE16K_PARAMS structure to be filled in with the CODE16K common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE16K

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCODE16K(PCODE16K_PARAMS pCode16k);

Example

```
public CODE16K_PARAMS m_Code128 = new CODE16K_PARAMS();
```

```

m_Imager.GetCODE16K(out m_Code16k);
CB_ENABLE.Checked = m_Code16k.bEnable;
TB_MINLEN.Text = m_Code16k.nMinLen.ToString();
TB_MAXLEN.Text = m_Code16k.nMaxLen.ToString();

```

4.2.17 GetCODE39

Description

Gets the option of CODE39 Barcode

Syntax

```
public bool GetCODE39(out CODE39_PARAMS pCode39);
```

Parameters

pCode39

Pointer to a CODE39_PARAMS structure to be filled in with the CODE39 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE39

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCODE39(PCODE39_PARAMS pCode39);

Example

```

private CODE39_PARAMS m_Code39 = new CODE39_PARAMS();
m_Imager.GetCODE39(out m_Code39);
CB_ENABLE.Checked = m_Code39.bEnable;
if (m_Code39.nFormat == 1)
    RD_CODE32.Checked = true;
else if (m_Code39.nFormat == 2)
    RD_TRIOPTIC.Checked = true;
else
    RD_STANDARD.Checked = true;

CB_CDV.Checked = m_Code39.bCDV;

```

```
CB_XCD.Checked = m_Code39.bXCD;
CB_FULLASCII.Checked = m_Code39.bFullASCII;
TB_MINLEN.Text = m_Code39.nMinLen.ToString();
TB_MAXLEN.Text = m_Code39.nMaxLen.ToString();
```

4.2.18 GetCODE49

Description

Gets the option of CODE49 Barcode.

Syntax

```
public bool GetCODE49(out CODE49_PARAMS pCode49);
```

Parameters

pCode49

Pointer to a CODE49_PARAMS structure to be filled in with the CODE49 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE49

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCODE49(PCODE49_PARAMS pCode49);

Example

```
public CODE49_PARAMS m_Code49 = new CODE49_PARAMS();
m_Imager.GetCODE49(out m_Code49);
CB_ENABLE.Checked = m_Code49.bEnable;
TB_MINLEN.Text = m_Code49.nMinLen.ToString();
TB_MAXLEN.Text = m_Code49.nMaxLen.ToString();
```

4.2.19 GetCODE93

Description

Gets the option of CODE93 Barcode.

Syntax

```
public bool GetCODE93(out CODE93_PARAMS pCode93);
```


Parameters

pCode93

Pointer to a CODE93_PARAMS structure to be filled in with the CODE93 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetCODE93

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCODE93(PCODE93_PARAMS pCode93);

Example

```
public CODE93_PARAMS m_Code93 = new CODE93_PARAMS();  
m_Imager.GetCODE93(out m_Code93);  
CB_ENABLE.Checked = m_Code93.bEnable;  
TB_MINLEN.Text = m_Code93.nMinLen.ToString();  
TB_MAXLEN.Text = m_Code93.nMaxLen.ToString();
```

4.2.20 GetCOMPOSITE

Description

Gets the option of COMPOSITE Barcode.

Syntax

```
public bool GetCOMPOSITE(out COMPOSITE_PARAMS pComposite);
```

Parameters

pComposite

Pointer to a COMPOSITE_PARAMS structure to be filled in with the COMPOSITE common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None.

See Also

SetCOMPOSITE

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetCOMPOSITE(PCOMPOSITE_PARAMS pComposite);

Example

```
private COMPOSITE_PARAMS m_Composite = new COMPOSITE_PARAMS();  
m_Imager.GetCOMPOSITE(out m_Composite);  
CB_ENABLE.Checked = m_Composite.bEnable;  
CB_COMPOSITE.Checked = m_Composite.bComposite_Upc;  
TB_MINLEN.Text = m_Composite.nMinLen.ToString();  
TB_MAXLEN.Text = m_Composite.nMaxLen.ToString();
```

4.2.21 GetDATAMATRIX

Description

Gets the option of DATAMATRIX Barcode.

Syntax

```
public bool GetDATAMATRIX(out DATAMATRIX_PARAMS pDataMatirx);
```

Parameters

pDataMatirx

Pointer to a DATAMATRIX_PARAMS structure to be filled in with the DATAMATRIX common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetDATAMATRIX

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetDATAMATRIX(PDATAMATRIX_PARAMS pDataMatirx);

Example

```
public DATAMATRIX_PARAMS m_Datamatrix = new DATAMATRIX_PARAMS();  
m_Imager.GetDATAMATRIX(out m_Datamatrix);  
CB_ENABLE.Checked = m_Datamatrix.bEnable;  
TB_MINLEN.Text = m_Datamatrix.nMinLen.ToString();  
TB_MAXLEN.Text = m_Datamatrix.nMaxLen.ToString();
```

4.2.22 GetDecOption

Description

Gets the option of Decoder.

Syntax

```
public bool GetDecOption(out DECOPTION_PARAMS pDecOption);
```

Parameters

pDecOption

Pointer to a DECOPTION_PARAMS structure to be filled in with the decoder parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetDecOption

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetDecOption(PDECOPTION_PARAMS pDecOption)

Example

None

4.2.23 GetDeviceType

Description

Gets the type of device.

Syntax

```
public SCAN_DEVICE_TYPE GetDeviceType();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : Imager.lib

Function : SCAN_DEVICE_TYPE IMAGER_GetDeviceType()

Example

None

4.2.24 GetEAN13

Description

Gets the option of EAN-13 Barcode.

Syntax

```
public bool GetEAN13(out EAN13_PARAMS pEan13);
```

Parameters

pEan13

Pointer to a EAN13_PARAMS structure to be filled in with the EAN13 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetEAN13

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetEAN13(PEAN13_PARAMS pEan13);

Example

```
private EAN13_PARAMS m_Ean13 = new EAN13_PARAMS();  
m_Imager.GetEAN13(out m_Ean13);  
CB_ENABLE.Checked = m_Ean13.bEnable;  
CB_XCD.Checked = m_Ean13.bXCD;  
CB_ADDON.Checked = m_Ean13.bAddOn;
```

4.2.25 GetEAN8

Description

Gets the option of EAN-8 Barcode

Syntax

```
public bool GetEAN8(out EAN8_PARAMS pEan8);
```

Parameters

pEan8

Pointer to a EAN8_PARAMS structure to be filled in with the EAN8 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetEAN8

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetEAN8(PEAN8_PARAMS pEan8);

Example

```
private EAN8_PARAMS m_Ean8 = new EAN8_PARAMS();  
m_Imager.GetEAN8(out m_Ean8);  
CB_ENABLE.Checked = m_Ean8.bEnable;  
CB_XCD.Checked = m_Ean8.bXCD;  
CB_ADDON.Checked = m_Ean8.bAddOn;
```

4.2.26 GetIATA25

Description

Gets the option of IATA25 Barcode.

Syntax

```
public bool GetIATA25(out IATA25_PARAMS plata25);
```

Parameters

plata25

Pointer to a IATA25_PARAMS structure to be filled in with the IATA25common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetIATA25

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetIATA25(PIATA25_PARAMS pIata25);

Example

```
public IATA25_PARAMS m_Iata25 = new IATA25_PARAMS();  
m_Imager.GetIATA25(out m_Iata25);  
CB_ENABLE.Checked = m_Iata25.bEnable;  
TB_MINLEN.Text = m_Iata25.nMinLen.ToString();  
TB_MAXLEN.Text = m_Iata25.nMaxLen.ToString();
```

4.2.27 GetImagerMode

Description

Gets the mode of imager.

Syntax

```
public int GetImagerMode();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

None

Example

None

4.2.28 GetINT25

Description

Sets the option of INT25 Barcode.

Syntax

```
public bool GetINT25(out INT25_PARAMS pInt25);
```

Parameters

pInt25

Pointer to a INT25_PARAMS structure to be filled in with the INT25 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetINT25

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetINT25(PINT25_PARAMS pInt25);

Example

```
private INT25_PARAMS m_Int25 = new INT25_PARAMS();  
m_Imager.GetINT25(out m_Int25);  
CB_ENABLE.Checked = m_Int25.bEnable;  
CB_CDV.Checked = m_Int25.bCDV;  
CB_XCD.Checked = m_Int25.bXCD;  
TB_MINLEN.Text = m_Int25.nMinLen.ToString();  
TB_MAXLEN.Text = m_Int25.nMaxLen.ToString();
```

4.2.29 GetKOREAPOST

Description

Gets the option of KOREAPOST Barcode.

Syntax

```
public bool GetKOREAPOST(out KOREAPOST_PARAMS pKoreaPost);
```

Parameters

pKoreaPost

Pointer to a KOREAPOST_PARAMS structure to be filled in with the KOREAPOST common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetKOREAPOST

For C++

Library : Imager.lib

Function : `BOOL IMAGER_GetKOREAPOST(PKOREAPOST_PARAMS pKoreaPost);`

Example

```
public KOREAPOST_PARAMS m_Koreapost = new KOREAPOST_PARAMS();  
m_Imager.GetKOREAPOST(out m_Koreapost);  
CB_ENABLE.Checked = m_Koreapost.bEnable;  
TB_MINLEN.Text = m_Koreapost.nMinLen.ToString();  
TB_MAXLEN.Text = m_Koreapost.nMaxLen.ToString();
```

4.2.30 GetMATRIX25

Description

Gets the option of MATRIX25 Barcode.

Syntax

```
public bool GetMATRIX25(out MATRIX25_PARAMS pMatrix25);
```

Parameters

pMatrix25

Pointer to a MATRIX25_PARAMS structure to be filled in with the MATRIX25 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetMATRIX25

For C++

Library : Imager.lib

Function : `BOOL IMAGER_GetMATRIX25(PMATRIX25_PARAMS pMatrix25);`

Example

```
public MATRIX25_PARAMS m_Matrix25 = new MATRIX25_PARAMS();  
m_Imager.GetMATRIX25(out m_Matrix25);  
CB_ENABLE.Checked = m_Matrix25.bEnable;  
TB_MINLEN.Text = m_Matrix25.nMinLen.ToString();  
TB_MAXLEN.Text = m_Matrix25.nMaxLen.ToString();
```


4.2.31 GetMAXICODE

Description

Gets the option of MAXICODE Barcode

Syntax

```
public bool GetMAXICODE(out MAXICODE_PARAMS pMaxiCode);
```

Parameters

pMaxiCode

Pointer to a MAXICODE_PARAMS structure to be filled in with the MAXICODE common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetMAXICODE

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetMAXICODE(PMAXICODE_PARAMS pMaxiCode);

Example

```
public MAXICODE_PARAMS m_Maxicode = new MAXICODE_PARAMS();  
m_Imager.GetMAXICODE(out m_Maxicode);  
CB_ENABLE.Checked = m_Maxicode.bEnable;  
TB_MINLEN.Text = m_Maxicode.nMinLen.ToString();  
TB_MAXLEN.Text = m_Maxicode.nMaxLen.ToString();
```

4.2.32 GetMICROPDF

Description

Gets the option of MICROPDF Barcode

Syntax

```
public bool GetMICROPDF(out MICROPDF_PARAMS pMicroPdf);
```

Parameters

pMicroPdf

Pointer to a MICROPDF_PARAMS structure to be filled in with the MICROPDF common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetMICROPDF

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetMICROPDF(PMICROPDF_PARAMS pMicroPdf);

Example

```
public MICROPDF_PARAMS m_Micropdf = new MICROPDF_PARAMS();  
m_Imager.GetMICROPDF(out m_Micropdf);  
CB_ENABLE.Checked = m_Micropdf.bEnable;  
TB_MINLEN.Text = m_Micropdf.nMinLen.ToString();  
TB_MAXLEN.Text = m_Micropdf.nMaxLen.ToString();
```

4.2.33 GetMSI

Description

Gets the option of MSI Barcode

Syntax

```
public bool GetMSI(out MSI_PARAMS pMsi);
```

Parameters

pMsi

Pointer to a k# MSI_PARAMS structure to be filled in with the MSI common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetMSI

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetMSI(PMSI_PARAMS pMsi);

Example

```
private MSI_PARAMS m_Msi = new MSI_PARAMS();  
m_Imager.GetMSI(out m_Msi);  
CB_ENABLE.Checked = m_Msi.bEnable;  
CB_XCD.Checked = m_Msi.bXCD;  
TB_MINLEN.Text = m_Msi.nMinLen.ToString();  
TB_MAXLEN.Text = m_Msi.nMaxLen.ToString();
```

4.2.34 GetOCR

Description

Gets the option of OCR Barcode.

Syntax

```
public bool GetOCR(out OCR_PARAMS pOcr);
```

Parameters

pOcr

Pointer to a OCR_PARAMS structure to be filled in with the OCR common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetOCR

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetOCR(POCR_PARAMS pOcr);

Example

```
private OCR_PARAMS m_Ocr = new OCR_PARAMS();  
m_Imager.GetOCR(out m_Ocr);  
CB_ENABLE.Checked = m_Ocr.bEnable;  
CB_MODE.SelectedIndex = m_Ocr.nMode;  
TB_TEMPLATE.Text = m_Ocr.szTemplate;  
TB_GROUPG.Text = m_Ocr.szGroupG;  
TB_GROUPH.Text = m_Ocr.szGroupH;  
TB_CHECKCHAR.Text = m_Ocr.szCheckChar;
```

4.2.35 GetOption

Description

Gets the option of imager.

Syntax

```
public bool GetOption(out DECODER_PARAMS pOption);
```

Parameters

pOption

Pointer to a DECODER_PARAMS structure to be filled in with the scanner parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetOption

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetOption(PDECODER_PARAMS pOption);

Example

```
private DECODER_PARAMS m_DecoderParams = new DECODER_PARAMS();
m_Imager.GetOption(out m_DecoderParams);
if (m_DecoderParams.nSound == 1)
    RD_BEEP.Checked = true;
else if (m_DecoderParams.nSound == 2)
    RD_NONE.Checked = true;
else
    RD_DEFAULT.Checked = true;
CB_TIMEOUT.SelectedIndex = m_DecoderParams.nTimeOut - 1;
CB_CENTER.Checked = m_DecoderParams.bCentering;
CB_VIBRATE.Checked = m_DecoderParams.bVibrate;
CB_AIMID.Checked = m_DecoderParams.bXmitAimID;
CB_CONTINUE.Checked = m_DecoderParams.bContinueMode;
CB_HEX.Checked = m_DecoderParams.bHexMode;
CB_LIGHTMODE.SelectedIndex = m_DecoderParams.nLightMode;
```

4.2.36 GetPDF417

Description

Gets the option of PDF417 Barcode.

Syntax

```
public bool GetPDF417(out PDF417_PARAMS pPdf417);
```

Parameters

pPdf417

Pointer to a PDF417_PARAMS structure to be filled in with the PDF417 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetPDF417

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetPDF417(PPDF417_PARAMS pPdf417);

Example

```
public PDF417_PARAMS m_Pdf417 = new PDF417_PARAMS();  
m_Imager.GetPDF417(out m_Pdf417);  
CB_ENABLE.Checked = m_Pdf417.bEnable;  
TB_MINLEN.Text = m_Pdf417.nMinLen.ToString();  
TB_MAXLEN.Text = m_Pdf417.nMaxLen.ToString();
```

4.2.37 GetPLANET

Description

Gets the option of PLANET Barcode.

Syntax

```
public bool GetPLANET(out PLANET_PARAMS pPlanet);
```

Parameters

pPlanet

Pointer to a PLANET_PARAMS structure to be filled in with the PLANET common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetPLANET

For C++

Library : Imager.lib

Function : `BOOL IMAGER_GetPLANET(PPLANET_PARAMS pPlanet);`

Example

```
private PLANET_PARAMS m_Planet = new PLANET_PARAMS();  
m_Imager.GetPLANET(out m_Planet);  
CB_ENABLE.Checked = m_Planet.bEnable;  
CB_XCD.Checked = m_Planet.bXCD;
```

4.2.38 GetPLESSEY

Description

Gets the option of PLESSEY Barcode.

Syntax

```
public bool GetPLESSEY(out PLESSEY_PARAMS pPlessey);
```

Parameters

pPlessey

Pointer to a PLESSEY_PARAMS structure to be filled in with the PLESSEY common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetPLESSEY

For C++

Library : Imager.lib

Function : `BOOL IMAGER_GetPLESSEY(PPLESSEY_PARAMS pPlessey);`

Example

```
public PLESSEY_PARAMS m_Plessey = new PLESSEY_PARAMS();  
m_Imager.GetPLESSEY(out m_Plessey);  
CB_ENABLE.Checked = m_Plessey.bEnable;
```

```
TB_MINLEN.Text = m_Plessey.nMinLen.ToString();  
TB_MAXLEN.Text = m_Plessey.nMaxLen.ToString();
```

4.2.39 GetPOSICODE

Description

Gets the option of POSICODE Barcode.

Syntax

```
public bool GetPOSICODE(out POSICODE_PARAMS pPosiCode);
```

Parameters

pPosiCode

Pointer to a POSICODE_PARAMS structure to be filled in with the POSICODE common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetPOSICODE

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetPOSICODE(PPOSICODE_PARAMS pPosiCode);

Example

```
private POSICODE_PARAMS m_Posicode = new POSICODE_PARAMS();  
m_Imager.GetPOSICODE(out m_Posicode);  
CB_ENABLE.Checked = m_Posicode.bEnable;  
CB_LIMITED1.Checked = m_Posicode.bPosi_Lim1;  
CB_LIMITED2.Checked = m_Posicode.bPosi_Lim2;  
TB_MINLEN.Text = m_Posicode.nMinLen.ToString();  
TB_MAXLEN.Text = m_Posicode.nMaxLen.ToString();
```

4.2.40 GetPOSTNET

Description

Gets the option of POSTNET Barcode.

Syntax

```
public bool GetPOSTNET(out POSTNET_PARAMS pPostNet);
```

Parameters

pPostNet

Pointer to a POSTNET_PARAMS structure to be filled in with the POSTNET common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetPOSTNET

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetPOSTNET(PPOSTNET_PARAMS pPostNet);

Example

```
private POSTNET_PARAMS m_Postnet = new POSTNET_PARAMS();  
m_Imager.GetPOSTNET(out m_Postnet);  
CB_ENABLE.Checked = m_Postnet.bEnable;  
CB_XCD.Checked = m_Postnet.bXCD;
```

4.2.41 GetQR

Description

Gets the option of QR Barcode

Syntax

```
public bool GetQR(out QR_PARAMS pQr);
```

Parameters

pQr

Pointer to a QR_PARAMS structure to be filled in with the QR common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetQR

For C++

Library : Imager.lib

Function : `BOOL IMAGER_GetQR(PQR_PARAMS pQr);`

Example

```
public QR_PARAMS m_Qr = new QR_PARAMS();  
m_Imager.GetQR(out m_Qr);  
CB_ENABLE.Checked = m_Qr.bEnable;  
TB_MINLEN.Text = m_Qr.nMinLen.ToString();  
TB_MAXLEN.Text = m_Qr.nMaxLen.ToString();
```

4.2.42 GetRSS

Description

Gets the option of RSS Barcode.

Syntax

```
public bool GetRSS(out RSS_PARAMS pRss);
```

Parameters

pRss

Pointer to a `RSS_PARAMS` structure to be filled in with the RSS common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

`SetRSS`

For C++

Library : `Imager.lib`

Function : `BOOL IMAGER_GetRSS(PRSS_PARAMS pRss);`

Example

```
private RSS_PARAMS m_Rss = new RSS_PARAMS();  
m_Imager.GetRSS(out m_Rss);  
CB_ENABLE.Checked = m_Rss.bEnable;  
CB_LIMITED.Checked = m_Rss.bRssLim;  
CB_EXPENDED.Checked = m_Rss.bRssExp;  
TB_MINLEN.Text = m_Rss.nMinLen.ToString();  
TB_MAXLEN.Text = m_Rss.nMaxLen.ToString();
```

4.2.43 GetScanData

Description

Gets the ScanData which is read by imager.

Syntax

```
public bool GetScanData(StringBuilder szBarType, StringBuilder szBarData);
```

Parameters

szBarType

Pointer to barcode type.

szBarData

Pointer to barcode data.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetScanData(TCHAR *pszBarType, TCHAR *pszBarData);

Example

None

4.2.44 GetSTRT25

Description

Gets the option of STRT25 Barcode.

Syntax

```
public bool GetSTRT25(out STRT25_PARAMS pStrt25);
```

Parameters

pStrt25

Pointer to a STRT25_PARAMS structure to be filled in with the STRT25 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetSTRT25

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetSTRT25(PSTRT25_PARAMS pStrt25);

Example

```
public STRT25_PARAMS m_Strt25 = new STRT25_PARAMS();  
m_Imager.GetSTRT25(out m_Strt25);  
CB_ENABLE.Checked = m_Strt25.bEnable;  
TB_MINLEN.Text = m_Strt25.nMinLen.ToString();  
TB_MAXLEN.Text = m_Strt25.nMaxLen.ToString();
```

4.2.45 GetSymbology

Description

Gets Enable/Disable of Symbologies.

Syntax

```
public bool GetSymbology(out DECODER pSymbology);
```

Parameters

pSymbology

Pointer to a DECODER structure to be filled in with the symbologies enable or disable.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetSymbology

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetSymbology(PDECODER pSymbology);

Example

```
public DECODER m_Decoder = new DECODER();  
m_Imager.GetSymbology(out m_Decoder);  
BARCODE_ENABLE[0] = m_Decoder.bAZTEC;  
BARCODE_ENABLE[1] = m_Decoder.bCODABAR;
```

```
BARCODE_ENABLE[2] = m_Decoder.bCODE11;  
BARCODE_ENABLE[3] = m_Decoder.bCODE128;  
BARCODE_ENABLE[4] = m_Decoder.bCODE39;  
BARCODE_ENABLE[5] = m_Decoder.bCODE49;  
BARCODE_ENABLE[6] = m_Decoder.bCODE93;  
BARCODE_ENABLE[7] = m_Decoder.bCOMPOSITE;  
BARCODE_ENABLE[8] = m_Decoder.bDATAMATRIX;  
BARCODE_ENABLE[9] = m_Decoder.bEAN8;  
BARCODE_ENABLE[10] = m_Decoder.bEAN13;  
BARCODE_ENABLE[11] = m_Decoder.bINT25;  
BARCODE_ENABLE[12] = m_Decoder.bMAXICODE;  
BARCODE_ENABLE[13] = m_Decoder.bMICROPDF;  
BARCODE_ENABLE[14] = m_Decoder.bOCR;  
BARCODE_ENABLE[15] = m_Decoder.bPDF417;  
BARCODE_ENABLE[16] = m_Decoder.bPOSTNET;  
BARCODE_ENABLE[17] = m_Decoder.bQR;  
BARCODE_ENABLE[18] = m_Decoder.bRSS;  
BARCODE_ENABLE[19] = m_Decoder.bUPCA;  
BARCODE_ENABLE[20] = m_Decoder.bUPCE;  
BARCODE_ENABLE[21] = m_Decoder.bISBT;  
BARCODE_ENABLE[22] = m_Decoder.bBPO;  
BARCODE_ENABLE[23] = m_Decoder.bCANPOST;  
BARCODE_ENABLE[24] = m_Decoder.bAUSPOST;  
BARCODE_ENABLE[25] = m_Decoder.bIATA25;  
BARCODE_ENABLE[26] = m_Decoder.bCODABLOCK;  
BARCODE_ENABLE[27] = m_Decoder.bJAPOST;  
BARCODE_ENABLE[28] = m_Decoder.bPLANET;  
BARCODE_ENABLE[29] = m_Decoder.bDUTCHPOST;  
BARCODE_ENABLE[30] = m_Decoder.bMSI;  
BARCODE_ENABLE[31] = m_Decoder.bTLCODE39;  
BARCODE_ENABLE[32] = m_Decoder.bSTRT25;  
BARCODE_ENABLE[33] = m_Decoder.bMATRIX25;  
BARCODE_ENABLE[34] = m_Decoder.bPLESSEY;  
BARCODE_ENABLE[35] = m_Decoder.bCHINAPOST;
```

```

BARCODE_ENABLE[36] = m_Decoder.bKOREAPOST;
BARCODE_ENABLE[37] = m_Decoder.bTELEPEN;
BARCODE_ENABLE[38] = m_Decoder.bCODE16K;
BARCODE_ENABLE[39] = m_Decoder.bPOSI CODE;
BARCODE_ENABLE[40] = m_Decoder.bCOUPONCODE;
BARCODE_ENABLE[41] = m_Decoder.bUSPS4CB;
BARCODE_ENABLE[42] = m_Decoder.bIDTAG;
BARCODE_ENABLE[43] = m_Decoder.bLABEL;
BARCODE_ENABLE[44] = m_Decoder.bGS1_128;
BARCODE_ENABLE[45] = m_Decoder.bHANXIN;
BARCODE_ENABLE[46] = m_Decoder.bGRIDMATRIX;

```

4.2.46 GetTELEPEN

Description

Gets the option of TELEPEN Barcode.

Syntax

```
public bool GetTELEPEN(out TELEPEN_PARAMS pTelepen);
```

Parameters

pTelepen

Pointer to a TELEPEN_PARAMS structure to be filled in with the TELEPEN common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetTELEPEN

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetTELEPEN(PTELEPEN_PARAMS pTelepen);

Example

```

private TELEPEN_PARAMS m_Telepen = new TELEPEN_PARAMS();
m_Imager.GetTELEPEN(out m_Telepen);
CB_ENABLE.Checked = m_Telepen.bEnable;
CB_NUMERIC.Checked = m_Telepen.bNumeric;

```

```
TB_MINLEN.Text = m_Telepen.nMinLen.ToString();  
TB_MAXLEN.Text = m_Telepen.nMaxLen.ToString();
```

4.2.47 GetUPCA

Description

Gets the option of UPC-A Barcode.

Syntax

```
public bool GetUPCA(out UPCA_PARAMS pUpca);
```

Parameters

pUpca

Pointer to a UPCA_PARAMS structure to be filled in with the UPC-A common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetUPCA

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetUPCA(PUPCA_PARAMS pUpca);

Example

```
private UPCA_PARAMS m_Upca = new UPCA_PARAMS();  
m_Imager.GetUPCA(out m_Upca);  
CB_ENABLE.Checked = m_Upca.bEnable;  
CB_XCD.Checked = m_Upca.bXCD;  
CB_XNUM.Checked = m_Upca.bXNum;  
CB_ADDON.Checked = m_Upca.bAddOn;
```

4.2.48 GetUPCE

Description

Gets the option of UPC-E Barcode.

Syntax

```
public bool GetUPCE(out UPCE_PARAMS pUpce);
```

Parameters

pUpce

Pointer to a UPCE_PARAMS structure to be filled in with the UPC-E common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetUPCE

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetUPCE(PUPCE_PARAMS pUpce);

Example

```
private UPCE_PARAMS m_Upce = new UPCE_PARAMS();  
m_Imager.GetUPCE(out m_Upce);  
CB_ENABLE.Checked = m_Upce.bEnable;  
CB_XCD.Checked = m_Upce.bXCD;  
CB_XNUM.Checked = m_Upce.bXNum;  
CB_ADDON.Checked = m_Upce.bAddOn;
```

4.2.49 GetVersionInfo

Description

Gets the information of imager driver and dll version.

Syntax

```
public string GetVersionInfo();
```

Parameters

pszVersion

Pointer to a TCHAR to be filled in with the version info.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

none

For C++

Library : Imager.lib

Function : BOOL IMAGER_GetVersionInfo(TCHAR* pszVersion);

Example

None

4.2.50 IQGetBarcodeData

Description

Gets the ScanData which is read by imager.

Syntax

```
public bool IQGetBarcodeData(StringBuilder szBarData);
```

Parameters

szBarData

Pointer to barcode data.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : Imager.lib

Function : BOOL IMAGER_IQGetBarcodeData(TCHAR *pszBarData);

Example

None

4.2.51 IQGetOption

Description

Gets the option of IQ Imaging

Syntax

```
public bool IQGetOption(out IQ_PARAMS plQOption);
```

Parameters

plQOption

Pointer to a IQ_PARAMS structure to be filled in with the IQ Imaging parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

IQSetOption

For C++

Library : Imager.lib

Function : BOOL IMAGER_IQGetOption(PIQ_PARAMS plQOption);

Example

```
m_Imager.IQGetOption(out m_IQParams);  
if (m_IQParams.nIQType == 0)  
    RD_AZTEC.Checked = true;  
else  
    RD_CODE39.Checked = true;  
TB_SAVEFOLDER.Text = m_IQParams.szSaveFolder;  
CB_SAVEMODE.SelectedIndex = m_IQParams.nSaveMode;  
TB_FILENAME.Text = m_IQParams.szFileName;
```

4.2.52 IQImagingStart

Description

Starts IQ Imaging.

Syntax

```
public bool IQImagingStart();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

IQImagingStop

For C++

Library : Imager.lib

Function : BOOL IMAGER_IQImagingStart();

Example

None

4.2.53 IQImagingStop

Description

Stops IQ Imaging.

Syntax

```
public bool IQImagingStop();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

[IQImagingStart](#)

For C++

Library : Imager.lib

Function : BOOL IMAGER_IQImagingStop();

Example

None

4.2.54 IQInit

Description

Initializes IQ imaging

Syntax

```
public bool IQInit(IntPtr hPictureWnd);
```

Parameters

hPictureWnd

Window that will show preview.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

IQUnInit

For C++

Library : Imager.lib

Function : BOOL IMAGER_IQInit(HWND hPictureWnd);

Example

None

4.2.55 IQSetOption

Description

Sets the option of IQ Imaging.

Syntax

```
public bool IQSetOption(ref IQ_PARAMS pIQOption);
```

Parameters

pIQOption

Pointer to a IQ_PARAMS structure holding the IQ Imaging parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

IQGetOption

For C++

Library : Imager.lib

Function : BOOL IMAGER_IQSetOption(PIQ_PARAMS pIQOption);

Example

```
public STRT25_PARAMS m_Strt25 = new STRT25_PARAMS();  
if (RD_AZTEC.Checked == true)  
    m_IQParams.nIQType = 0;  
else  
    m_IQParams.nIQType = 1;  
m_IQParams.szSaveFolder = TB_SAVEFOLDER.Text;  
m_IQParams.nSaveMode = CB_SAVEMODE.SelectedIndex;  
m_IQParams.szFileName = TB_FILENAME.Text;
```

```
m_Imager.IQSetOption(ref m_IQParams);
```

4.2.56 IQUnInit

Description

Uninitializes IQ imaging.

Syntax

```
public bool IQUnInit();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

IQInit

For C++

Library : Imager.lib

Function : BOOL IMAGER_IQUnInit();

Example

None

4.2.57 Open

Description

Opens a imager.

Syntax

```
public bool Open();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Close

For C++

Library : Imager.lib

Function : BOOL IMAGER_Open();

Example

None

4.2.58 Read

Description

Starts the beaming of imager

Syntax

```
public bool Read();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

ReadCancel

For C++

Library : Imager.lib

Function : BOOL IMAGER_Read();

Example

None

4.2.59 ReadCancel

Description

Stops the beaming of imager

Syntax

```
public bool ReadCancel();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Read

For C++

Library : Imager.lib

Function : BOOL IMAGER_ReadCancel();

Example

None

4.2.60 RegHotKey

Description

Registers Scanner Button as a hot key.

Syntax

```
public bool RegHotKey(int id, uint vk, bool SyncMode);
```

Parameters

id

Identifier of the hot key. No other hot key in the calling thread should have the same identifier. An application must specify a value in the range 0x0000 through 0xBFFF.

vk

The value of Virtual Key.

SyncMode

The state is either true = Sync Mode, false = ASync Mode.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

UnRegHotKey

For C++

None

Example

```
public const int m_nHotKeyCE = 133;    // VK_F22(WinCE)
public const int m_nHotKeyWM = 125;    // VK_F14(WM)
// Get Device Type
```

```

m_DeviceType = m_Imager.GetDeviceType();

// OS Type

if ((m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3SMART_CE) || (m_DeviceType ==
SCAN_DEVICE_TYPE.DEVICE_M3GREEN) || (m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3T) ||
(m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3POS))

    m_bWinCE = true;

//Scanner Open

m_Imager.Open();

if (m_bWinCE == true)

    m_Imager.RegHotKey(1, m_nHotKeyCE, m_bSyncMode);

else

    m_Imager.RegHotKey(1, m_nHotKeyWM, m_bSyncMode);

```

4.2.61 SetAZTEC

Description

Sets the option of AZTEC Barcode.

Syntax

```
public bool SetAZTEC(ref AZTEC_PARAMS pAztec);
```

Parameters

pAztec

Pointer to a AZTEC_PARAMS structure holding the AZTEC common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetAZTEC

For C++

Library : Imager.lib

Function : IMAGER_SetAZTEC(PAZTEC_PARAMS pAztec);

Example

```

public AZTEC_PARAMS m_Aztec = new AZTEC_PARAMS();

m_Aztec.bEnable = CB_ENABLE.Checked;

m_Aztec.nMinLen = Convert.ToInt32(TB_MINLEN.Text);

m_Aztec.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);

```

```
m_Imager.SetAZTEC(ref m_Aztec);
```

4.2.62 SetCenteringWindow

Description

Enables centering window function.

Syntax

```
public bool SetCenteringWindow(ref RECT_PARAM pRect);
```

Parameters

pRect

Defines the region of the image.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCenteringWindow

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetCenteringWindow(RECT* pRect);

Example

```
private RECT_PARAM m_Rect = new RECT_PARAM();  
m_Rect.top = (int)NUD_TOP.Value;  
m_Rect.left = (int)NUD_LEFT.Value;  
m_Rect.right = (int)NUD_RIGHT.Value;  
m_Rect.bottom = (int)NUD_BOTTOM.Value;  
m_Imager.SetCenteringWindow(ref m_Rect);
```

4.2.63 SetCHINAPOST

Description

Sets the option of CHINAPOST Barcode.

Syntax

```
public bool SetCHINAPOST(ref CHINAPOST_PARAMS pChinaPost);
```

Parameters

pChinaPost

Pointer to a CHINAPOST_PARAMS structure holding the CHINAPOST common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCHINAPOST

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetCHINAPOST(PCHINAPOST_PARAMS pChinaPost);

Example

```
public CHINAPOST_PARAMS m_Chinapost = new CHINAPOST_PARAMS();  
m_Chinapost.bEnable = CB_ENABLE.Checked;  
m_Chinapost.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Chinapost.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetCHINAPOST(ref m_Chinapost);
```

4.2.64 SetCODABAR

Description

Sets the option of CODABAR Barcode.

Syntax

```
public bool SetCODABAR(ref CODABAR_PARAMS pCodabar);
```

Parameters

pCodabar

Pointer to a CODABAR_PARAMS structure holding the CODABAR common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODABAR

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetCODABAR(PCODABAR_PARAMS pCodabar);

Example

```
private CODABAR_PARAMS m_Codabar = new CODABAR_PARAMS();  
m_Codabar.bEnable = CB_ENABLE.Checked;  
m_Codabar.bCDV = CB_CDV.Checked;  
m_Codabar.bXCD = CB_XCD.Checked;  
m_Codabar.bXSS = CB_XMITSS.Checked;  
m_Codabar.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Codabar.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetCODABAR(ref m_Codabar);
```

4.2.65 SetCODABLOCK

Description

Sets the option of CODABLOCK Barcode.

Syntax

```
public bool SetCODABLOCK(ref CODABLOCK_PARAMS pCodablock);
```

Parameters

pCodablock

Pointer to a CODABLOCK_PARAMS structure holding the CODABLOCK common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODABLOCK

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetCODABLOCK(PCODABLOCK_PARAMS pCodablock);

Example

```
public CODABLOCK_PARAMS m_Codablock = new CODABLOCK_PARAMS();  
m_Codablock.bEnable = CB_ENABLE.Checked;  
m_Codablock.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Codablock.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetCODABLOCK(ref m_Codablock);
```

4.2.66 SetCODE11

Description

Sets the option of CODE11 Barcode.

Syntax

```
public bool SetCODE11(ref CODE11_PARAMS pCode11);
```

Parameters

pCode11

Pointer to a CODE11_PARAMS structure holding the CODE11 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE11

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetCODE11(PCODE11_PARAMS pCode11);

Example

```
private CODE11_PARAMS m_Code11 = new CODE11_PARAMS();  
m_Code11.bEnable = CB_ENABLE.Checked;  
m_Code11.bCDV = CB_CDV.Checked;  
m_Code11.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Code11.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetCODE11(ref m_Code11);
```

4.2.67 SetCODE128

Description

Sets the option of CODE128 Barcode.

Syntax

```
public bool SetCODE128(ref CODE128_PARAMS pCode128);
```

Parameters

pCode128

Pointer to a CODE128_PARAMS structure holding the CODE128 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE128

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetCODE128(PCODE128_PARAMS pCode128);

Example

```
public CODE128_PARAMS m_Code128 = new CODE128_PARAMS();  
m_Code128.bEnable = CB_ENABLE.Checked;  
m_Code128.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Code128.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetCODE128(ref m_Code128);
```

4.2.68 SetCODE16K

Description

Sets the option of CODE16K Barcode.

Syntax

```
public bool SetCODE16K(ref CODE16K_PARAMS pCode16k);
```

Parameters

pCode16k

Pointer to a CODE16K_PARAMS structure holding the CODE16K common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE16K

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetCODE16K(PCODE16K_PARAMS pCode16k);

Example

```
public CODE16K_PARAMS m_Code16k = new CODE16K_PARAMS();
```

```

m_Code16k.bEnable = CB_ENABLE.Checked;
m_Code16k.nMinLen = Convert.ToInt32(TB_MINLEN.Text);
m_Code16k.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);
m_Imager.SetCODE16K(ref m_Code16k);

```

4.2.69 SetCODE39

Description

Sets the option of CODE39 Barcode.

Syntax

```
public bool SetCODE39(ref CODE39_PARAMS pCode39);
```

Parameters

pCode39

Pointer to a CODE39_PARAMS structure holding the CODE39 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE39

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetCODE39(PCODE39_PARAMS pCode39);

Example

```

private CODE39_PARAMS m_Code39 = new CODE39_PARAMS();
m_Code39.bEnable = CB_ENABLE.Checked;
if (RD_CODE32.Checked == true)
    m_Code39.nFormat = 1;
else if (RD_TRIOPTIC.Checked == true)
    m_Code39.nFormat = 2;
else
    m_Code39.nFormat = 0;
m_Code39.bCDV      = CB_CDV.Checked;
m_Code39.bXCD      = CB_XCD.Checked;
m_Code39.bFullASCII = CB_FULLASCII.Checked;

```

```
m_Code39.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Code39.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetCODE39(ref m_Code39);
```

4.2.70 SetCODE49

Description

Sets the option of CODE49 Barcode.

Syntax

```
public bool SetCODE49(ref CODE49_PARAMS pCode49);
```

Parameters

pCode49

Pointer to a CODE49_PARAMS structure holding the CODE49 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE49

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetCODE49(PCODE49_PARAMS pCode49);

Example

```
public CODE49_PARAMS m_Code49 = new CODE49_PARAMS();  
m_Code49.bEnable = CB_ENABLE.Checked;  
m_Code49.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Code49.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetCODE49(ref m_Code49);
```

4.2.71 SetCODE93

Description

Sets the option of CODE93 Barcode.

Syntax

```
public bool SetCODE93(ref CODE93_PARAMS pCode93);
```

Parameters

pCode93

Pointer to a CODE93_PARAMS structure holding the CODE93 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCODE93

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetCODE93(PCODE93_PARAMS pCode93);

Example

```
public CODE93_PARAMS m_Code93 = new CODE93_PARAMS();  
m_Code93.bEnable = CB_ENABLE.Checked;  
m_Code93.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Code93.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetCODE93(ref m_Code93);
```

4.2.72 SetCOMPOSITE

Description

Sets the option of COMPOSITE Barcode.

Syntax

```
public bool SetCOMPOSITE(ref COMPOSITE_PARAMS pComposite);
```

Parameters

pComposite

Pointer to a COMPOSITE_PARAMS structure holding the COMPOSITE common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCOMPOSITE

For C++

Library : Imager.lib

Function : `BOOL IMAGER_SetCOMPOSITE(PCOMPOSITE_PARAMS pComposite);`

Example

```
private COMPOSITE_PARAMS m_Composite = new COMPOSITE_PARAMS();  
m_Composite.bEnable = CB_ENABLE.Checked;  
m_Composite.bComposite_Upc = CB_COMPOSITE.Checked;  
m_Composite.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Composite.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetCOMPOSITE(ref m_Composite);
```

4.2.73 SetDATAMATRIX

Description

Sets the option of DATAMATRIX Barcode.

Syntax

```
public bool SetDATAMATRIX(ref DATAMATRIX_PARAMS pDataMatrix);
```

Parameters

pDataMatrix

Pointer to a DATAMATRIX_PARAMS structure holding the DATAMATRIX common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetDATAMATRIX

For C++

Library : Imager.lib

Function : `BOOL IMAGER_SetDATAMATRIX(PDATAMATRIX_PARAMS pDataMatrix);`

Example

```
public DATAMATRIX_PARAMS m_Datamatrix = new DATAMATRIX_PARAMS();  
m_Datamatrix.bEnable = CB_ENABLE.Checked;  
m_Datamatrix.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Datamatrix.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetDATAMATRIX(ref m_Datamatrix);
```


4.2.74 SetDecOption

Description

Sets the option of Decoder.

Syntax

```
public bool SetDecOption(ref DECOPTION_PARAMS pDecOption);
```

Parameters

pDecOption

Pointer to a DECOPTION_PARAMS structure holding the decoder parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetDecOption

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetDecOption(PDECOPTION_PARAMS pDecOption);

Example

None

4.2.75 SetEAN13

Description

Sets the option of EAN-13 Barcode.

Syntax

```
public bool SetEAN13(ref EAN13_PARAMS pEan13);
```

Parameters

pEan13

Pointer to a EAN13_PARAMS structure holding the EAN-13 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetEAN13

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetEAN13(PEAN13_PARAMS pEan13);

Example

```
private EAN13_PARAMS m_Ean13 = new EAN13_PARAMS();  
m_Ean13.bEnable = CB_ENABLE.Checked;  
m_Ean13.bXCD = CB_XCD.Checked;  
m_Ean13.bAddOn = CB_ADDON.Checked;  
m_Imager.SetEAN13(ref m_Ean13);
```

4.2.76 SetEAN8

Description

Sets the option of EAN-8 Barcode.

Syntax

```
public bool SetEAN8(ref EAN8_PARAMS pEan8);
```

Parameters

pEan8

Pointer to a EAN8_PARAMS structure holding the EAN-8 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetEAN8

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetEAN8(PEAN8_PARAMS pEan8);

Example

```
private EAN8_PARAMS m_Ean8 = new EAN8_PARAMS();  
m_Ean8.bEnable = CB_ENABLE.Checked;  
m_Ean8.bXCD = CB_XCD.Checked;  
m_Ean8.bAddOn = CB_ADDON.Checked;  
m_Imager.SetEAN8(ref m_Ean8);
```

4.2.77 SetIATA25

Description

Sets the option of IATA25 Barcode.

Syntax

```
public bool SetIATA25(ref IATA25_PARAMS plata25);
```

Parameters

plata25

Pointer to a IATA25_PARAMS structure holding the IATA25 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetIATA25

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetIATA25(PIATA25_PARAMS plata25);

Example

```
public IATA25_PARAMS m_lata25 = new IATA25_PARAMS();  
m_lata25.bEnable = CB_ENABLE.Checked;  
m_lata25.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_lata25.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetIATA25(ref m_lata25);
```

4.2.78 SetINT25

Description

Sets the option of INT25 Barcode.

Syntax

```
public bool SetINT25(ref INT25_PARAMS plnt25);
```

Parameters

plnt25

Pointer to a INT25_PARAMS structure holding the INT25 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetINT25

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetINT25(PINT25_PARAMS pInt25);

Example

```
private INT25_PARAMS m_Int25 = new INT25_PARAMS();  
m_Int25.bEnable = CB_ENABLE.Checked;  
m_Int25.bCDV = CB_CDV.Checked;  
m_Int25.bXCD = CB_XCD.Checked;  
m_Int25.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Int25.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetINT25(ref m_Int25);
```

4.2.79 SetKOREAPOST

Description

Sets the option of KOREAPOST Barcode.

Syntax

```
public bool SetKOREAPOST(ref KOREAPOST_PARAMS pKoreaPost);
```

Parameters

pKoreaPost

Pointer to a KOREAPOST_PARAMS structure holding the KOREAPOST common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetKOREAPOST

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetKOREAPOST(PKOREAPOST_PARAMS pKoreaPost);

Example

```
public KOREAPOST_PARAMS m_Koreapost = new KOREAPOST_PARAMS();  
m_Koreapost.bEnable = CB_ENABLE.Checked;  
m_Koreapost.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Koreapost.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetKOREAPOST(ref m_Koreapost);
```

4.2.80 SetMATRIX25

Description

Sets the option of MATRIX25 Barcode.

Syntax

```
public bool SetMATRIX25(ref MATRIX25_PARAMS pMatrix25);
```

Parameters

pMatrix25

Pointer to a MATRIX25_PARAMS structure holding the MATRIX25 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetMATRIX25

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetMATRIX25(PMATRIX25_PARAMS pMatrix25);

Example

```
public MATRIX25_PARAMS m_Matrix25 = new MATRIX25_PARAMS();  
m_Matrix25.bEnable = CB_ENABLE.Checked;  
m_Matrix25.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Matrix25.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetMATRIX25(ref m_Matrix25);
```

4.2.81 SetMAXICODE

Description

Sets the option of MAXICODE Barcode.

Syntax

```
public bool SetMAXICODE(ref MAXICODE_PARAMS pMaxiCode);
```

Parameters

pMaxiCode

Pointer to a MAXICODE_PARAMS structure holding the MAXICODE common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetMAXICODE

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetMAXICODE(PMAXICODE_PARAMS pMaxiCode);

Example

```
public MAXICODE_PARAMS m_Maxicode = new MAXICODE_PARAMS();  
m_Maxicode.bEnable = CB_ENABLE.Checked;  
m_Maxicode.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Maxicode.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetMAXICODE(ref m_Maxicode);
```

4.2.82 SetMICROPDF

Description

Sets the option of MICROPDF Barcode.

Syntax

```
public bool SetMICROPDF(ref MICROPDF_PARAMS pMicroPdf);
```

Parameters

pMicroPdf

Pointer to a MICROPDF_PARAMS structure holding the MICROPDF common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetMICROPDF

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetMICROPDF(PMICROPDF_PARAMS pMicroPdf);

Example

```
public MICROPDF_PARAMS m_Micropdf = new MICROPDF_PARAMS();  
m_Micropdf.bEnable = CB_ENABLE.Checked;  
m_Micropdf.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Micropdf.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetMICROPDF(ref m_Micropdf);
```

4.2.83 SetMSI

Description

Sets the option of MSI Barcode.

Syntax

```
public bool SetMSI(ref MSI_PARAMS pMsi);
```

Parameters

pMsi

Pointer to a MSI_PARAMS structure holding the MSI common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetMSI

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetMSI(PMSI_PARAMS pMsi);

Example

```
private MSI_PARAMS m_Msi = new MSI_PARAMS();  
m_Msi.bEnable = CB_ENABLE.Checked;  
m_Msi.bXCD = CB_XCD.Checked;  
m_Msi.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Msi.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetMSI(ref m_Msi);
```

4.2.84 SetOCR

Description

Sets the option of OCR Barcode.

Syntax

```
public bool SetOCR(ref OCR_PARAMS pOcr);
```

Parameters

pOcr

Pointer to a OCR_PARAMS structure holding the OCR common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetOCR

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetOCR(POCR_PARAMS pOcr);

Example

```
private OCR_PARAMS m_Ocr = new OCR_PARAMS();  
m_Ocr.bEnable = CB_ENABLE.Checked;  
m_Ocr.nMode = CB_MODE.SelectedIndex;  
m_Ocr.szTemplate = TB_TEMPLATE.Text;  
m_Ocr.szGroupG = TB_GROUPG.Text;  
m_Ocr.szGroupH = TB_GROUPH.Text;  
m_Ocr.szCheckChar = TB_CHECKCHAR.Text;  
m_Imager.SetOCR(ref m_Ocr);
```

4.2.85 SetOption

Description

Sets the option of imager.

Syntax

```
public bool SetOption(ref DECODER_PARAMS pOption);
```

Parameters

pOption

Pointer to a DECODER_PARAMS structure holding the imager parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetOption

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetOption(PDECODER_PARAMS pOption);

Example

```
private DECODER_PARAMS m_DecoderParams = new DECODER_PARAMS();
if (RD_BEEP.Checked == true)
    m_DecoderParams.nSound = 1;
else if (RD_NONE.Checked == true)
    m_DecoderParams.nSound = 2;
else
    m_DecoderParams.nSound = 0;
m_DecoderParams.nTimeOut = CB_TIMEOUT.SelectedIndex + 1;
m_DecoderParams.bCentering = CB_CENTER.Checked;
m_DecoderParams.bVibrate = CB_VIBRATE.Checked;
m_DecoderParams.bXmitAimID = CB_AIMID.Checked;
m_DecoderParams.bContinueMode = CB_CONTINUE.Checked;
m_DecoderParams.bHexMode = CB_HEX.Checked;
m_DecoderParams.nLightMode = CB_LIGHTMODE.SelectedIndex;
m_Imager.SetOption(ref m_DecoderParams);
```

4.2.86 SetPDF417

Description

Sets the option of PDF417 Barcode.

Syntax

```
public bool SetPDF417(ref PDF417_PARAMS pPdf417);
```

Parameters

pPdf417

Pointer to a PDF417_PARAMS structure holding the PDF417 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetPDF417

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetPDF417(PPDF417_PARAMS pPdf417);

Example

```
public PDF417_PARAMS m_Pdf417 = new PDF417_PARAMS();  
m_Pdf417.bEnable = CB_ENABLE.Checked;  
m_Pdf417.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Pdf417.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetPDF417(ref m_Pdf417);
```

4.2.87 SetPLANET

Description

Sets the option of PLANET Barcode.

Syntax

```
public bool SetPLANET(ref PLANET_PARAMS pPlanet);
```

Parameters

pPlanet

Pointer to a PLANET_PARAMS structure holding the PLANET common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetPLANET

For C++

Library : Imager.lib

Function : `BOOL IMAGER_SetPLANET(PPLANET_PARAMS pPlanet);`

Example

```
private PLANET_PARAMS m_Planet = new PLANET_PARAMS();  
m_Planet.bEnable = CB_ENABLE.Checked;  
m_Planet.bXCD = CB_XCD.Checked;  
m_Imager.SetPLANET(ref m_Planet);
```

4.2.88 SetPLESSEY

Description

Sets the option of PLESSEY Barcode.

Syntax

```
public bool SetPLESSEY(ref PLESSEY_PARAMS pPlessey);
```

Parameters

pPlessey

Pointer to a PLESSEY_PARAMS structure holding the PLESSEY common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetPLESSEY

For C++

Library : `Imager.lib`

Function : `BOOL IMAGER_SetPLESSEY(PPLESSEY_PARAMS pPlessey);`

Example

```
public PLESSEY_PARAMS m_Plessey = new PLESSEY_PARAMS();  
m_Plessey.bEnable = CB_ENABLE.Checked;  
m_Plessey.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Plessey.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetPLESSEY(ref m_Plessey);
```

4.2.89 SetPOSICODE

Description

Sets the option of POSICODE Barcode.

Syntax

```
public bool SetPOSICODE(ref POSICODE_PARAMS pPosiCode);
```

Parameters

pPosiCode

Pointer to a POSICODE_PARAMS structure holding the POSICODE common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetPOSICODE

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetPOSICODE(PPOSICODE_PARAMS pPosiCode);

Example

```
private POSICODE_PARAMS m_Posicode = new POSICODE_PARAMS();  
m_Posicode.bEnable = CB_ENABLE.Checked;  
m_Posicode.bPosi_Lim1 = CB_LIMITED1.Checked;  
m_Posicode.bPosi_Lim2 = CB_LIMITED2.Checked;  
m_Posicode.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Posicode.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetPOSICODE(ref m_Posicode);
```

4.2.90 SetPOSTNET

Description

Sets the option of POSTNET Barcode.

Syntax

```
public bool SetPOSTNET(ref POSTNET_PARAMS pPostNet);
```

Parameters

pPostNet

Pointer to a POSTNET_PARAMS structure holding the POSTNET common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetPOSTNET

For C++

Library : Imager.lib

Function : `BOOL IMAGER_SetPOSTNET(PPOSTNET_PARAMS pPostNet);`

Example

```
public PPOSTNET_PARAMS pPostnet = new PPOSTNET_PARAMS();
pPostnet.bEnable = CB_ENABLE.Checked;
pPostnet.bXCD = CB_XCD.Checked;
m_Imager.SetPPOSTNET(ref pPostnet);
```

4.2.91 SetQR

Description

Sets the option of QR Barcode.

Syntax

```
public bool SetQR(ref QR_PARAMS pQr);
```

Parameters

pQr

Pointer to a QR_PARAMS structure holding the QR common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetQR

For C++

Library : Imager.lib

Function : `BOOL IMAGER_SetQR(PQR_PARAMS pQr);`

Example

```
public QR_PARAMS m_Qr = new QR_PARAMS();
m_Qr.bEnable = CB_ENABLE.Checked;
m_Qr.nMinLen = Convert.ToInt32(TB_MINLEN.Text);
m_Qr.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);
```

```
m_Imager.SetQR(ref m_Qr);
```

4.2.92 SetRSS

Description

Sets the option of RSS Barcode.

Syntax

```
public bool SetRSS(ref RSS_PARAMS pRss);
```

Parameters

pRss

Pointer to a RSS_PARAMS structure holding the RSS common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetRSS

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetRSS(PRSS_PARAMS pRss);

Example

```
private RSS_PARAMS m_Rss = new RSS_PARAMS();  
m_Rss.bEnable = CB_ENABLE.Checked;  
m_Rss.bRssLim = CB_LIMITED.Checked;  
m_Rss.bRssExp = CB_EXPENDED.Checked;  
m_Rss.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Rss.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetRSS(ref m_Rss);
```

4.2.93 SetSTRT25

Description

Sets the option of STRT25 Barcode.

Syntax

```
public bool SetSTRT25(ref STRT25_PARAMS pStrt25);
```

Parameters

pStrt25

Pointer to a STRT25_PARAMS structure holding the STRT25 common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetSTRT25

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetSTRT25(PSTRT25_PARAMS pStrt25);

Example

```
public STRT25_PARAMS m_Strt25 = new STRT25_PARAMS();  
m_Strt25.bEnable = CB_ENABLE.Checked;  
m_Strt25.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Strt25.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetSTRT25(ref m_Strt25);
```

4.2.94 SetSymbology

Description

Sets the Enable/Disable of Symbologies.

Syntax

```
public bool SetSymbology(ref DECODER pSymbology);
```

Parameters

pSymbology

Pointer to a DECODER structure holding the symbologies enable or disable.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetSymbology

For C++

Library : Imager.lib

Function : `BOOL IMAGER_SetSymbology(PDECODER pSymbology);`

Example

```
public DECODER m_Decoder = new DECODER();  
m_Decoder.bAZTEC = LV_SYMLIST.Items[0].Checked;  
m_Decoder.bCODABAR = LV_SYMLIST.Items[1].Checked;  
m_Decoder.bCODE11 = LV_SYMLIST.Items[2].Checked;  
m_Decoder.bCODE128 = LV_SYMLIST.Items[3].Checked;  
m_Decoder.bCODE39 = LV_SYMLIST.Items[4].Checked;  
m_Decoder.bCODE49 = LV_SYMLIST.Items[5].Checked;  
m_Decoder.bCODE93 = LV_SYMLIST.Items[6].Checked;  
m_Decoder.bCOMPOSITE = LV_SYMLIST.Items[7].Checked;  
m_Decoder.bDATAMATRIX = LV_SYMLIST.Items[8].Checked;  
m_Decoder.bEAN8 = LV_SYMLIST.Items[9].Checked;  
m_Decoder.bEAN13 = LV_SYMLIST.Items[10].Checked;  
m_Decoder.bINT25 = LV_SYMLIST.Items[11].Checked;  
m_Decoder.bMAXICODE = LV_SYMLIST.Items[12].Checked;  
m_Decoder.bMICROPDF = LV_SYMLIST.Items[13].Checked;  
m_Decoder.bOCR = LV_SYMLIST.Items[14].Checked;  
m_Decoder.bPDF417 = LV_SYMLIST.Items[15].Checked;  
m_Decoder.bPOSTNET = LV_SYMLIST.Items[16].Checked;  
m_Decoder.bQR = LV_SYMLIST.Items[17].Checked;  
m_Decoder.bRSS = LV_SYMLIST.Items[18].Checked;  
m_Decoder.bUPCA = LV_SYMLIST.Items[19].Checked;  
m_Decoder.bUPCE = LV_SYMLIST.Items[20].Checked;  
m_Decoder.bISBT = LV_SYMLIST.Items[21].Checked;  
m_Decoder.bBPO = LV_SYMLIST.Items[22].Checked;  
m_Decoder.bCANPOST = LV_SYMLIST.Items[23].Checked;  
m_Decoder.bAUSPOST = LV_SYMLIST.Items[24].Checked;  
m_Decoder.biATA25 = LV_SYMLIST.Items[25].Checked;  
m_Decoder.bCODABLOCK = LV_SYMLIST.Items[26].Checked;  
m_Decoder.bJAPOST = LV_SYMLIST.Items[27].Checked;  
m_Decoder.bPLANET = LV_SYMLIST.Items[28].Checked;  
m_Decoder.bDUTCHPOST = LV_SYMLIST.Items[29].Checked;  
m_Decoder.bMSI = LV_SYMLIST.Items[30].Checked;
```



```

m_Decoder.bTLCODE39 = LV_SYMLIST.Items[31].Checked;
m_Decoder.bSTRT25 = LV_SYMLIST.Items[32].Checked;
m_Decoder.bMATRIX25 = LV_SYMLIST.Items[33].Checked;
m_Decoder.bPLESSEY = LV_SYMLIST.Items[34].Checked;
m_Decoder.bCHINAPOST = LV_SYMLIST.Items[35].Checked;
m_Decoder.bKOREAPOST = LV_SYMLIST.Items[36].Checked;
m_Decoder.bTELEPEN = LV_SYMLIST.Items[37].Checked;
m_Decoder.bCODE16K = LV_SYMLIST.Items[38].Checked;
m_Decoder.bPOSICODE = LV_SYMLIST.Items[39].Checked;
m_Decoder.bCOUPONCODE = LV_SYMLIST.Items[40].Checked;
m_Decoder.bUSPS4CB = LV_SYMLIST.Items[41].Checked;
m_Decoder.bIDTAG = LV_SYMLIST.Items[42].Checked;
m_Decoder.bLABEL = LV_SYMLIST.Items[43].Checked;
m_Decoder.bGS1_128 = LV_SYMLIST.Items[44].Checked;
m_Decoder.bHANXIN = LV_SYMLIST.Items[45].Checked;
m_Decoder.bGRIDMATRIX = LV_SYMLIST.Items[46].Checked;
m_Imager.SetSymbology(ref m_Decoder);

```

4.2.95 SetSymbologyAll

Description

Enables all of Symbologies.

Syntax

```
public bool SetSymbologyAll();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetSymbologyDefault

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetSymbologyAll();

Example

None

4.2.96 SetSymbologyDefault

Description

Initializes all option of scanner.

Syntax

```
public bool SetSymbologyDefault();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetSymbologyAll

For C++

Library : Imager.lib

Function : bool SetSymbologyDefault()

Example

None

4.2.97 SetTELEPEN

Description

Sets the option of TELEPEN Barcode

Syntax

```
public bool SetTELEPEN(ref TELEPEN_PARAMS pTelepen);
```

Parameters

pTelepen

Pointer to a TELEPEN_PARAMS structure holding the TELEPEN common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetTELEPEN

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetTELEPEN(PTELEPEN_PARAMS pTelepen);

Example

```
private TELEPEN_PARAMS m_Telepen = new TELEPEN_PARAMS();  
m_Telepen.bEnable = CB_ENABLE.Checked;  
m_Telepen.bNumeric = CB_NUMERIC.Checked;  
m_Telepen.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Telepen.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetTELEPEN(ref m_Telepen);
```

4.2.98 SetUPCA

Description

Sets the option of UPC-A Barcode

Syntax

```
public bool SetUPCA(ref UPCA_PARAMS pUpca);
```

Parameters

pUpca

Pointer to a UPCA_PARAMS structure holding the UPC-A common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetUPCA

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetUPCA(PUPCA_PARAMS pUpca);

Example

```
private UPCA_PARAMS m_Upca = new UPCA_PARAMS();  
m_Upca.bEnable = CB_ENABLE.Checked;  
m_Upca.bXCD = CB_XCD.Checked;
```

```
m_Upca.bXNum = CB_XNUM.Checked;  
m_Upca.bAddOn = CB_ADDON.Checked;  
m_Imager.SetUPCA(ref m_Upca);
```

4.2.99 SetUPCE

Description

Sets the option of UPC-E Barcode

Syntax

```
public bool SetUPCE(ref UPCE_PARAMS pUpce);
```

Parameters

pUpce

Pointer to a UPCE_PARAMS structure holding the UPC-E common parameters.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetUPCE

For C++

Library : Imager.lib

Function : BOOL IMAGER_SetUPCE(PUPCE_PARAMS pUpce);

Example

```
private UPCE_PARAMS m_Upce = new UPCE_PARAMS();  
m_Upce.bEnable = CB_ENABLE.Checked;  
m_Upce.bXCD = CB_XCD.Checked;  
m_Upce.bXNum = CB_XNUM.Checked;  
m_Upce.bAddOn = CB_ADDON.Checked;  
m_Imager.SetUPCE(ref m_Upce);
```

4.2.100 UnRegHotKey

Description

Free Scanner Button as a hot key.

Syntax

```
public bool UnRegHotKey(int id);
```

Parameters

id

Identifier of the hot key to be freed.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

RegHotKey

For C++

None

Example

None

4.3 CAMERA (CE)

Status Return value

Please refer to the below table for the status value definition.

Status Value Definition	Code	Meaning
TRUE	1	Success
FALSE	0	General Error

Constant Value
<pre>//Image Effect public const int OPTION_IMAGE_NOMAL_EFFECT = 0; public const int OPTION_IMAGE_SEPIA_EFFECT = 1; public const int OPTION_IMAGE_BLACKWHITE_EFFECT = 2; public const int OPTION_IMAGE_NEGATIVE_EFFECT = 3; public const int OPTION_IMAGE_UVRED_EFFECT = 4; public const int OPTION_IMAGE_SMART_AQUA_EFFECT = 4; public const int OPTION_IMAGE_UVBLUE_EFFECT = 5; public const int OPTION_IMAGE_UVGREEN_EFFECT = 6; //Resolution public const int OPTION_RESOLUTION_2048X1536 = 5; public const int OPTION_RESOLUTION_1600X1200 = 4; public const int OPTION_RESOLUTION_1280X1024 = 0; public const int OPTION_RESOLUTION_640X480 = 1; public const int OPTION_RESOLUTION_320X240 = 2; //Save Format public const int OPTION_SAVE_FORMAT_BMP = 0; public const int OPTION_SAVE_FORMAT_JPG = 1; //Image File Name Prefix public const int OPTION_IMAGE_FILENAME_PREF_DATE = 0; public const int OPTION_IMAGE_FILENAME_PREF_SERIAL = 1; // AF Status Massege public const int WM_STATUS_AF = (0x0400 + 0x1002); // GPS Status Messasge public const int WM_GPS_ON = (WM_USER + 0x1003); // Convert 2D Imager public const int WM_GPS_ON = (WM_USER + 232); // Capture Status Message public const int WM_GPS_ON = (WM_USER + 5);;</pre>
Enum
<pre>public enum AF_STATUS { AF_STATUS_IDLE = -1,</pre>

```

    AF_STATUS_FINISH = 0,
    AF_STATUS_START = 1,
}

public enum MODEL_TYPE
{
    MODEL_GREEN = 0,
    MODEL_T = 1,
    MODEL_POS = 2,
    MODEL_SMART = 3,

    MODEL_UNKNOWN = 4,
}

enum CAP_STATUS
{
    CAP_STATUS_START = 0x1,
    CAP_STATUS_IMG_TRANSFORM = 0x2,
    CAP_STATUS_IMG_SCALE_UP = 0x4,
    CAP_STATUS_IMG_SAVE = 0x8,
    CAP_STATUS_END = 0x10,
    CAP_STATUS_ALL_PROCESS_END = 0x16,
    CAP_STATUS_COPY_DATA = 0x20
};

enum SCANNER_TYPE
{
    SCANNER_OPTICON,
    SCANNER_SYMBOL,
    SCANNER_INTERMEC,
    SCANNER_HHP,
    SCANNER_ERR = -1
};

```

Structure

```

public struct CAMERA_OPTION
{
    public int nImgbalance;
    public int nImgEffect;
    public int nImgRotatesave;
    public int nImgRotateview;
    public int nInFormat;
    public int nJpegQuality;
    public int nNamePrefix;
    public int nResolution;
    public int nSaveFormat;
}

public struct ExifInfo

```

```
{  
    public string Make;  
    public string Model;  
    public string TitleName;  
}
```


Functions for Camera (CE)

Name	Description
Open	Opens the Camera.
Close	Closes the Camera.
PreviewStart	Starts viewing the Preview screen.
PreviewStop	Stops viewing the Preview screen.
Capture	Captures the Still Picture.
GetLastSaveFilePath	Gets name of the latest file.
AutoFocus	Focuses automatically.
EnableAutoAF	Performs AF by automatically recognizing preview image changes.
Zoom	Zooms in and/or out.
SetCameraOption	Sets the Camera options.
GetCameraOption	Gets currently set Options of Camera.
Brightness	Changes the Brightness of Camera.
FlashOn	Turns on the Flash.
FlashOff	Turns off the Flash.
RawData	Gets the raw data of Preview screen.
InsertExifInformation	Insert EXIF information to Jpeg image.
UseGPSExifData	Insert GPS information in EXIF Information.
GetScannerType	Gets Scanner Type of device.
GetVersion	Gets dll version.

4.3.1 Open

Description

Opens the Camera.

Syntax

```
public MODEL_TYPE Open(IntPtr hMainWnd, IntPtr hPreviewWnd);
```

Parameters

hMainWnd

Windows Handle of Camera Application

hPreviewWnd

Windows handle to show image

Return Value

MODEL_TYPE is enum type value. Gets model name of device.

Remarks

None

See Also

Close

For C++

Library: CAM.lib

Function : MODEL_TYPE CAM_Open(HWND hMainWnd, HWND hPreviewWnd);

Example

```
MODEL_TYPE m_Model;

m_Model = CAM_Open(m_hWnd, m_ctrPreview.m_hWnd) ;

if(m_Model == MODEL_TYPE.MODEL_UNKNOWN)
{
    // Error
    return;
}
```

4.3.2 Close

Description

Closes the Camera.

Syntax

```
public bool Close();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Open

For C++

Library: CAM.lib

Function : BOOL CAM_Close();

Example

```
m_Cam.Close();
```

4.3.3 PreviewStart

Description

Starts viewing the Preview screen.

Syntax

```
public bool PreviewStart();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

PreviewStop

For C++

Library: CAM.lib

Function : BOOL CAM_PreviewStart();

Example

```
m_Cam.PreviewStart();
```

4.3.4 PreviewStop

Description

Stops viewing the Preview screen.

Syntax

```
public bool PreviewStop()
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

PreviewStart

For C++

Library: CAM.lib

Function : BOOL CAM_PreviewStop();

Example

```
m_Cam.PreviewStop();
```

4.3.5 Capture

Description

Captures the Still Picture.

Syntax

```
public bool Capture(string strPath)
```

Parameters

strFilePath

Input the name and/or route of file to be stored.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library: CAM.lib

Function : BOOL CAM_Capture(LPCTSTR strFilePath);

Example

```
if(!m_bCapturing)
{
    m_Cam.Capture("Flash Disk\\Camera\\");
}
```

4.3.6 GetLastSaveFilePath

Description

Gets name of the latest file.

Syntax

```
public bool GetLastSaveFilePath(out string strFilePath);
```

Parameters

strFilePath

Obtains the last captured picture's name.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Capture

For C++

Library: CAM.lib

Function : BOOL CAM_GetLastSaveFilePath(LPTSTR strOutFilePath);

Example

```
String strOutFileName = "";
m_Cam.GetLastSaveFilePath(out strOutFileName);
```

4.3.7 AutoFocus

Description

Focuses automatically.

Syntax

```
public bool AutoFocus();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

Supported in M3 SMART and M3 T.

See Also

EnableAutoAF

For C++

Library: CAM.lib

Function : BOOL CAM_AutoFocus();

Example

```
m_Cam.AutoFocus();
```

4.3.8 EnableAutoAF

Description

Performs AF by automatically recognizing preview image changes.

Syntax

```
public bool EnableAutoAF(bool bEnable)
```

Parameters

bEnable

Performs AF by automatically recognizing preview image changes. (supported in M3T)

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

Only supported in M3T.

See Also

AutoFocus

For C++

Library: CAM.lib

Function : BOOL CAM_EnableAutoAF(BOOL bEnable);

Example

```
m_Cam.EnableAutoAF(true);
```

4.3.9 Zoom

Description

Zooms in and/or out.

Syntax

```
public bool Zoom(int nZoom);
```

Parameters

index

Sets zoom function according to Index.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library: CAM.lib

Function : BOOL CAM_Zoom(int nZoom);

Example

```
m_Cam.Zoom(1);
```

4.3.10 SetCameraOption

Description

Sets the Camera options.

Syntax

```
public bool SetCameraOption(ref Cam.CAMERA_OPTION option, string strSavePath)
```

Parameters

option

CAMERA_OPTION Structure pointer inputs Option Value to set.

strSaveFolder

Inputs file name and/or route to be stored.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetCameraOption

For C++

Library: CAM.lib

Function : `BOOL CAM_SetCameraOption(LPCAMERA_OPTION option, LPTSTR strSaveFolder);`

Example

```
CAMERA_OPTION CamOption = new CAMERA_OPTION();  
GetCameraOption(out CamOption, tzSavePath);  
CamOption.nSaveFormat = 1;  
m_Cam.CameraOption(ref CamOption, "Flash Disk\\Camera\\");
```

4.3.11 GetCameraOption

Description

Gets currently set Options of Camera.

Syntax

```
public bool GetCameraOption(out Cam.CAMERA_OPTION option, StringBuilder strSavePath)
```

Parameters

option

CAMERA_OPTION Structure pointer obtains the Option Value.

strSaveFolder

Obtains file name and/or route to be stored.

Return Value

Nonzero indicates success. Zero indicates failure

Remarks

None

See Also

SetCameraOption

For C++

Library: CAM.lib

Function : `BOOL CAM_GetCameraOption(LPCAMERA_OPTION option, LPTSTR strSaveFolder);`

Example

```
StringBuilder tzSavePath = new StringBuilder(260);  
CAMERA_OPTION CamOption = new CAMERA_OPTION();  
m_Cam.GetCameraOption(out CamOption, tzSavePath);
```


4.3.12 Brightness

Description

Changes the Brightness of Camera.

Syntax

```
public bool Brightness(int nBrightness);
```

Parameters

nBrightness

Sets the brightness.

Return Value

Nonzero indicates success. Zero indicates failure

Remarks

None

See Also

SetCameraOption, GetCameraOption

For C++

Library: CAM.lib

Function : BOOL CAM_Brightness(int nBrightness);

Example

```
m_Cam.Brightness(1);
```

4.3.13 FlashOn

Description

Turns on the Flash.

Syntax

```
public bool FlashOn();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

FlashOff

For C++

Library: CAM.lib

Function : BOOL CAM_FlashOn();

Example

```
m_Cam.FlashOn();
```

4.3.14 FlashOff

Description

Turns off the Flash.

Syntax

```
public bool FlashOff()
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

FlashOn

For C++

Library: CAM.lib

Function : BOOL CAM_FlashOff();

Example

```
m_Cam.FlashOff();
```

4.3.15 RawData

Description

Obtains data of Preview screen.

Syntax

```
public void RawData(byte[] data)
```

Parameters

data

Obtains data of current preview screen.

Return Value

void

Remarks

Not supported in M3 SMART.

See Also

None

For C++

Library: CAM.lib

Function : void CAM_RawData(byte* data);

Example

None

4.3.16 InsertExifInformation

Description

Inserts EXIF information to Jpeg image.

Syntax

```
public bool InsertExifInformation(string FileName, Cam.ExifInfo e);
```

Parameters

FileName

File name of JPEG which EXIF Information is to be stored.

Info

ExifInfo Structure.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

UseGPSExifData

For C++

Library: CAM.lib

Function : BOOL CAM_InsertExifInformation(TCHAR* FileName, ExifInfo info);

Example

```
ExifInfo info = new ExifInfo();
```

```
info.TitleName = "Picture0.jpg";
```

```
info.Make = "M3 Mobile Co.Ltd";
```

```
info.Model = "M3 SMART";
```

```
m_Cam.InsertExifInformation("Flash Disk\\Camera\\Photo\\Picture0.jpg", info);
```

4.3.17 UseGPSExifData

Description

Inserts GPS information to EXIF Information.

Syntax

```
public bool UseGPSExifData(int bUse);
```

Parameters

bUse

Whether the GPS information is included in EXIF Information or not.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

Once GPS function is set to use, GPS satellite should be visualized. GPS information starts to be inserted in EXIF information if GPS signal can be received from satellite.

See Also

InsertExifInformation

For C++

Library: CAM.lib

Function : BOOL CAM_UseGPSExifData(BOOL bUse);

Example

```
m_Cam.UseGPSExifData(true);
```

4.3.18 GetScannerType

Description

Gets Scanner Type of device.

Syntax

```
public SCANNER_TYPE GetScannerType();
```

Parameters

None

Return Value

SCANNER_TYPE is Enum type value. Gets scanner type of device.

Remarks

None

See Also

None

For C++

Library: CAM.lib

Function : SCANNER_TYPE CAM_GetScannerType()

Example

```
SCANNER_TYPE type = m_Cam.GetScannerType();
```

4.3.19 GetVersion

Description

Gets current version of Camera DLL.

Syntax

```
public bool GetVersion(StringBuilder strDllVersion, StringBuilder strReleaseDate, StringBuilder strPixels);
```

Parameters

strDllVersion

Obtains DLL version.

strReleaseDate

Obtains released date of DLL.

strPixels

Obtains pixels of Camera.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library: CAM.lib

Function : BOOL CAM_GetVersion()

Example

None

4.4 CAMERA (WM)

Status Return value

Please refer to the below table for the status value definition.

Status Value Definition	Code	Meaning
TRUE	1	Success
FALSE	0	General Error

Enum
<pre>public enum MODEL_TYPE { MODEL_SKY = 0, MODEL_MM3 = 1, MODEL_ORANGE = 2, MODEL_SMART = 3, MODEL_ORANGE_PLUS = 4, MODEL_UNKONWN = 5, } enum SCANNER_TYPE { SCANNER_OPTICON, SCANNER_SYMBOL, SCANNER_INTERMEC, SCANNER_HHP }; public enum CAMERA_MODE { STILL_MODE = 0, VIDEO_MODE = 1, CLOSE_MODE = 2, } public enum VIDEO_TYPE { VIDEO_ASF = 0, VIDEO_WMV = 1, } public enum AF_MODE { AF_MANUAL = 0, AF_ALWAYS = 1, AF_AUTO = 2, } public enum AF_STATUS { </pre>

```
    AF_STATUS_READY = 0,  
    AF_STATUS_START = 1,  
    AF_STATUS_FINISH = 2,  
}  
public enum WHITE_BALANCE  
{  
    WB_Auto = 0,  
    WB_Sunny = 1,  
    WB_Cloudy = 2,  
    WB_Fluorescent = 3,  
    WB_Incandescent = 4,  
}  
  
enum SAVE_TYPE {  
    SAVE_DATE=0,  
    SAVE_CUSTIM=1  
};
```

Structure

```
public struct ExifInfo  
{  
    public string Make;  
    public string Model;  
    public string TitleName;  
}
```

Functions for Camera (WM)

Name	Description
Open	Opens the Camera.
Close	Closes the Camera.
SetPreviewWindow	Sets the size of Preview screen.
PreviewStart	Starts viewing the Preview screen.
PreviewStop	Stops viewing the Preview screen.
GetRegCapturePath	Get save filename.(old version)
GetRegCapturePathEx	Get save filename.
SetRegCapturePathEx	Set save filename.
Capture	Captures the Still Picture. (old version)
CaptureEx	Captures the Still Picture.
VideoStart	Starts Videoing.
VideoStop	Stops Videoing. (old version)
VideoStopEx	Stops Videoing.
GetFlashState	Flash On/Off check
AlwaysFlash	Turns on the Flash all the time.
CaptureFlash	Sets whether the flash is on or not when capturing picture.
AutoFocus	Focuses automatically.
SetAfMode	Sets AF Mode.
Zoom	Zooms in and/or out.
SetResolution	Sets Resolution of picture.
GetResolution	Gets setting value of resolution.
SetQuality	Sets quality of Jpeg Still Shot.
GetQuality	Gets quality value of Jpeg Still shot.
SetBrightness	Sets brightness of camera.
GetBrightness	Gets brightness of camera.
SetWhiteBalance	Sets white balance of camera.
GetWhiteBalance	Gets white balance of camera.
SetContrast	Sets contrast of camera.
GetContrast	Gets contrast of camera.
SetSharpness	Sets sharpness of camera.

<u>GetSharpness</u>	Gets sharpness of camera.
<u>SetHistoEqual</u>	Sets whether performing of Histogram Equalization when taking pictures.
<u>GetHistoEqual</u>	Gets whether performing of Histogram Equalization when taking pictures.
<u>RegisterPreview</u>	Registers Callback function to get the preview screen.
<u>InsertExifInformation</u>	Insert EXIF information to Jpeg image.
<u>UseGPSExifData</u>	Insert GPS information in EXIF Information.
<u>GetModelType</u>	Gets model type of device.
<u>GetScannerType</u>	Gets Scanner Type of device.
<u>GetVersion</u>	Gets dll version.

4.4.1 Open

Description

Opens the Camera.

Syntax

```
public bool Open(IntPtr hWnd, CAMERA_MODE cameramode, VIDEO_TYPE videotype);
```

Parameters

hWnd

Windows Handle of Camera Application

cameramode

Sets Camera mode. CAMERA_MODE is enum data. If this value is STILL_MODE, setting camera to make still image. If this value is VIDEO_MODE, setting camera to make video.

Videotype

Sets Video mode. VIDEO_TYPE is enum data. If this value is VIDEO_WMV, videos file is saved *.wmv. If this value is VIDEO_ASF, videos file is saved *.asf.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None.

See Also

Close

For C++

Library: CAM.lib

Function : BOOL CAM_Open(HWND hWnd, CAMERA_MODE cameramode, VIDEO_TYPE videotype);

Example

```
using CamNet;

Cam m_Cam = new Cam();

MODEL_TYPE model = m_Cam.Open(this.Handle, pictureBox_Preview.Handle);

if (model == MODEL_TYPE.MODEL_UNKNOWN)
{
    MessageBox.Show("Camera can not open");
    return;
}
```

4.4.2 Close

Description

Closes the Camera.

Syntax

```
public bool Close();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Open

For C++

Library: CAM.lib

Function : BOOL CAM_Close();

Example

```
m_Cam.Close();
```

4.4.3 SetPreviewWindow

Description

Sets the size of Preview screen.

Syntax

```
public bool SetPreviewWindow(long left, long top, long width, long height);
```

Parameters

left

Assigns left location of preview screen.

top

Assigns upper location of preview screen.

width

Assigns width of preview screen.

height

Assigns height of preview screen.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

PreviewStart, PreviewStop

For C++

Library: CAM.lib

Function : BOOL CAM_SetPreviewWindow(long left, long top, long width, long height);

Example

```
m_Cam.SetPreviewWindow(80,90,320,240);    // in MM3
```

4.4.4 PreviewStart

Description

Starts viewing the Preview screen.

Syntax

```
public bool PreviewStart()
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

PreviewStop

For C++

Library: CAM.lib

Function : BOOL CAM_PreviewStart();

Example

```
m_Cam.PreviewStart();
```

4.4.5 PreviewStop

Description

Stops viewing the Preview screen.

Syntax

```
public bool PreviewStop();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

PreviewStart

For C++

Library: CAM.lib

Function : BOOL CAM_PreviewStop();

Example

```
m_Cam.PreviewStop();
```

4.4.6 GetRegCapturePath

Description

Get save filename

Syntax

```
public bool GetRegCapturePath(StringBuilder strPath);
```

Parameters

strPath

File name in capture.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetRegCapturePath

For C++

Library: CAM.lib

Function : BOOL CAM_GetRegCapturePath(TCHAR* tzFullName);

Example

```
StringBuilder tzFilename = new StringBuilder(260);
```

```
GetRegCapturePath(m_tzFilename)
```

GetRegCapturePathEx

Description

Get save filename

Syntax

```
public bool GetRegCapturePathEx(StringBuilder strPath, StringBuilder strName);
```

Parameters

strPath

save path

strName

save filename

Return Value

Nonzero indicates success.Zero indicates failure.

Remarks

None

See Also

SetRegCapturePath

For C++

Library: CAM.lib

Function : BOOL CAM_GetRegCapturePathEx(TCHAR* tzPath, TCHAR* tzName);

Example

```
StringBuilder tzFilename = new StringBuilder(260);
```

```
GetRegCapturePath(m_tzFilename)
```

4.4.7 SetRegCapturePathEx

Description

Set save filename

Syntax

```
public bool SetRegCapturePath(StringBuilder strPath, StringBuilder strName);
```

Parameters

strPath

save path

strName

save filename

Return Value

Nonzero indicates success.Zero indicates failure.

Remarks

None

See Also

GetRegCapturePathEx

For C++

Library: CAM.lib

Function : BOOL CAM_SetRegCapturePathEx(TCHAR* tzPath, TCHAR* tzName);

Example

```
StringBuilder tzFilename = new StringBuilder(260);  
GetRegCapturePath(m_tzFilename)
```

4.4.8 Capture

Description

Captures the Still Picture.

Syntax

```
public bool Capture(string tzInFileName, char[] tzOutFileName, bool bAutoInit);
```

Parameters

tzInFileName

Inputs file name and/or route to be stored.

tzOutFileName

Obtains file name.

bAutoInit

Initiates camera automatically after capturing pictures.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library: CAM.lib

Function : BOOL CAM_Capture(const TCHAR* tzInFileName, TCHAR* tzOutFileName, BOOL bAutoInit);

Example

```
char[] strOutFile = new char[260];  
m_Cam.Capture("Flash Disk\\Camera\\", strOutFile, true);
```

4.4.9 CaptureEx

Description

Captures the Still Picture.

Syntax

```
public bool CaptureEx(SAVE_TYPE nSaveType, bool bAutoInit, char[] tzOutFileName);
```

Parameters

nSaveType

SAVE_TYPE

bAutoInit

Initiates camera automatically after capturing pictures.

tzOutFileName

Obtains file name..

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library: CAM.lib

Function : BOOL CAM_CaptureEx(SAVE_TYPE nSaveType, BOOL bAutoInit, TCHAR* tzOutFileName);

Example

```
char[] strOutFile = new char[260];  
m_Cam.CaptureEx(SAVE_TYPE.SAVE_DATE, true, strOutFile);
```

4.4.10 VideoStart

Description

Starts Video capturing.

Syntax

```
public bool VideoStart();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

VideoStop

For C++

Library: CAM.lib

Function : BOOL CAM_VideoStart();

Example

```
m_Cam.VideoStart();
```

4.4.11 VideoStop

Description

Stops Video capturing

Syntax

```
public bool VideoStop(char[] tzInFileName, char[] tzOutFileName);
```

Parameters

tzInFileName

Inputs file name and/or route to be stored.

tzOutFileName

Obtains file name.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

VideoStart

For C++

Library: CAM.lib

Function : BOOL CAM_VideoStop(const TCHAR* tzInFileName, TCHAR* tzOutFileName);

Example

```
char[] strOutFile = new char[260];
```

```
m_Cam.VideoStop("Flash Disk\\Camera\\", strOutFile);
```

4.4.12 VideoStopEx

Description

Stops Video capturing

Syntax

```
public bool VideoStopEx(SAVE_TYPE nSaveType, char[] tzOutFileName);
```

Parameters

nSaveType

SAVE_TYPE.

tzOutFileName

Obtains file name.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

VideoStart

For C++

Library: CAM.lib

Function : BOOL CAM_VideoStopEx(SAVE_TYPE nSaveType, TCHAR* tzOutFileName);

Example

```
char[] strOutFile = new char[260];
```

```
m_Cam.VideoStopEx(SAVE_TYPE.SAVE_DATE, strOutFile);
```

4.4.13 GetFlashState

Description

Get Flash Status.

Syntax

```
public bool GetFlashState();
```

Parameters

None.

Return Value

Nonzero indicates flash on. Zero indicates flash off.

Remarks

None.

See Also

None.

For C++

Library: CAM.lib

Function : BOOL CAM_GetFlashState();

Example

```
bool bFlashOn = GetFlashState();
```

4.4.14 AlwaysFlash

Description

Turns on the Flash all the time.

Syntax

```
public bool AlwaysFlash(bool bOn);
```

Parameters

bOn

Sets the Flash mode.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

CaptureFlash

For C++

Library: CAM.lib

Function : BOOL CAM_AlwaysFlash(BOOL bOn);

Example

```
m_Cam.AlwaysFlash(true);
```

4.4.15 CaptureFlash

Description

Sets whether the flash is on or not when capturing picture.

Syntax

```
public bool CaptureFlash(int bOn);
```

Parameters

bOn

Sets the Flash mode.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

AlwaysFlash

For C++

Library: CAM.lib

Function : BOOL CAM_CaptureFlash(BOOL bOn);

Example

```
m_Cam.CaptureFlash(true);
```

4.4.16 AutoFocus

Description

Focuses automatically.

Syntax

```
public bool AutoFocus();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SetAfMode

For C++

Library: CAM.lib

Function : BOOL CAM_AutoFocus();

Example

```
CAM_AutoFocus();
```

4.4.17 SetAfMode

Description

Sets AF Mode.

Syntax

```
public AF_MODE SetAfMode(AF_MODE mode);
```

Parameters

mode

Sets camera mode to perform the AutoFocus. Af_MODE is enum data

0 : Manual AutoFocus – Performs AF function when user sets.

1 : Always AutoFocus – Performs AF function before capturing pictures.

3 : Auto AutoFocus – Performs AF function as recognizing the preview screen change.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

AutoFocus

For C++

Library: CAM.lib

Function : AF_MODE CAM_SetAfMode(AF_MODE mode);

Example

```
m_Cam.SetAfMode(0);
```

4.4.18 Zoom

Description

Zooms in and/or out.

Syntax

```
public int Zoom(int index);
```

Parameters

index

Sets zoom function according to Index.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library: CAM.lib

Function :BOOL CAM_Zoom(int index);

Example

```
m_Cam.Zoom(1);
```

4.4.19 SetResolution

Description

Sets Resolution of picture.

Syntax

```
public int SetResolution(int nResolution);
```

Parameters

nResolution

The range of value is 0 to 6 .

(0 : 176*144, 1 : 320*240, 2 : 352*288, 3 : 640*480, 4 : 800*600, 5 : 1280*960, 6 : 1600*1200)

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetResolution

For C++

Library: CAM.lib

Function : BOOL CAM_SetResolution(int nResolution);

Example

```
m_Cam.SetResolution(0);
```

4.4.20 GetResolution

Description

Gets setting value of resolution.

Syntax

```
public int GetResolution();
```

Parameters

None

Return Value

Gets setting value of resolution.

(0 : 176*144, 1 : 320*240, 2 : 352*288, 3 : 640*480, 4 : 800*600, 5 : 1280*960, 6 : 1600*1200)

Remarks

None

See Also

SetResolution

For C++

Library: CAM.lib

Function : int CAM_GetResolution();

Example

```
int nResolution = m_Cam.GetResolution();
```

4.4.21 SetQuality

Description

Sets quality of Jpeg Still Shot.

Syntax

```
public bool SetQuality(int nQuality);
```

Parameters

nQuality

The value specifies quality (0 : Low, 1 : Normal, 2 : High)

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetQuality

For C++

Library: CAM.lib

Function : BOOL CAM_SetQuality(int nQuality);

Example

```
m_Cam.SetQuality(0);
```

4.4.22 GetQuality

Description

Gets quality value of Jpeg Still shot.

Syntax

```
public int GetQuality();
```

Parameters

None

Return Value

Gets quality value of Jpeg Still Shot.

Remarks

None

See Also

SetQuality

For C++

Library: CAM.lib

Function : int CAM_GetQuality();

Example

```
int nQuality = m_Cam.GetQuality();
```

4.4.23 SetBrightness

Description

Sets brightness of camera.

Syntax

```
public bool SetBrightness(int nBrightness);
```

Parameters

nBrightness

Value's range is +4~-4.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetBrightness

For C++

Library: CAM.lib

Function : BOOL CAM_SetBrightness(int nBrightness);

Example

```
m_Cam.SetBrightness(0);
```

4.4.24 GetBrightness

Description

Gets brightness of camera.

Syntax

```
public int GetBrightness();
```

Parameters

None

Return Value

Gets birhgtness of camera.

Remarks

None

See Also

SetBrightness

For C++

Library: CAM.lib

Function : int CAM_GetBrightness();

Example

```
int nBrightness = m_Cam.GetBrightness();
```

4.4.25 SetWhiteBalance

Description

Sets white balance of camera.

Syntax

```
public bool SetWhiteBalance(WHITE_BALANCE nWhiteBalance);
```

Parameters

nWhiteBalance

white balance value (0 : Auto, 1 : Sunny, 2 : Cloudy, 3 : Fluorescent, 4 : Incandescent).

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetWhiteBalance

For C++

Library: CAM.lib

Function : `BOOL CAM_SetWhiteBalance(WHITE_BALANCE nWhiteBalance);`

Example

```
m_Cam.SetWhiteBalance(WB_Auto);
```

4.4.26 GetWhiteBalance

Description

Gets white balance of camera.

Syntax

```
public WHITE_BALANCE GetWhiteBalance();
```

Parameters

None

Return Value

WHITE_BALANCE is enum. (0 : Auto, 1 : Sunny, 2 : Cloudy, 3 : Fluorescent, 4 : Incandescent).

Remarks

None

See Also

SetWhiteBalance

For C++

Library: CAM.lib

Function : `WHITE_BALANCE CAM_GetWhiteBalance();`

Example

```
WHITE_BALANCE wbValue = m_Cam.GetWhiteBalance();
```

4.4.27 SetContrast

Description

Sets contrast of camera.

Syntax

```
public bool SetContrast(int nContrast)
```

Parameters

nContrast

Inputs contrast value of camera.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetContrast

For C++

Library: CAM.lib

Function : BOOL CAM_SetContrast(int nContrast);

Example

```
m_Cam.SetContrast(1);
```

4.4.28 GetContrast

Description

Gets contrast of camera.

Syntax

```
public int GetContrast();
```

Parameters

none

Return Value

Gets contrast of camera.

Remarks

None

See Also

SetContrast

For C++

Library: CAM.lib

Function : int CAM_GetContrast();

Example

```
int nContrast = m_Cam.GetContrast();
```

4.4.29 SetSharpness

Description

Sets sharpness of camera.

Syntax

```
public bool SetSharpness(int nSharpness)
```

Parameters

nSharpness

Inputs sharpness value of camera.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetSharpness

For C++

Library: CAM.lib

Function : BOOL CAM_SetSharpness(int nSharpness);

Example

```
m_Cam.SetSharpness(0);
```

4.4.30 GetSharpness

Description

Gets sharpness of camera.

Syntax

```
public int GetSharpness();
```

Parameters

none

Return Value

Gets sharpness value of camera.

Remarks

None

See Also

SetSharpness

For C++

Library: CAM.lib

Function : int CAM_GetSharpness();

Example

```
Int nSharpness = m_Cam.GetSharpness();
```

4.4.31 SetHistoEqual

Description

Sets whether performing of Histogram Equalization when taking pictures.

Syntax

```
public void SetHistoEqual(bool Enable);
```

Parameters

Enable

Sets whether performing of Histogram Equalization when taking pictures.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetHistoEqual

For C++

Library: CAM.lib

Function : void CAM_SetHistoEqual(BOOL Enable);

Example

```
m_Cam.SetHistoEqual(true);
```

4.4.32 GetHistoEqual

Description

Gets whether performing of Histogram Equalization when taking pictures.

Syntax

```
public int GetHistoEqual();
```

Parameters

None

Return Value

Gets whether performing of Histogram Equalization when taking pictures.

Remarks

None

See Also

SetHistoEqual

For C++

Library: CAM.lib

Function : BOOL CAM_GetHistoEqual();

Example

None

4.4.33 RegisterPreview

Description

Registers Callback function to get the preview screen.

Syntax

```
public bool RegisterPreview(Cam.SampleProcessedProc fpPreview)
```

Parameters

fpPreview

CallBack Procedure formed a SampleProcessedProc function.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library: CAM.lib

Function : BOOL CAM_RegisterPreview(MANAGED_SAMPLEPROCESSEDPROC fpPreview);

Example

None

4.4.34 InsertExifInformation

Description

Insert EXIF information to Jpeg image.

Syntax

```
public bool InsertExifInformation(string FileName, Cam.ExifInfo e);
```

Parameters

FileName

Jpeg file name which includes the EXIF Information.

Info

ExifInfo Structure.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

UseGPSExifData

For C++

Library: CAM.lib

Function : BOOL CAM_InsertExifInformation(TCHAR* FileName, ExifInfo info);

Example

```
ExifInfo info = new ExifInfo();  
info.TitleName = "Picture0.jpg";  
info.Make = "M3 Mobile Co.Ltd";  
info.Model = "M3 SMART";  
m_Cam.InsertExifInformation("Flash Disk\\Camera\\Photo\\Picture0.jpg", info);
```

4.4.35 UseGPSExifData

Description

Insert GPS information in EXIF Information.

Syntax

```
public bool UseGPSExifData(bool bUse);
```

Parameters

bUse

Whether the GPS information is included in EXIF Information or not.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

Once GPS function is set to use, GPS satellite should be visualized. GPS information starts to be inserted in EXIF information if GPS signal can be received from satellite.

See Also

InsertExifInformation

For C++

Library: CAM.lib

Function : `BOOL CAM_UseGPSExifData(BOOL bUse);`

Example

```
m_Cam.UseGPSExifData(true);
```

i. GetMo

GetRegExifGpsEnable

Description

Gets model type of device.

Syntax

```
public bool GetRegExifGpsEnable();
```

Parameters

None

Return Value

Enable if true, disable if false.

Remarks

None

See Also

None.

For C++

Library: CAM.lib

Function : `BOOL CAM_GetRegExifEnable();`

Example

```
bool =m_Cam.GetRegExufEnable();
```

ii. delType

4.4.36 GetModelType

Description

Gets model type of device.

Syntax

```
public MODEL_TYPE GetModelType();
```

Parameters

None

Return Value

MODEL_TYPE is Enum Type value. Obtains model type of device.

Remarks

None

See Also

GetScannerType

For C++

Library: CAM.lib

Function : MODEL_TYPE CAM_GetModelType();

Example

```
MODEL_TYPE type =m_Cam.GetModelType();
```

4.4.37 GetScannerType

Description

Gets Scanner Type of device.

Syntax

```
public SCANNER_TYPE GetScannerType();
```

Parameters

None

Return Value

SCANNER_TYPE is Enum type value. Obtains scanner type of device.

Remarks

None

See Also

GetModelType

For C++

Library: CAM.lib

Function : SCANNER_TYPE CAM_GetScannerType();

Example

```
SCANNER_TYPE type = m_Cam.GetScannerType();
```

4.4.38 GetVersion

Description

Gets dll version.

Syntax

```
public bool GetVersion(StringBuilder strDllVersion, StringBuilder strDllRelease);
```

Parameters

strDllVersion

Obtains DLL version.

strDllRelease

Obtains released date of DLL.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library: CAM.lib

Function : BOOL CAM_GetVersion()

Example

None

4.5 RFID (LF/HF)

Status Return value

Please refer to the below table for the status value definition.

Status Value Definition	Code	Meaning
TRUE	1	Success
FALSE	0	General Error

Enum
<pre>public enum READ_MODE { READ_ASYNC = 0, READ_SYNC = 1, READ_CONTINUOUS = 2, } public enum RESULT_MODE { MODE_HEX = 0, MODE_DEC = 1, MODE_CHAR = 2, } public enum RFID_TYPE { RFID_LF = 0, RFID_HF = 1, RFID_NOTHING = 2, } public enum SOUND_MODE { SOUND_NO = 0, SOUND_1 = 1, SOUND_2 = 2, SOUND_3 = 3, VIB_1 = 4, VIB_2 = 5, } public enum STATE_SOUND { STATE_MSG = 0, STATE_READ = 1, STATE_WRITE = 2, } public enum TAG_TYPE_HF</pre>

```
{  
    ISO_14443_TYPE_A = 0,  
    ISO_14443_TYPE_B = 1,  
    ICODE_UID = 2,  
    ICODE_EPC = 3,  
    ICODE = 4,  
    SR176 = 5,  
    ACTIVATE_ALL_TAG = 6,  
    ISO_15693 = 7,  
}  
public enum TAG_TYPE_LF  
{  
    ACTIVATE_ALL_TAGS = 0,  
    EM4x02 = 1,  
    EM4x05 = 2,  
    EM4x50 = 3,  
    HITAG1_S = 4,  
    HITAG2 = 5,  
    TI_RFID_SYSTEMS = 6,  
}
```

Functions for RFID

Name	Description
Open	Opens RFID.
Close	Closes RFID
PowerSupply	Supplies power to RFID module.
GetRadioType	Gets RFID Radio Type of device.
SelectTag	Searches and Selects one tag within Antenna field.
HighSelectTag	Searches one tag within Antenna field and selects as High baudrate.
LoginTag	Login the tag if necessary.
ReadBlock	Reads memory block of the tag.
WriteBlock	Writes data in memory block of the tag.
ReadMultiBlock	Reads multiple data blocks on a card.
WriteMultiBlock	Writes multiple data blocks on a card.
SetAntenna	Controls antenna power of the RFID Module.
GetVersion	Gets DLL information and FW information of reader.
EnableMultiTag	Enables Anti-Collision function so that multi tags can be read.
SendReadMultiTag	Sends commander reading multi tags to reader.
SendTransferCommand	Allows card-specific communication.
SendContinuousRead	Reads and displays serial numbers continuously while one or more tags remain in the field.
SendCommand	Sends a command to a reader specified by its handle
SendCommandGetData	Sends a command to a reader specified by its handle and receives data.
GetData	Gets the result that performs commander stored in reader.
CheckResult	Checks obtained result from reader if it is valid.
SoundPlay	Plays sound and vibration.
SetTagType	Sets up the reader for a specific tag type.
GetTagType	Gets tag type.
GetTagTypeToString	Gets tag type as string.
TagItInventory	Performs Inventory commander of TagIt tag.
TagItReadBlock	Reads memory block of TagIt tag.
TagItWriteBlock	Writes memory block of TagIt tag.

4.5.1 Open

Description

Opens RFID.

Syntax

```
public RFID_TYPE Open();
```

Parameters

None

Return Value

RFID_TYPE is enum value. Can be identified whether it is LF reader or HF reader.

Remarks

Instead of not supporting the Close function, PowerSupply function cuts the power then close the application.

See Also

PowerSupply

For C++

Library: RFID.lib

Function : RFID_TYPE RFID_Open();

Example

```
using RFIDNet;

RFIDCommon RfidCommon = new RFIDCommon();

RFID_TYPE type = RfidCommon.Open();

if(type == RFID_TYPE .RFID_LF)
{
    // LF RFID
}

else if(type == RFID_TYPE.RFID_HF)
{
    // HF RFID
}

else
{
    return;
}
```

4.5.2 Close

Description

Closes RFID.

Syntax

```
public bool Close();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

RFID_Close is supported in CFReader.dll version 4.1.5.7. If earlier version is used, power off via RFID_PowerSupply then close the program in M3 SKY. In M3 ORANGE, closing the program will close RFID.

See Also

PowerSupply

For C++

Library: RFID.lib

Function : BOOL RFID_Close()

Example

```
RfidCommon.Close();
```

4.5.3 PowerSupply

Description

Supplies power to RFID module.

Syntax

```
public bool PowerSupply(bool bOn);
```

Parameters

bOn

Supplies power to reader.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

This function is not necessary in M3 ORANGE.

See Also

None

For C++

Library: RFID.lib

Function : BOOL RFID_PowerSupply(BOOL bOn);

Example

```
RfidCommon.PowerSupply(false);
```

4.5.4 GetRadioType

Description

Gets RFID Radio Type of device.

Syntax

```
public RFID_TYPE GetRadioType ();
```

Parameters

None

Return Value

RFID_TYPE is enum value. Can be identified whether it is LF reader or HF reader.

Remarks

None

See Also

None

For C++

Library: RFID.lib

Function : RFID_TYPE RFID_GetType();

Example

```
using RFIDNet;

RFIDCommon RfidCommon = new RFIDCommon();

RFID_TYPE type = RfidCommon.GetType();

if(type == RFID_TYPE .RFID_LF)
{
    // LF RFID
}

else if(type == RFID_TYPE.RFID_HF)
{
    // HF RFID
}
```



```
else
{
    return;
}
```

4.5.5 SelectTag

Description

Searches and Selects one tag within Antenna field.

Syntax

```
public bool SelectTag(StringBuilder strSerial);
```

Parameters

strSerial

Obtains serial number of selected tag.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

HighSelectTag, ReadBlock, WriteBlock

For C++

Library: RFID.lib

Function : BOOL RFID_SelectTag(LPWSTR strSerial);

Example

```
StringBuilder strSerial = new StringBuilder(260);
RfidCommon.SelectTag(strSerial);
```

4.5.6 HighSelectTag

Description

Searches one tag within Antenna field and selects as High baudrate.

Syntax

```
public bool HighSelectTag(StringBuilder strSerial);
```

Parameters

strSerial

Obtains serial number of selected tag.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

This function can be used when the tag allows High Baudrate communication.

See Also

SelectTag, ReadBlock, WriteBlock

For C++

Library: RFID.lib

Function : BOOL RFID_HighSelectTag(LPWSTR strSerial);

Example

```
stringBuilder strSerial = new StringBuilder(260);
```

```
RfidCommon.HighSelectTag(strSerial);
```

4.5.7 LoginTag

Description

Login the tag if necessary.

Syntax

```
public char LoginTag(string strLogin);
```

Parameters

strLogin

Inputs password to login.

Return Value

Obtains result of login as Wide Character. 'L' means success.

Remarks

None

See Also

ReadBlock, WriteBlock

For C++

Library: RFID.lib

Function : TCHAR RFID_LoginTag(LPCWSTR strLogin);

Example

```
stringBuilder strSerial = new StringBuilder(260);
```

```
stringBuilder strData = new StringBuilder(260);
```

```
if(RfidCommon.SelectTag(strSerial))
```

```

{
    char cRet = RfidCommon.LoginTag("01AFFFFFFFFFFFFF");
    if(cRet == 'L' || cRet == 'I')
    {
        RfidCommon.ReadBlock("01",strData);
    }
}

```

4.5.8 ReadBlock

Description

Reads memory block of the tag.

Syntax

```
public bool ReadBlock(string strBlock, StringBuilder strData);
```

Parameters

strBlock

Inputs Block Number to read.

strData

Obtains read Block Data.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

WriteBlock

For C++

Library: RFID.lib

Function : BOOL RFID_ReadBlock(LPCWSTR strBlock, LPWSTR strData)

Example

```

StringBuilder strSerial = new StringBuilder();
RfidCommon.SelectTag(strSerial);
const int nMaxData = 1024;
StringBuilder strData = new StringBuilder(nMaxData);
RfidCommon.ReadBlock("01", strData);
if (RfidCommon.CheckResult(strData.ToString(), strOutData))

```

```
{  
    // Success  
}
```

4.5.9 WriteBlock

Description

Writes data in memory block of the tag.

Syntax

```
public bool WriteBlock(string strBlock, string strInData, StringBuilder strOutData);
```

Parameters

strBlock

Inputs Block Number to write.

strInData

Inputs Block Data to write.

strOutData

Obtains written result.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

ReadBlock, WriteMultiBlock

For C++

Library: RFID.lib

Function : BOOL RFID_WriteBlock(LPCWSTR strBlock, LPCWSTR strInData, LPWSTR strOutData)

Example

```
StringBuilder strSerial = new StringBuilder();  
RfidCommon.SelectTag(strSerial);  
const int nMaxData = 1024;  
StringBuilder strOutData = new StringBuilder(nMaxData);  
m_Rfid.WriteBlock("01", "00112233", strOutData);
```

4.5.10 ReadMultiBlock

Description

Reads multiple data blocks on a card.

Syntax

```
public void ReadMultiBlock(string strStartBlock, string strNumBlock, StringBuilder strOutData);
```

Parameters

strStartBlock

Inputs Start Block Number to read.

strNumBlock

Inputs number of Memory Block to read.

strData

Obtains Block Data to read.

Return Value

None

Remarks

This function can be used upper RFID version of 1.22.

See Also

WriteBlock , WriteMultiBlock

For C++

Library: RFID.lib

Function : void RFID_ReadMultiBlock(LPCWSTR strStartBlock, LPCWSTR strNumBlock, LPWSTR strData)

Example

```
StringBuilder strSerial = new StringBuilder();  
RfidCommon.SelectTag(strSerial);  
const int nMaxData = 1024;  
StringBuilder strData = new StringBuilder(nMaxData);  
RfidCommon.ReadMultiBlock("01", "03", strData);  
if (RfidCommon.CheckResult(strData.ToString(), strOutData))  
{  
    // Success  
}
```

4.5.11 WriteMultiBlock

Description

Writes multiple data blocks on a card.

Syntax

```
public void WriteMultiBlock(string strStartBlock, string strNumBlock, string strWriteData, StringBuilder strOutData);
```

Parameters

strStartBlock

Inputs Start Block Number to write.

strNumBlock

Inputs number of Memory Block to write.

strWriteData

Inputs Block Data to write.

strOut

Obtains written result.

Return Value

void

Remarks

This function can be used upper RFID version of 1.22.

See Also

ReadBlock, ReadMultiBlock

For C++

Library: RFID.lib

Function : void RFID_WriteMultiBlock(LPCWSTR strStartBlock, LPCWSTR strNumBlock, LPCWSTR strWriteData, LPWSTR strOut);

Example

```
StringBuilder strSerial = new StringBuilder();  
RfidCommon.SelectTag(strSerial);  
const int nMaxData = 1024;  
StringBuilder strOutData = new StringBuilder(nMaxData);  
m_Rfid.WriteMultiBlock("01", "02", "0011223344556677", strOutData);
```

4.5.12 SetAntenna

Description

Controls antenna power of the RFID Module.

Syntax

```
public void SetAntenna(bool bOn);
```

Parameters

bOn

Sets the condition of Antenna.

Return Value

None

Remarks

Main battery can be saved by turning off the antenna.

See Also

None

For C++

Library: RFID.lib

Function : void RFID_SetAntenna(BOOL bSet);

Example

```
if(bAntennaOn == TRUE)
{
    Rfid_SetAntenna(TRUE);
}
else
{
    Rfid_SetAntenna(FALSE);
}
```

4.5.13 GetVersion

Description

Gets DLL information and FW information of reader.

Syntax

```
public bool GetVersion(StringBuilder strFwVersion, StringBuilder strReaderDllVersion, StringBuilder strRfidDllVersion);
```

Parameters

strFwVersion

Obtains FW version of reader.

strReaderDllVersion

Obtains version of CFReader.dll.

strRfidDllVersion

Obtains version of RFID.dll.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library: RFID.lib

Function : BOOL RFID_GetVersion(LPWSTR szFwVersion, LPWSTR szReaderDllVersion, LPWSTR szRfidDllVersion);

Example

```
StringBuilder strFwVersion = new StringBuilder(260);  
StringBuilder strReaderDllVersion = new StringBuilder(260);  
StringBuilder strDllVersion = new StringBuilder(260);  
StringBuilder strTagType = new StringBuilder(260);  
RfidCommon.GetVersion(strFwVersion, strReaderDllVersion, strDllVersion);
```

4.5.14 EnableMultiTag

Description

Enables Anti-Collision function so that multi tags can be read.

Syntax

```
public bool EnableMultiTag(bool bEnable);
```

Parameters

bEnable

Enables Anti-Collision function of reader.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

SendReadMultiTag , GetData

For C++

Library: RFID.lib

Function : BOOL RFID_EnableMultiTag(BOOL bEnable);

Example

```
RfidCommon.EnableMultiTag(true);
```


4.5.15 SendReadMultiTag

Description

Sends commander reading multi tags to reader.

Syntax

```
public void SendReadMultiTag();
```

Parameters

None

Return Value

None

Remarks

MultiTag can be read with SendContinuousRead function after EnableMultiTag.

See Also

EnableMultiTag, GetData, SendContinuousRead

For C++

Library: RFID.lib

Function : void RFID_SendReadMultiTag();

Example

None

4.5.16 SendTransferCommand

Description

Allows card-specific communication.

Syntax

```
public void SendTransferCommand(string strData)
```

Parameters

strData

Normal mode

Downlink length (1 byte) <> 0 Option byte (1 byte) Data (n bytes)

Transmit/Receive mode

Downlink length (1 byte) = 0 Downlink length new (1 byte) Option byte (1 byte) Transmit byte (1 byte) Receive byte (1 byte) CRC Preset LSB (1 byte) CRC Preset MSB (1 byte) Data (n bytes).

Return Value

None

Remarks

None

See Also

GetData

For C++

Library: RFID.lib

Function : void RFID_SendTransferCommand(LPWSTR strData)

Example

None

4.5.17 SendContinuousRead

Description

Reads and displays serial numbers continuously while one or more tags remain in the field.

Syntax

```
public bool SendContinuousRead(bool bStart);
```

Parameters

bStart

Sets whether starting Continuous read or not.

Return Value

None

Remarks

MultiTag can be read when EnableMultiTag is enabled.

See Also

GetData , EnableMultiTag

For C++

Library: RFID.lib

Function : void RFID_SendContinuousRead(BOOL bStart);

Example

```
RfidCommon.SendContinuousRead(true);  
  
bool bRead = true;  
while(bRead)  
{  
    StringBuilder strSerial = new StringBuilder(260);  
    RfidCommon.GetData(strSerial);  
}  
  
RfidCommon.SendContinuousRead(false);
```

4.5.18 SendCommand

Description

The RFID_SendCommand function sends a command to a reader specified by its handle

Syntax

```
public void SendCommand(string strCommand, string strData);
```

Parameters

strCommand

Zero terminated string with the command

strData

Zero terminated string containing the data

Return Value

None.

Remarks

None

See Also

GetData

For C++

Library: RFID.lib

Function : void SendCommand(LPCWSTR strCommand, LPCWSTR strData)

Example

```
RfidCommon.SendCommand("s", "");  
  
bool bRead = true;  
while(bRead)  
{  
    StringBuilder strSerial = new StringBuilder(260);  
    RfidCommon.GetData(strSerial);  
}
```

4.5.19 SendCommandGetData

Description

The RFID_SendCommandGetData function sends a command to a reader specified by its handle and receives data.

Syntax

```
public void SendCommandGetData(string strCommand, string strInputData, StringBuilder strOutputData);
```

Parameters

strCommand

Zero terminated string with the command

strInputData

Zero terminated string containing the data

strOutputData

Gets the result that performs commander stored in reader

Return Value

None.

Remarks

None

See Also

SendCommand, GetData

For C++

Library: RFID.lib

Function : void RFID_SendCommandGetData(LPCWSTR strCommand, LPCWSTR strInputData, LPWSTR strOutputData)

Example

```
StringBuilder strSerial = new StringBuilder(260);
```

```
RfidCommon.SendCommandGetData("s", "", strSerial);
```

4.5.20 GetData

Description

Gets the result that performs commander stored in reader.

Syntax

```
public bool GetData(StringBuilder strData);
```

Parameters

strData

Result stored in reader can be obtained.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

This function can obtain the result from commanders from Send functions.

See Also

SendReadMultiTag, SendTransferCommand, SendContinuousRead

For C++

Library: RFID.lib

Function : BOOL RFID_GetData(LPWSTR strData)

Example

```
RfidCommon.SendContinuousRead(true);  
  
bool bRead = true;  
while(bRead)  
{  
    StringBuilder strSerial = new StringBuilder(260);  
    RfidCommon.GetData(strSerial);  
}  
  
RfidCommon.SendContinuousRead(false);
```

4.5.21 CheckResult

Description

Checks obtained result from reader if it is valid.

Syntax

```
public bool CheckResult(string strInputResult, StringBuilder strOutputResult);
```

Parameters

strInputResult

Inputs result obtained from reader.

strOutputResult

Obtains message if it is valid result from reader.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetData

For C++

Library: RFID.lib

Function : BOOL RFID_CheckResult(TCHAR* tzInputResult, TCHAR* tzOutputResult)

Example

```
StringBuilder strSerial = new StringBuilder();
RfidCommon.SelectTag(strSerial);
const int nMaxData = 1024;
StringBuilder strData = new StringBuilder(nMaxData);
StringBuilder strOutData = new StringBuilder(nMaxData);
RfidCommon.ReadBlock("01", strData);
RfidCommon.ReadMultiBlock("01", "03", strData);
if (RfidCommon.CheckResult(strData.ToString(), strOutData))
{
    // Success
}
```

4.5.22 SoundPlay

Description

Plays sound and vibration.

Syntax

```
public bool SoundPlay(SOUND_MODE sound);
```

Parameters

Sound

SOUND_MODE is enum value type and displays sound and vibration.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library: RFID.lib

Function : BOOL RFID_SoundPlay(SOUND_MODE Sound)

Example

```
RfidCommon.SoundPlay(SOUND_MODE .SOUND_1);.
```

4.5.23 SetTagType

Description

Sets up the reader for a specific tag type.

Syntax

```
public int SetTagType(int Type)
```

Parameters

nType

TAG_TYPE_HF is applied to HF RFID and TAG_TYPE_LF is applied to LF RFID.

Return Value

Current Tag Type is returned.

Remarks

None

See Also

GetTagType, GetTagTypeToString

For C++

Library: RFID.lib

Function : nt RFID_SetTagType(int nType)

Example

```
RfidCommon.SetTagType(6);
```

4.5.24 GetTagType

Description

Gets tag type.

Syntax

```
public int GetTagType();
```

Parameters

None

Return Value

TAG_TYPE_HF is applied with HF RFID and TAG_TYPE_LF is applied with LF RFID.

Remarks

None

See Also

GetTagTypeToString, SetTagType

For C++

Library: RFID.lib

Function : int RFID_GetTagType();

Example

```
int nType = RfidCommon.GetTagType();.
```

4.5.25 GetTagTypeToString

Description

Gets tag type as string.

Syntax

```
public void GetTagTypeToString(StringBuilder strTagType);
```

Parameters

strTagType

Obtains current TagType as string.

Return Value

None

Remarks

None

See Also

SetTagType, GetTagType

For C++

Library: RFID.lib

Function : void RFID_GetTagTypeToString(TCHAR* tzTagType);

Example

```
StringBuilder strType = new StringBuilder(260);
```

```
RfidCommon.GetTagTypeToString(strType);.
```

4.5.26 TagInventory

Description

Performs Inventory commander of TagInventory tag.

Syntax

```
RFID_API BOOL RFID_TagInventory(TCHAR* tzUID)
```

Parameters

tzUID

Obtains UID of tag.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

TagItReadBlock, TagItWriteBlock

For C++

Library: RFID.lib

Function : BOOL RFID_TagItInventory(TCHAR* tzUID)

Example

```
StringBuilder strSerial = new StringBuilder(512);  
StringBuilder strBlock = new StringBuilder(512);  
m_TagIt.Inventory(strSerial);  
m_TagIt.ReadBlock(1, strBlock);
```

4.5.27 TagItReadBlock

Description

Reads memory block of TagIt tag.

Syntax

```
public bool ReadBlock(int nBlockNo, StringBuilder strBlockData);
```

Parameters

nBlockNo

Inputs Block Number to read.

tzBlockData

Obtains Block Data to read.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

TagItInventory, TagItWriteBlock

For C++

Library: RFID.lib

Function : BOOL RFID_TagItReadBlock(int nBlockNo, TCHAR* tzBlockData);

Example

```
StringBuilder strSerial = new StringBuilder(512);  
StringBuilder strBlock = new StringBuilder(512);  
m_TagIt.Inventory(strSerial);  
m_TagIt.ReadBlock(1, strBlock);
```

4.5.28 TagItWriteBlock

Description

Writes memory block of TagIt tag.

Syntax

```
public bool WriteBlock(int nBlockNo, string strWriteBlockData, StringBuilder strWriteResult);
```

Parameters

nBlockNo

Inputs Block Number to write.

tzWriteBlockData

Inputs Block Data to write.

tzWriteResult

Obtains written result.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

TagItInventory, TagItReadBlock

For C++

Library: RFID.lib

Function : BOOL RFID_TagItWriteBlock(int nBlockNo, CONST TCHAR* tzWriteBlockData, TCHAR* tzWriteResult)

Example

```
StringBuilder strSerial = new StringBuilder(512);  
StringBuilder strWriteResult = new StringBuilder(512);  
m_TagIt.Inventory(strSerial);  
m_TagIt.WriteBlock(1, "00112233" strWriteResult);
```

4.6 UHF GUN READER

Status Return value

Please refer to the below table for the status value definition.

Status Value Definition	Code	Meaning
TRUE	1	Success
FALSE	0	General Error

Event
<pre>public delegate void ReceivedInventory(); public delegate void ReceivedPower(bool a_bPowerOn); public delegate void ReceivedMemoryData();</pre>
Enum
<pre>public enum BATTERY_STATUS { BATTERY_ERROR = 0, BATTERY_0 = 1, BATTERY_1 = 2, BATTERY_2 = 3, BATTERY_3 = 4, BATTERY_FULL = 5, } public enum RFIDErrorcode { TAG_OTHERERROR = 0x0, TAG_SUCCESS = 0x1, TAG_MEMORYOVERRUN = 0x3, TAG_MEMORYLOCKED = 0x4, TAG_INSUFFICIENTPOWER = 0xB, TAG_NONSPECIFICERROR = 0xF, MAC_NOERROR = 0x10, MAC_HANDLEMISSMATCH = 0x11, MAC_CRCERROR = 0x12, MAC_NOTAGREPLY = 0x13, MAC_INVALIDPASSWD = 0x14, MAC_ZEROKILLPASSWD = 0x15, MAC_TAGLOST = 0x16, MAC_COMMANDFORMATERROR = 0x17, MAC_READCOUNTINVALID = 0x18, MAC_OUTOFRETRIES = 0x19 };</pre>

```
public enum RFIDTagBank
```

```
{  
    TAG_RESERVED = 0,  
    TAG_EPC = 1,  
    TAG_TID = 2,  
    TAG_USER = 3  
};
```

```
public enum RFIDRegion
```

```
{  
    RFID_REGION_KOREA_NEW = 0,  
    RFID_REGION_KOREA_WEAK = 1,  
    RFID_REGION_KOREA_OLD = 2,  
    RFID_REGION_USA = 3,  
    RFID_REGION_EURO = 4,  
    RFID_REGION_EURO_NEW = 5,  
    RFID_REGION_JAPAN = 6,  
    RFID_REGION_CHINA = 7,  
    RFID_REGION_AUSTRALIA = 8,  
    RFID_REGION_BRAZIL = 9,  
    RFID_REGION_MALAYSIA = 10,  
    RFID_REGION_TAIWAN = 11  
};
```

```
public enum RFIDLockPermission
```

```
{  
    PERMISSION_ACCESSIBLE = 0,  
    PERMISSION_ALWAYS_ACCESSIBLE = 1,  
    PERMISSION_SECURED_ACCESSIBLE = 2,  
    PERMISSION_ALWAYS_NOT_ACCESSIBLE = 3,  
    PERMISSION_NO_CHANGE = 4  
};
```

```
public enum RFID_STATUS
```

```
{  
    /* Success */  
    RFID_STATUS_OK,  
    /* Attempted to open a radio that is already open */  
    RFID_ERROR_ALREADY_OPEN = -9999, /* -9999 */  
    /* Buffer supplied is too small */  
    RFID_ERROR_BUFFER_TOO_SMALL, /* -9998 */  
    /* General failure */  
    RFID_ERROR_FAILURE, /* -9997 */  
    /* Failed to load radio bus driver */  
    RFID_ERROR_DRIVER_LOAD, /* -9996 */  
    /* Library cannot use version of radio bus driver present on system */
```

```

RFID_ERROR_DRIVER_MISMATCH,                /* -9995 */
/* This error code is no longer used, maintain slot in enum in case
 * anyone is using hard-coded error codes for some reason */
RFID_ERROR_RESERVED_01,                    /* -9994 */
/* Antenna number is invalid */
RFID_ERROR_INVALID_ANTENNA,                /* -9993 */
/* Radio handle provided is invalid */
RFID_ERROR_INVALID_HANDLE,                /* -9992 */
/* One of the parameters to the function is invalid */
RFID_ERROR_INVALID_PARAMETER,              /* -9991 */
/* Attempted to open a non-existent radio */
RFID_ERROR_NO_SUCH_RADIO,                  /* -9990 */
/* Library has not been successfully initialized */
RFID_ERROR_NOT_INITIALIZED,                /* -9989 */
/* Function not supported */
RFID_ERROR_NOT_SUPPORTED,                  /* -9988 */
/* Operation was cancelled by call to cancel operation, close radio, or
 * shut down the library */
RFID_ERROR_OPERATION_CANCELLED,            /* -9987 */
/* Library encountered an error allocating memory */
RFID_ERROR_OUT_OF_MEMORY,                  /* -9986 */
/* The operation cannot be performed because the radio is currently busy */
RFID_ERROR_RADIO_BUSY,                     /* -9985 */
/* The underlying radio module encountered an error */
RFID_ERROR_RADIO_FAILURE,                  /* -9984 */
/* The radio has been detached from the system */
RFID_ERROR_RADIO_NOT_PRESENT,              /* -9983 */
/* The RFID library function is not allowed at this time. */
RFID_ERROR_CURRENTLY_NOT_ALLOWED,          /* -9982 */
/* The radio module's MAC firmware is not responding to requests. */
RFID_ERROR_RADIO_NOT_RESPONDING,          /* -9981 */
/* The MAC firmware encountered an error while initiating the nonvolatile
 * memory update. The MAC firmware will return to its normal idle state
 * without resetting the radio module. */
RFID_ERROR_NONVOLATILE_INIT_FAILED,        /* -9980 */
/* An attempt was made to write data to an address that is not in the
 * valid range of radio module nonvolatile memory addresses. */
RFID_ERROR_NONVOLATILE_OUT_OF_BOUNDS,      /* -9979 */
/* The MAC firmware encountered an error while trying to write to the
 * radio module's nonvolatile memory region. */
RFID_ERROR_NONVOLATILE_WRITE_FAILED,       /* -9978 */
/* The underlying transport layer detected that there was an overflow
 * error resulting in one or more bytes of the incoming data being
 * dropped. The operation was aborted and all data in the pipeline was
 * flushed. */
RFID_ERROR_RECEIVE_OVERFLOW,               /* -9977 */

```

```

/* An unexpected value was returned to this function by the MAC firmware */
RFID_ERROR_UNEXPECTED_VALUE,                /* -9976 */
/* The MAC firmware encountered CRC errors while trying to          */
/* write to the radio module's nonvolatile memory region.            */
RFID_ERROR_NONVOLATILE_CRC_FAILED,            /* -9975 */
/* The MAC firmware encountered unexpected values in the packet header */
RFID_ERROR_NONVOLATILE_PACKET_HEADER,         /* -9974 */
/* The MAC firmware received more than the specified maximum packet size */
RFID_ERROR_NONVOLATILE_MAX_PACKET_LENGTH      /* -9973 */
};

```

Structure

```

public struct RFID_VERSION
{
    /* The major version (i.e., in 1.x.x.x, the 1) */
    public INT32U major;
    /* The minor version (i.e., in x.1.x.x, the 1) */
    public INT32U minor;
    /* The maintenance number (i.e., in x.x.1.x, the 1) */
    public INT32U maintenance;
    /* The release number (i.e., in x.x.x.1, the 1) */
    public INT32U release;
};

public struct RFIDReadCmd
{
    internal HWND hWnd; //Diag handle Receive for Message
    public RFIDTagBank bank;
    public INT8U wlength;
    public INT8U offset;
    public INT32U accpwd;
};

public struct RFIDWriteCmd
{
    internal HWND hWnd; //Diag handle Receive for Message
    public RFIDTagBank bank;
    public INT8U wlength;
    public INT8U offset;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 32)]
    public INT16U[] wdata;
    public INT32U accpwd;
};

public struct RFIDLockCmd

```

```
{  
    internal HWND hWnd; //Diag handle Receive for Message  
    public RFIDLockPermission lockkillpwd;  
    public RFIDLockPermission lockaccesspwd;  
    public RFIDLockPermission lockepc;  
    public RFIDLockPermission locktid;  
    public RFIDLockPermission lockuser;  
    public INT32U accpwd;  
};  
  
public struct RFIDKillCmd  
{  
    internal HWND hWnd; //Diag handle Receive for Message  
    public INT32U killpwd;  
};
```

Functions for UHF GUN READER

Name	Description
GetError	Check the last error
IsReady	Check UHF GUN READER's availability by checking the power
Init	Initialize RFID
Close	Close RFID
Inventory	Start Inventory by reading EPC data
InventoryStop	Stop Inventory
Read	Read memory from tag
Write	Write data to tag
Lock	Limit access to tag
Kill	Kill the tag (disabling the tag)
GetData	Get tag data from Inventory and Read
SetRegionFrequency	Set RFID frequency to suit regional regulation
GetRegionFrequency	Check current regional frequency setting
ReadBattery	Check battery voltage and ADC value
ReadBatteryStatus	Check battery level (0~4 level)
Version	Check DLL and F/W version

4.6.1 GetError

Description

Check the last error

Syntax

```
RFIDUHF.RFIDErrorCode GetError();
```

Parameters

None

Return Value

Retuens RFIDErrorCode of enum type

Remarks

None

See Also

None

For C++

Library : RFID_UHF.lib

Function : RFIDErrorCode UHF_GetError();

Example

```
int nLength = 0;
StringBuilder strData = new StringBuilder(260);
RFIDUHF.RFIDErrorCode error;
nLength = _UHFNet.GetData(strData);
if (nLength == 0)
{
    error = _UHFNet.GetError();
    MessageBox.Show("OnReceivedData: " + error.ToString());
}
```

4.6.2 IsReady

Description

Check UHF GUN READER's availability by checking the power

Syntax

```
bool IsReady();
```

Parameters

None

Return Value

TRUE = Success

FALSE = Fail

Remarks

Following cases will return false:

1. PDA detached from gun reader
2. RFID / SCAN switch is at SCAN position
3. Gun reader's battery is detached or empty
4. PDA in gun reader is synced with PC

See Also

None

For C++

Library : RFID_UHF.lib

Function : BOOL UHF_IsReady();

Example

```
if(!_UHFNet.IsReady())  
    return;
```

4.6.3 Init

Description

Initialize RFID

Syntax

RFIDUHF.RFID_STATUS Init();

Parameters

None

Return Value

Returns RFID_STATUS of enum type

Remarks

None

See Also

Close

For C++

Library : RFID_UHF.lib

Function : RFID_STATUS UHF_Init (HWND hDlgWnd);

Example

```
RFIDUHF.RFID_STATUS status = new RFIDUHF.RFID_STATUS();  
String strstatus = null;  
status = UHFNet.Init();  
if (status != RFIDUHF.RFID_STATUS.RFID_STATUS_OK)  
{  
    strstatus = String.Format("RFID Init Error-{0:G}", status);  
}
```

4.6.4 Close

Description

Close RFID

Syntax

```
RFIDUHF.RFID_STATUS Close();
```

Parameters

None

Return Value

Returns RFID_STATUS of enum type

Remarks

None

See Also

Init

For C++

Library : RFID_UHF.lib

Function : RFID_STATUS UHF_Close();

Example

```
RFIDUHF.RFID_STATUS status = new RFIDUHF.RFID_STATUS();  
String strstatus = null;  
status = UHFNet.Close();  
if (status != RFIDUHF.RFID_STATUS.RFID_STATUS_OK)  
{  
    strstatus = String.Format("{0:G}", status);  
}
```

4.6.5 Inventory

Description

Start Inventory by reading EPC data

Syntax

```
void Inventory();
```

Parameters

None

Return Value

None

Remarks

When tag is detected after starting inventory, an event is logged through delegate void ReceivedInventory()

See Also

InventoryStop

For C++

Library : RFID_UHF.lib

Function : void UHF_Inventory(HWND hWnd);

Example

```
UHFNet.Inventory();
```

4.6.6 InventoryStop

Description

Stop Inventory

Syntax

```
void InventoryStop();
```

Parameters

None

Return Value

None

Remarks

None

See Also

Inventory

For C++

Library : RFID_UHF.lib

Function : void UHF_InventoryStop();

Example

```
UHFNet.InventoryStop();
```

4.6.7 Read

Description

Read memory from tag

Syntax

```
RFIDUHF.RFID_STATUS Read(ref RFIDUHF.RFIDReadCmd cmd);
```

Parameters

cmd

RFIDReadCmd structure is used to assign tag memory location

Return Value

Returns RFID_STATUS of enum type

Remarks

When tag is detected using read, an event is logged through delegate void ReceivedMemoryData(). Then, the data can be obtained using GetData.

See Also

GetData

For C++

Library : RFID_UHF.lib

Function : RFID_STATUS UHF_Read(RFIDReadCmd *cmd);

Example

```
RFIDUHF.RFID_STATUS status = new RFIDUHF.RFID_STATUS();
```

```
RFIDUHF.RFIDReadCmd ReadCmd = new RFIDUHF.RFIDReadCmd();
```

```
ReadCmd.bank = RFIDUHF.RFIDTagBank.TAG_EPC;
```

```
ReadCmd.wlength = 6;
```

```
ReadCmd.offset = 2;
```

```
ReadCmd.accpwd = Convert.ToUInt16(textBox_RWPwd.Text, 16); // 00000000
```

```
status = _UHFNet.Read(ref ReadCmd);
```

```
if (status != RFIDUHF.RFID_STATUS.RFID_STATUS_OK)
```

```
{
    MessageBox.Show("RFID Read Fail!");
}
```

4.6.8 Write

Description

Write data to tag

Syntax

```
RFIDUHF.RFID_STATUS Write(ref RFIDUHF.RFIDWriteCmd cmd);
```

Parameters

cmd

RFIDWriteCmd structure is used to assign tag memory location

Return Value

Returns RFID_STATUS of enum type

Remarks

An event is logged through delegate void ReceivedMemoryData() after write command. Then, the writing result can be obtained using GetError.

See Also

GetError

For C++

Library : RFID_UHF.lib

Function : RFID_STATUS UHF_Write(RFIDWriteCmd *cmd);

Example

```
RFIDUHF.RFID_STATUS status = new RFIDUHF.RFID_STATUS();
```

```
RFIDUHF.RFIDWriteCmd WriteCmd = new RFIDUHF.RFIDWriteCmd();
```

```
WriteCmd.accpwd = Convert.ToUInt32(textBox_RWPwd.Text, 16); // ex. 00000000
```

```
WriteCmd.wlength = (INT8U)Convert.ToUInt32(textBox_WCnt.Text); // ex. 6
```

```
WriteCmd.offset = (INT8U)Convert.ToUInt32(textBox_OffSet.Text); // ex. 2
```

```
WriteCmd.bank = RFIDUHF.RFIDTagBank.TAG_EPC;
```

```
String WriteData = textBox_Write_Data.Text; //
```

```
UInt16[] Data = new UInt16[32];
```

```
for (int i = 0; i < Convert.ToUInt32(textBox_WCnt.Text); i++)
```

```
{
```

```
Data[i] = Convert.ToUInt16(WriteData.Substring(i * 4, 4), 16);  
}
```

```
WriteCmd.wdata = Data;
```

```
status = _UHFNet.Write(ref WriteCmd);
```

```
if (status != RFIDUHF.RFID_STATUS.RFID_STATUS_OK)  
{  
    MessageBox.Show("RFID WRITE FAIL!");  
}
```

4.6.9 Lock

Description

Limit access to tag

Syntax

```
RFIDUHF.RFID_STATUS Lock(ref RFIDUHF.RFIDLockCmd cmd);
```

Parameters

cmd

RFIDLockCmd structure is used to limit access to tag

Return Value

Returns RFID_STATUS of enum type

Remarks

An event is logged through delegate void ReceivedMemoryData() after lock command. Then, the result can be obtained using GetError.

Access password in reserved memory is used to limit access.

RFIDLockPermission Enum value in RFIDLockCmd structure is used.

PERMISSION_ACCESSIBLE:

Read and write of password is possible. Change permission is also possible.

PERMISSION_ALWAYS_ACCESSIBLE: Read and write of password is possible. But, change permission is not possible.

PERMISSION_SECURED_ACCESSIBLE: Read and write of password is not possible. But, change permission is possible.

PERMISSION_ALWAYS_NOT_ACCESSIBLE: Read and write of password is not possible. Change permission is also not possible.

Read / Write accessibility setting is possible in reserved area. EPC, USER area only allow setting for write, where read is always possible. TID is read only.

See Also

GetError

For C++

Library : RFID_UHF.lib

Function : RFID_STATUS UHF_Lock(RFIDLockCmd *cmd);

Example

```
RFIDUHF.RFID_STATUS status = new RFIDUHF.RFID_STATUS();
```

```
RFIDUHF.RFIDLockCmd LockCmd = new RFIDUHF.RFIDLockCmd();
```

```
LockCmd.lockkillpwd = RFIDUHF.RFIDLockPermission.PERMISSION_NO_CHANGE;
```

```
LockCmd.lockaccesspwd = RFIDUHF.RFIDLockPermission.PERMISSION_SECURED_ACCESSIBLE;
```

```
LockCmd.lockepc = RFIDUHF.RFIDLockPermission.PERMISSION_SECURED_ACCESSIBLE;
```

```
LockCmd.locktid = RFIDUHF.RFIDLockPermission.PERMISSION_NO_CHANGE;
```

```
LockCmd.lockuser = RFIDUHF.RFIDLockPermission.PERMISSION_NO_CHANGE;
```

```
LockCmd.accpwd = Convert.ToInt32(textBox_Lock_Pwd.Text, 16);
```

```
status = _UHFNet.Lock(ref LockCmd);
```

```
if (status != RFIDUHF.RFID_STATUS.RFID_STATUS_OK)
```

```
{  
    MessageBox.Show("RFID LOCK FAIL!");  
}
```

4.6.10 Kill

Description

Kill the tag (disabling the tag)

Syntax

```
RFIDUHF.RFID_STATUS Kill(ref RFIDUHF.RFIDKillCmd cmd);
```

Parameters

cmd

RFIDKillCmd structure is used to disable tag

Return Value

Returns RFID_STATUS of enum type

Remarks

An event is logged through delegate void ReceivedMemoryData() after kill command. Then, the result can be obtained using GetError.

See Also

GetError

For C++

Library : RFID_UHF.lib

Function : RFID_STATUS UHF_Kill(RFIDKillCmd *cmd);

Example

```
RFIDUHF.RFID_STATUS status = new RFIDUHF.RFID_STATUS();  
RFIDUHF.RFIDKillCmd KillCmd = new RFIDUHF.RFIDKillCmd();
```

```
KillCmd.killpwd = uint.Parse(textBox_Kill_Pwd.Text);
```

```
status = _UHFNet.Kill(ref KillCmd);
```

```
if (status != RFIDUHF.RFID_STATUS.RFID_STATUS_OK)  
{  
    MessageBox.Show("RFID KILL FAIL!");  
}
```

4.6.11 GetData

Description

Get tag data from Inventory and Read

Syntax

```
int GetData(StringBuilder strTagData);
```

Parameters

strTagData

[out] get current data

Return Value

Returns the length of the data

Remarks

Data obtained by inventory or read can be checked. If return value is 0, error can be checked using GetError.

See Also

Inventory, Read, GetError

For C++

Library : RFID_UHF.lib

Function : int UHF_GetData (INT8U *data);

Example

```
int nTaglenth = 0;
String strTagData = null;
StringBuilder strData = new StringBuilder(260);
nTaglenth = UHFNet.GetData(strData);
```

4.6.12 SetRegionFrequency

Description

Set RFID frequency to suit regional regulation

Syntax

RFIDUHF.RFID_STATUS SetRegionFrequency(RFIDUHF.RFIDRegion region);

Parameters

region

Set regional frequency by assigning Enum type variable

Return Value

Returns RFID_STATUS of enum type

Remarks

None

See Also

GetRegionFrequency

For C++

Library : RFID_UHF.lib

Function : RFID_STATUS UHF_SetRegionFrequency(RFIDRegion region);

Example

```
RFIDUHF.RFID_STATUS status = new RFIDUHF.RFID_STATUS();
status = _UHFNet.SetRegionFrequency(_nRegionIndex[listBox_Region.SelectedIndex]);
if(status != RFIDUHF.RFID_STATUS.RFID_STATUS_OK)
{
    MessageBox.Show("SetRegionFrequency Set Fail!");
}
```

```
    return;  
}
```

4.6.13 GetRegionalFrequency

Description

Check current regional frequency setting

Syntax

```
RFIDUHF.RFIDRegion GetRegionFrequency();
```

Parameters

None

Return Value

Returns RFID_STATUS of enum type

Remarks

None

See Also

SetRegionFrequency

For C++

Library : RFID_UHF.lib

Function : RFIDRegion UHF_GetRegionFrequency();

Example

```
RFIDUHF.RFIDRegion region = _UHFNet.GetRegionFrequency();
```

4.6.14 ReadBattery

Description

Check battery voltage and ADC value

Syntax

```
RFIDUHF.RFID_STATUS ReadBattery(ref uint nADCValue, ref float fVolt);
```

Parameters

nADCValue

[out] Check ADC value of battery

fVolt

[out] Check current battery voltage

Return Value

Returns RFID_STATUS of enum type

Remarks

None

See Also

ReadBatteryStatus

For C++

Library : RFID_UHF.lib

Function : RFID_STATUS UHF_ReadBattery(INT32U *nADCValue, float *fVolt);

Example

None

4.6.15 ReadBatteryStatus

Description

Check battery level (0~4 level)

Syntax

RFIDUHF.RFID_STATUS ReadBatteryStatus(ref RFIDUHF.BATTERY_STATUS step);

Parameters

step

[out]Check battery level through Enum type BATTERY_STATUS

Return Value

Returns RFID_STATUS of enum type

Remarks

None

See Also

ReadBattery

For C++

Library : RFID_UHF.lib

Function : RFID_STATUS UHF_ReadBatteryStatus(BATTERY_STATUS *step);

Example

None

4.6.16 Version

Description

Check DLL and F/W version

Syntax

```
bool Version(ref RFIDUHF.RFID_VERSION LibVer, ref RFIDUHF.RFID_VERSION MacVer, StringBuilder strDllVersion);
```

Parameters

LibVer

[out] Check version of NRFMCCE.dll

MacVer

[out] Check reader firmware version

strDllVersion

[out] Check RFDI_UHF.dll version

Return Value

TRUE = Success

FALSE = Fail

Remarks

None

See Also

None

For C++

Library : RFID_UHF.lib

Function : RFID BOOL UHF_Version(RFID_VERSION *LibVer, RFID_VERSION *MacVer, TCHAR *tzDllVersion);

Example

```
RFIDUHF.RFID_VERSION LibVer = new RFIDUHF.RFID_VERSION();
```

```
RFIDUHF.RFID_VERSION MacVer = new RFIDUHF.RFID_VERSION();
```

```
StringBuilder strVersion = new StringBuilder(260);
```

```
_UHFNet.Version(ref LibVer, ref MacVer, strVersion);
```

```
label_Version.Text = String.Format("Firmware: {0}.{1}.{2} App: {3}", MacVer.major, MacVer.minor, MacVer.maintenance, Program.APP_VERSION);
```

4.7 WLAN

Status Return value

Please refer to the below table for the status value definition.

Status Value Definition	Code	Meaning
TRUE	1	Success
FALSE	0	General Error

Enum

```
Public enum WLAN_ERROR_RESULT
{
    SUCCESS=0,
    FAIL,
    INVALID_NAME,
    INVALID_CONFIG,
    INVALID_DELETE,
    POWERCYCLE_REQUIRED,
    INVALID_PARAMETER,
    INVALID_EAP_TYPE,
    INVALID_WEP_TYPE,
    INVALID_FILE
};
```

Structure

```
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
public struct WLAN_STATUS
{
    public int    Channel;
    public int    Rssi;
    public double BitRate;
    public int    txPower;
    public int    DTIM;
    public int    BeaconPeriod;
    public int    BeaconReceived;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 32)]
    public char[] SSID;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 36)]
    public char[] CardState;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public byte[] Client_Mac;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
    public byte[] Client_IP;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
```

```

    public byte[] AP_Mac;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
    public byte[] AP_IP;
};
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
public class WLAN_SSID
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]
    public String SSIDName;
    public int privacy;
    public int Rssi;
};
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
public struct WLAN_SSID_LIST
{
    public Int32 count;
    public IntPtr SSID_list;
};
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
public class WLAN_CONFIG_NAME
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]
    public String ConfigName;
};
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
public struct WLAN_CONFIG_NAME_LIST
{
    public Int32 count;
    public IntPtr m_ConfigName;
};

```

Functions for WLAN

Name	Description
ActivateConfig	Activate the configuration with the given name.
Close	Free WLAN.dll.
ConnectAP	Connecting to AP.
DeleteConfig	This function deletes the configuration matching 'name'.
ExportConfig	This function exports configurations.
ImportConfig	This function imports configurations.
Init	WLAN.dll initiation
GetAllConfigName	This function retrieves all of the configurations.
GetCurrentAPInfo	Get current AP information
GetBssidList	Scans and lists connectable APs.
GetPowerStatus	Get power status.
PowerOn	Inserts WLAN card.
PowerOff	Removes WLAN card.

4.7.1 ActivateConfig

Description

Activate the configuration with the given name.

Syntax

```
bool ActivateConfig(string szConfigName);
```

Parameters

szConfigName

Name of the configuration to make the active one.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

This function succeeds even if the card is not present so, when it is inserted, this becomes the active configuration.

See Also

None

For .C++

Function : WLAN_ActivateConfig(TCHAR* ConfigName);

Example

```
bool result=Wlan.ActivateConfig("SSID");  
if(!result)  
    MessageBox.Show("Fail To ActivateConfig()");
```

4.7.2 Close

Description

Free WLAN.dll.

Syntax

```
bool Close();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Init

For .C++

Function : WLAN_API BOOL WLAN_Close()

Example

```
if(Wlan.Close() == false)
{
    MessageBox.Show("Fail To WLAN_Close()");
}
```

4.7.3 ConnectAP

Description

Connecting to AP.

Syntax

bool ConnectAP(string szBssid);

bool ConnectAP(string szBssid, string szPassword, int iEncryptionType);

Parameters

szBssid

Name of the configuration to connect the one.

szPassword

Password.

iEncryptionType

Type	Description
0	Open
1	WEP
2	WPA PSK
3	WPA2 PSK

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For .C++

Function : WLAN_API int WLAN_ConnectAP(TCHAR* szBssid, TCHAR* password, int encryptionType);

Example

```

bool result=false;

// Security
result= Wlan.ConnectAP("SSID");
if(result==false)
    MessageBox.Show("Fail to ConnectAP()");

// Open
result=Wlan.ConnectAP("SSID", "1234", 1);
if(result==false)
    MessageBox.Show("Fail to ConnectAP()");

```

4.7.4 ConnectAPEX

Description

Connecting to AP.

Syntax

```
bool ConnectAPEX(string szBssid, string szPassword, int iEncryptionType, int iWepKeyType)
```

Parameters

szBssid

Name of the configuration to connect the one.

szPassword

Password.

iEncryptionType

Type	Description
0	Open
1	WEP
2	WPA PSK
3	WPA2 PSK

iWepKeyType

iWepKeyType of default parameter is zero.

Type	Description
0	Not Set
1	40bit
2	128bit

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For .C++

Function : `WLAN_API int WLAN_ConnectAPEX(TCHAR* szBssid, TCHAR* szPassword, int iEncryptionType , int iWepKeyType);`

Example

```
bool result=false;

// Security

result= Wlan.ConnectAP("SSID");

if(result==false)

    MessageBox.Show("Fail to ConnectAP()");

// Open

result=Wlan.ConnectAPEX("SSID", "1234", 1, 2);

if(result==false)

    MessageBox.Show("Fail to ConnectAP()");
```

4.7.5 DeleteConfig

Description

This function deletes the configuration matching 'name'.

Syntax

```
bool DeleteConfig(string szConfigName);
```

Parameters

szConfigName

Name of the configuration to delete the one.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

You are not allowed to delete the active configuration.

See Also

DeleteConfig

For C++

Function : bool BOOL WLAN_DeleteConfig(TCHAR *ConfigName)

Example

```
bool result= Wlan.DeleteConfig("SSID");  
if(!result)  
    MessageBox.Show("Activate config can not removed!!");
```

4.7.6 ExportConfig

Description

This function exports configurations.

Syntax

```
bool ExportConfig(string szExportPath);
```

Parameters

szExportPath

File name and route to be stored.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

ImportConfig

For .C++

Function : WLAN_API BOOL WLAN_ExportConfig(TCHAR* ExportPath);

Example

```
if (FileDialog.ShowDialog() == DialogResult.OK)  
    m_filepath.Text = FileDialog.FileName;  
if (Wlan.ExportConfig(m_filepath.Text))  
    m_result.Text="Export Success!!";  
else  
    m_result.Text = "Export Fail!!!";
```

4.7.7 ImportConfig

Description

This function imports configurations.

Syntax

```
bool ImportConfig(string szImportPath);
```

Parameters

szImportPath

File name and route to get.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

ExportConfig

For C++

Function : WLAN_API BOOL WLAN_ImportConfig(TCHAR* ImportPath);

Example

```
if (FileDialog2.ShowDialog() == DialogResult.OK)
    m_filepath.Text = FileDialog2.FileName;
if (Wlan.ImportConfig(m_filepath.Text))
    m_result.Text="Import Success!!";
else
    m_result.Text = "Import Fail!!!";
```

4.7.8 Init

Description

WLAN moudule initiation.

Syntax

```
bool Init() ;
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Close

For C++

Function : WLAN_API BOOL WLAN_Init();

Example

```
if(Wlan.Init() == false)
    MessageBox.Show("Fail To Init());
```

4.7.9 GetAllConfigName

Description

This function retrieves all of the configurations.

Syntax

```
unsafe int GetAllConfigName(ref WLAN_CONFIG_NAME[] pstConfigList) ;
```

Parameters

WLAN_CONFIG_NAME[] pstConfigList

Pointer to a WLAN_CONFIG_NAME structure to be filled in with AP list parameters.

Return Value

Return WLAN_ERROR_RESULT

Remarks

Except ThirdPartyConfig.

See Also

None

For C++

Function : WLAN_API int WLAN_GetAllConfigName(WLAN_CONFIG_NAME_LIST* pstConfigList);

Example

```
WLAN_CONFIG_NAME[] NameList = new WLAN_CONFIG_NAME[20];
Wlan.GetAllConfigName(ref NameList);
```

4.7.10 GetCurrentAPInfo

Description

Gets current AP information except txPower .

Syntax

```
bool GetCurrentAPInfo(ref WLAN_STATUS pstStatus);
```

Parameters

WLAN_STATUS pstStatus

Pointer to a WLAN_STATUS structure to be filled in with AP info parameters.

Return Value

Return WLAN_ERROR_RESULT

Remarks

None

See Also

None

For C++

Function : WLAN_API int WLAN_GetCurrentAPInfo(P_WLAN_STATUS st)

Example

```
WLAN_STATUS st = new WLAN_STATUS();
Wlan.GetCurrentAPInfo(ref st);
if(result == 0)
    MessageBox.Show("Success To GetCurrentAPInfo ()");
```

4.7.11 GetBssidList

Description

Scans and lists connectable APs.

Syntax

unsafe int GetBssidList(ref WLAN_SSID[] pstSsidlist);

Parameters

WLAN_SSID[] pstSsidList

Pointer to a WLAN_SSID structure to be filled in SSIDs.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Function : WLAN_API BOOL WLAN_GetBssidList(WLAN_SSID_LIST* SsidList);

Example

```
WLAN_SSID[] ssidlist = new WLAN_SSID[40];
Cursor.Current = Cursors.WaitCursor;
do
{
```



```
Wlan.GetBssidList(ref ssidlist);  
} while (ssidlist[0].SSIDName == "");
```

4.7.12 GetPowerStatus

Description

Gets power status.

Syntax

```
bool GetPowerStatus();
```

Parameters

None

Return Value

The return value is SCAN_ENGINE_TYPE.

Remarks

None

See Also

None

For C++

Function : WLAN_API BOOL WLAN_GetPowerStatus()

Example

```
if(WLAN_GetPowerStatus())  
    AfxMessageBox(_T("Power On!"));  
else  
    AfxMessageBox(_T("Power On!"));
```

4.7.13 PowerOn

Description

Inserts WLAN card.

Syntax

```
bool PowerOn();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

PowerOff

For C++

Function : WLAN_API BOOL WLAN_PowerOn()

Example

```
bool result = Wlan.PowerOn();  
if(!result)  
    MessageBox.Show("Fail To PowerOn()");
```

4.7.14 PowerOff

Description

Removing WLAN card.

Syntax

```
bool PowerOff();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

PowerOn

For C++

Function : WLAN_API BOOL WLAN_PowerOff()

Example

```
bool result = Wlan.PowerOff();  
if(!result)  
    MessageBox.Show("Fail To PowerOff()");
```

4.8 BLUETOOTH

Status Return value & API Error Codes

Please refer to the below table for the status value definition.

Status Value Definition	Code	Meaning
TRUE	1	Success
FALSE	0	General Error

Using
<pre>using HLOCAL = System.IntPtr; using HANDLE = System.IntPtr; using BT_Connection_ID = System.UInt32; using HRESULT = System.Int32; using LPVOID = System.Array; using BT_Device_Find = System.IntPtr; using BT_Service_Find = System.IntPtr; using BT_Connection_Find = System.IntPtr;</pre>
Event
<pre>// Authentication callback prototype public delegate Int32 BT_Authentication_Callback_t(BD_ADDR_t BD_ADDR, SByte Passkey, LPVOID CallbackParameter); // Connection Event callback prototype. public delegate void BT_Connection_Callback_t(BT_Connection_ID ConnectionID, bool ConnectionState, LPVOID CallbackParameter);</pre>
Const
<pre>// Message Queue Commands public const Byte PING = 0; public const Byte BT_FIND_FIRST_DEVICE = 1; public const Byte BT_FIND_NEXT_DEVICE = 2; public const Byte BT_FIND_DEVICE_CLOSE = 3; public const Byte BT_FIND_FIRST_SERVICE = 4; public const Byte BT_FIND_NEXT_SERVICE = 5; public const Byte BT_FIND_SERVICE_CLOSE = 6; public const Byte BT_FIND_FIRST_CONNECTION = 7; public const Byte BT_FIND_NEXT_CONNECTION = 8; public const Byte BT_FIND_CONNECTION_CLOSE = 9; public const Byte BT_CREATE_CONNECTION = 10; public const Byte BT_DELETE_CONNECTION = 11; public const Byte BT_CONNECT = 12; public const Byte BT_DISCONNECT = 13; public const Byte BT_SET_AUTHENTICATION_CALLBACK = 14; public const Byte BT_SET_CONNECTION_CALLBACK = 15; public const Byte BT_FIND_LOCAL_DEVICE = 16;</pre>

```
public const Byte BT_SET_INCOMING_PIN = 17;
public const Byte BT_SET_refGOING_PIN = 18;
public const Byte BT_PERFORM_ACTION = 19;
public const Byte BT_SET_SECURITY_MODE = 20;
public const Byte BT_GET_SECURITY_MODE = 21;
public const Byte BT_SET_SCO_CONNECTION_STATE = 22;
public const Byte BT_GET_SCO_CONNECTION_STATE = 23;
// Maximum message size allowed by the message queues.
public const int MAX_MSG_SIZE = 1024;
public const int MAX_NAME_LENGTH = 248;
// Maximum name length, including delimiter.
public const int BLUETOOTH_MAX_NAME_SIZE = MAX_NAME_LENGTH + 1;
// Error codes.
public const Byte BT_ERROR = 0;
public const Byte BT_ERROR_SUCCESS = 1;
public const Byte BT_ERROR_INVALID_PARAMETER = 2;
public const Byte BT_ERROR_INVALID_HANDLE = 3;
public const Byte BT_ERROR_NO_MORE = 4;
public const Byte BT_ERROR_MSG_SEND = 5;
public const Byte BT_ERROR_MSG_RECEIVE = 6;
public const Byte BT_ERROR_NO_COM_PORT = 7;
public const Byte BT_ERROR_BTEXP_NOT_RUNNING = 8;
public const Byte BT_ERROR_NO_KEY_AVAILABLE = 9;
// Device states
public const Byte BT_DEVICE_STATE_OFF = 0;
public const Byte BT_DEVICE_STATE_ON = 1;
// SCO Connection States
public const Byte BT_SCO_STATE_DISCONNECTED = 0;
public const Byte BT_SCO_STATE_CONNECTED = 1;
// BT_Perform_Action API.
public const UInt32 BT_ACTION_SHOW_SETTINGS = 0x00000001;
public const UInt32 BT_ACTION_DELETE_ALL_DEVICES = 0x00000002;
// Searching for remote devices
public const UInt32 BT_DEVICE_NONE = 0x0001;
public const UInt32 BT_DEVICE_AUTHENTICATED = 0x0002;
public const UInt32 BT_DEVICE_REMEMBERED = 0x0004;
public const UInt32 BT_DEVICE_CONNECTED = 0x0008;
public const UInt32 BT_DEVICE_ALL = 0x8000;
// Flags for the BTP_Connection_Query_t structure.
public const Byte BT_CONNECTION_NONE = 0;
public const Byte BT_CONNECTION_REMEMBERED = 1;
public const Byte BT_CONNECTION_ACTIVE = 2;
public const Byte BT_CONNECTION_ALL = 3;
public const Byte MESSAGE_TO_BTE_HEADER_SIZE = 12;
public const Byte MESSAGE_FROM_BTE_HEADER_SIZE = 4;
```

Enum

```
public enum BT_Profile_Type
{
    BT_PROFILE_SPP,
    BT_PROFILE_DUN,
    BT_PROFILE_FAX,
    BT_PROFILE_LAN,
    BT_PROFILE_FILE_TRANSFER,
    BT_PROFILE_HEADSET,
    BT_PROFILE_HEADSET_AUDIO_GATEWAY,
    BT_PROFILE_HANDS_FREE,
    BT_PROFILE_HANDS_FREE_AUDIO_GATEWAY,
    BT_PROFILE_HID_HOST,
    BT_PROFILE_HID_DEVICE,
    BT_PROFILE_UNKNOWN = -1
};

public enum BT_DeviceType_t
{
    bdAll,
    bdHID
};

public enum SDP_Data_Element_Type_t
{
    deNIL,
    deNULL,
    deUnsignedInteger1Byte,
    deUnsignedInteger2Bytes,
    deUnsignedInteger4Bytes,
    deUnsignedInteger8Bytes,
    deUnsignedInteger16Bytes,
    deSignedInteger1Byte,
    deSignedInteger2Bytes,
    deSignedInteger4Bytes,
    deSignedInteger8Bytes,
    deSignedInteger16Bytes,
    deTextString,
    deBoolean,
    deURL,
    deUUID_16,
    deUUID_32,
    deUUID_128,
    deSequence,
    deAlternative
};

public enum BT_Callback_Type
```

```

{
    BT_AUTHENTICATION_CALLBACK,
    BT_CONNECTION_CALLBACK
};

public enum BT_Security_Mode_Type
{
    BT_SECURITYMODE_NONE,
    BT_SECURITYMODE_AUTHENTICATE,
    BT_SECURITYMODE_AUTHENTICATE_AND_ENCRYPT,
};

public enum ListType
{
    DeviceFind = 1,
    ServiceFind,
    ConnectionFind
};

```

Structure

```

// Structure specifying the criteria for device searches.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Device_Query_t
{
    public UInt16 DeviceAttributes;
    public Byte InquiryTimeref;
};

// Structure specifying the criteria for device searches.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Device_Query_Ex_t
{
    public uint Size;
    public Byte Version;
    public UInt16 DeviceAttributes;
    public Byte InquiryTimeout;
    public BT_DeviceType_t DeviceType;
};

[StructLayout(LayoutKind.Sequential)]
public struct BD_ADDR_t
{
    public Byte BD_ADDR0;
    public Byte BD_ADDR1;
    public Byte BD_ADDR2;
    public Byte BD_ADDR3;
    public Byte BD_ADDR4;
    public Byte BD_ADDR5;
};

[StructLayout(LayoutKind.Sequential)]

```

```

public struct Class_of_Device_t
{
    public Byte Class_of_Device0;
    public Byte Class_of_Device1;
    public Byte Class_of_Device2;
};

// Structure containing information regarding a remote device.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Device_Info_t
{
    public BD_ADDR_t BD_ADDR;
    public Class_of_Device_t ClassOfDevice;
    public UInt16 DeviceAttributes;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = MAX_NAME_LENGTH + 1)]
    public byte[] Name;
};

[StructLayout(LayoutKind.Sequential)]
public struct BT_Device_Save_t
{
    public BT_Device_Query_Ex_t DeviceQueryEx;
    public UInt32 index;
    public UInt32 numDevices;
};

[StructLayout(LayoutKind.Sequential)]
public struct UUID_16_t
{
    public Byte UUID_Byte0;
    public Byte UUID_Byte1;
};

[StructLayout(LayoutKind.Sequential)]
public struct UUID_32_t
{
    public Byte UUID_Byte0;
    public Byte UUID_Byte1;
    public Byte UUID_Byte2;
    public Byte UUID_Byte3;
};

[StructLayout(LayoutKind.Sequential)]
public struct UUID_128_t
{
    public Byte UUID_Byte0;
    public Byte UUID_Byte1;
    public Byte UUID_Byte2;
    public Byte UUID_Byte3;
    public Byte UUID_Byte4;
    public Byte UUID_Byte5;
}

```

```

    public Byte UUID_Byte6;
    public Byte UUID_Byte7;
    public Byte UUID_Byte8;
    public Byte UUID_Byte9;
    public Byte UUID_Byte10;
    public Byte UUID_Byte11;
    public Byte UUID_Byte12;
    public Byte UUID_Byte13;
    public Byte UUID_Byte14;
    public Byte UUID_Byte15;
};

[StructLayout(LayoutKind.Explicit)]
public struct UUID_Value
{
    [FieldOffset(0)]
    public UUID_16_t UUID_16;
    [FieldOffset(0)]
    public UUID_32_t UUID_32;
    [FieldOffset(0)]
    public UUID_128_t UUID_128;
};

[StructLayout(LayoutKind.Sequential)]
public struct SDP_UUID_Entry_t
{
    public SDP_Data_Element_Type_t SDP_Data_Element_Type;
    public UUID_Value value;
};

[StructLayout(LayoutKind.Sequential)]
public struct BT_Service_Query_t
{
    public BD_ADDR_t BD_ADDR;
    public UInt16 NumberServiceUUID;
    public IntPtr Service;
};

[StructLayout(LayoutKind.Sequential)]
public struct BT_Service_Info_t
{
    public BT_Profile_Type ProfileType;
    public Byte MajorVersion;
    public Byte MinorVersion;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = BLUETOOTH_MAX_NAME_SIZE)]
    public byte[] ServiceName;
    public UInt32 RFCOMMPort;
};

[StructLayout(LayoutKind.Sequential)]
public struct BTSerialPortProfileInfo_t

```



```

{
    public UInt32 RFCOMMServerPort;
    public bool UseActiveSync;
};

[StructLayout(LayoutKind.Sequential)]
public struct BTHIDProfileInfo_t
{
    public UInt32 L2CAPControlChannel;
    public UInt32 L2CAPInterruptChannel;
    public Byte DeviceSubclass;
    public bool VirtualCableSupported;
    public bool DeviceAutomaticReconnect;
    public bool DeviceNormallyConnectable;
};

[StructLayout(LayoutKind.Explicit)]
public struct ProfileInformation
{
    [FieldOffset(0)]
    public BTSerialPortProfileInfo_t RemoteSerialPortProfileInfo;
    [FieldOffset(0)]
    public BTHIDProfileInfo_t RemoteHIDProfileInfo;
};

// Structure containing remote service or Favorite device information.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Service_Info_Ex_t
{
    public UInt32 Size;
    public UInt32 Version;
    public BT_Profile_Type ProfileType;
    public Byte MajorVersion;
    public Byte MinorVersion;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = BLUETOOTH_MAX_NAME_SIZE)]
    public Byte[] ServiceName;
    public UInt32 ListIndex; //not used in version 2. For future enhancements
    public ProfileInformation _profileinformation;
};

// Structure used to store data between subsequent BTP_Find_Next_Service calls.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Service_Save_t
{
    public BT_Service_Query_t ServiceQuery;
    public UInt32 index;
    public UInt32 numProfiles;
    public HLOCAL RemoteProfileList;
};

public struct BT_Connection_Query_t

```

```

{
    public UInt16 ConnectionAttributes;
};
[StructLayout(LayoutKind.Sequential)]
public struct BT_Connection_Info_t
{
    public BT_Connection_ID ConnectionID;
    public BD_ADDR_t BD_ADDR;
    public UInt32 RFCOMMPort;
    public int LocalCOMPort;
    public Byte MajorVersion;
    public Byte MinorVersion;
    public UInt16 ConnectionAttributes;
    public BT_Profile_Type ProfileType;
};
[StructLayout(LayoutKind.Sequential)]
public struct BT_Connection_Info_Ex_t
{
    public UInt32 Size;
    public UInt32 Version;
    public BT_Connection_ID ConnectionID;
    public BD_ADDR_t BD_ADDR;
    public int LocalCOMPort;
    public Byte MajorVersion;
    public Byte MinorVersion;
    public UInt16 ConnectionAttributes;
    public BT_Profile_Type ProfileType;
    public HLOCAL ListHandle;// not used in version 2. For future enhancements.
    public UInt32 ListIndex;// not used in version 2. For future enhancements.
    public ProfileInformation _profileinformation;
};
// Structure used to store data between subsequent BTP_Find_Next_Connection calls.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Connection_Save_t
{
    public UInt32 index1;
    public UInt32 index2;
    public UInt32 index3;
    public UInt32 numDevices;
    public UInt32 numFavorites;
    public HLOCAL RemoteDeviceList;
    public HLOCAL FavoriteDeviceList;
    public BT_Connection_Query_t ConnectionQuery;
};
// Structure used to store a security PIN to be used for authentication.
[StructLayout(LayoutKind.Sequential)]

```

```

public struct BT_PIN_Code_t
{
    public Byte PIN_Code0;
    public Byte PIN_Code1;
    public Byte PIN_Code2;
    public Byte PIN_Code3;
    public Byte PIN_Code4;
    public Byte PIN_Code5;
    public Byte PIN_Code6;
    public Byte PIN_Code7;
    public Byte PIN_Code8;
    public Byte PIN_Code9;
    public Byte PIN_Code10;
    public Byte PIN_Code11;
    public Byte PIN_Code12;
    public Byte PIN_Code13;
    public Byte PIN_Code14;
    public Byte PIN_Code15;
};

[StructLayout(LayoutKind.Sequential)]
public struct BT_Authentication_Callback_Data_t
{
    public BD_ADDR_t BD_ADDR;
    public LPVOID CallbackParameter;
    public BT_Authentication_Callback_t AuthenticationCallback;
    public BT_Authentication_Callback_Data_t[] NextAuthenticationCallbackData;
};

// This structure allows for creating linked lists of Connection callbacks. [StructLayout(LayoutKind.Sequential)]
public struct BT_Connection_Callback_List_t
{
    public BT_Connection_Callback_t ConnectionCallback;
    public LPVOID CallbackParameter;
    public BT_Connection_Callback_List_t[] NextConnectionCallback;
};

// Structure containing all pertinent data associated with the Connection Callback so that a list of callbacks can be made.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Connection_Callback_Data_t
{
    public BT_Connection_ID ConnectionID;
    public BT_Connection_Callback_List_t ConnectionCallbackList;
    public BT_Connection_Callback_Data_t[] NextConnectionCallbackData;
};

[StructLayout(LayoutKind.Explicit)]
public struct CallbackData
{

```

```

[FieldOffset(0)]
public BD_ADDR_t BD_ADDR;
[FieldOffset(0)]
public BT_Connection_ID ConnectionID;
};

// Structure containing all data needed by CallbackNotifyThreadProc to make the proper callback.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Callback_Notify_Data_t
{
    public BT_Callback_Type CallbackType;
    public bool ConnectionState;
    CallbackData _callbackdata;
};

// Structure containing all the data returned to BTE Explorer by any of the callbacks.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Callback_Reply_Data_t
{
    public bool Success;
    public SByte Password; //[sizeof(PIN_Code_t)+1]
};

// Device Find message information to BTE Explorer
public struct BT_Find_First_Device_To_BTE_t
{
    public BT_Device_Query_Ex_t DeviceQuery;
};

public struct BT_Find_Next_Device_To_BTE_t
{
    public BT_Device_Find DeviceFind;
};

public struct BT_Find_Device_Close_To_BTE_t
{
    public BT_Device_Find DeviceFind;
};

// Service Find message information to BTE Explorer
[StructLayout(LayoutKind.Sequential)]
public struct BT_Find_First_Service_To_BTE_t
{
    public BT_Service_Query_t ServiceQuery;
    public SDP_Data_Element_Type_t Service;
};

public struct BT_Find_Next_Service_To_BTE_t
{
    public BT_Service_Find ServiceFind;
};

public struct BT_Find_Service_Close_To_BTE_t

```

```

{
    public BT_Service_Find ServiceFind;
};

// Connection Find message information to BTEplorer
public struct BT_Find_First_Connection_To_BTE_t
{
    public BT_Connection_Query_t ConnectionQuery;
};

public struct BT_Find_Next_Connection_To_BTE_t
{
    public BT_Connection_Find ConnectionFind;
};

public struct BT_Find_Connection_Close_To_BTE_t
{
    public BT_Connection_Find ConnectionFind;
};

// Set Connection Callback message information to BTEplorer
[StructLayout(LayoutKind.Sequential)]
public struct BT_Set_Connection_Callback_to_BTE_t
{
    public BT_Connection_ID ConnectionID;
    public HANDLE NotifyMsgQueue;
};

// Set Perform Action message information to BTEplorer
[StructLayout(LayoutKind.Sequential)]
public struct BT_Perform_Action_To_BTE_t
{
    public UInt32 Action;
    public UInt32 Param1;
    public UInt32 Param2;
};

[StructLayout(LayoutKind.Sequential)]
public struct BT_PIN_t
{
    public UInt32 PINLength;
    public BT_PIN_Code_t PINCode;
};

[StructLayout(LayoutKind.Sequential)]
public struct BT_Set_SCO_Connection_State_To_BTE_t
{
    public BD_ADDR_t BD_ADDR;
    public UInt32 State;
};

// Set (and get) security mode message to BTEplorer
public struct BT_Security_Mode_t
{

```

```

    public BT_Security_Mode_Type SecurityMode;
};

[StructLayout(LayoutKind.Sequential)]
public struct Message_To_BTE_t
{
    public int Command;
    public HANDLE ResponseQueueHandle;
    public UInt32 ProcessID;
    public CommandData _commanddata;
};

// Find local device message information from BTEplorer
public struct BT_Find_Local_Device_From_BTE_t
{
    public BT_Device_Info_t DeviceInfo;
};

[StructLayout(LayoutKind.Sequential)]
public struct BT_Find_First_Device_From_BTE_t
{
    public BT_Device_Find DeviceFind;
    public BT_Device_Info_t DeviceInfo;
};

public struct BT_Find_Next_Device_From_BTE_t
{
    public BT_Device_Info_t DeviceInfo;
};

// Find Service message information from BTEplorer.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Find_First_Service_From_BTE_t
{
    public BT_Service_Find ServiceFind;
    public BT_Service_Info_Ex_t ServiceInfo;
};

public struct BT_Find_Next_Service_From_BTE_t
{
    public BT_Service_Info_Ex_t ServiceInfo;
};

// Find Connection message information from BTEplorer.
[StructLayout(LayoutKind.Sequential)]
public struct BT_Find_First_Connection_From_BTE_t
{
    public BT_Connection_Find ConnectionFind;
    public BT_Connection_Info_Ex_t ConnectionInfo;
};

public struct BT_Find_Next_Connection_From_BTE_t
{
    public BT_Connection_Info_Ex_t ConnectionInfo;
};

```

```

};

[StructLayout(LayoutKind.Sequential)]
public struct BT_Perform_Action_From_BTE_t
{
    public UInt32 Return1;
    public UInt32 Return2;
};

public struct BT_Get_SCO_Connection_State_From_BTE_t
{
    public UInt32 State;
};

[StructLayout(LayoutKind.Explicit)]
public struct CommandData
{
    [FieldOffset(0)]
    public BT_Find_First_Device_From_BTE_t FirstDeviceData;
    [FieldOffset(0)]
    public BT_Find_Next_Device_From_BTE_t NextDeviceData;
    [FieldOffset(0)]
    public BT_Find_First_Service_From_BTE_t FirstServiceData;
    [FieldOffset(0)]
    public BT_Find_Next_Service_From_BTE_t NextServiceData;
    [FieldOffset(0)]
    public BT_Find_First_Connection_From_BTE_t FirstConnectionData;
    [FieldOffset(0)]
    public BT_Find_Next_Connection_From_BTE_t NextConnectionData;
    [FieldOffset(0)]
    public BT_Find_Local_Device_From_BTE_t LocalDeviceData;
    [FieldOffset(0)]
    public BT_Get_SCO_Connection_State_From_BTE_t SCOConnectionData;
    [FieldOffset(0)]
    public BT_Connection_Info_Ex_t ConnectionInfo;
    [FieldOffset(0)]
    public BT_PIN_t PINData;
    [FieldOffset(0)]
    public BT_Perform_Action_From_BTE_t ActionResult;
    [FieldOffset(0)]
    public BT_Security_Mode_t SecurityModeData;
};

// Common Incoming Message Structure.
[StructLayout(LayoutKind.Sequential)]
public struct Message_From_BTE_t
{
    public HRESULT Result;
    public CommandData _commanddata;
};

```

Functions for Bluetooth

Name	Description
Close	Performs cleanup after the API is no longer needed.
Connect	Activates the specified connection in the connection list.
CreateConnection	Defines a new connection, adding it to a list of connections.
CreateConnectionEx	Defines a new connection, adding it to a list of connections.
DeleteConnection	Deletes a connection from the list of connections.
Disconnect	Disconnects from the specified connection.
FindConnectionClose	Frees remote resources.
FindDeviceClose	Frees remote resources.
FindFirstConnection	Finds the first connection meeting the specified criteria.
FindFirstConnectionEx	Finds the first connection meeting the specified criteria.
FindFirstDevice	Performs a GAP inquiry to discover devices, returning the first device meeting the specified criteria.
FindFirstDeviceEx	Performs a GAP inquiry to discover devices, returning the first device meeting the specified criteria. This supports searching for a device of specific type.
FindFirstService	Performs an SDP query to discover remote services for a specified device, returning the first service in the list.
FindFirstServiceEx	Performs an SDP query to discover remote services for a specified device, returning the first service in the list.
FindLocalDevice	Returns device information for the local device.
FindNextConnection	Returns the next connection meeting the criteria specified in the call to FindFirstConnection.
FindNextConnectionEx	Returns the next connection meeting the criteria specified in the call to FindFirstConnection.
FindNextDevice	Returns the next device in the list meeting the initial criteria, specified in the call to FindFirstDevice.
FindNextService	Returns the next service in the list.
FindNextServiceEx	Returns the next service in the list.
FindServiceClose	Frees remote resources.
GetBluetoothState	Gets the Bluetooth device state.
GetSCOConnectionState	Gets the SCO connection state.
GetSecurityMode	Gets the current authentication and encryption settings.
Open	Initializes the BTE Explorer API module.
PerformAction	Triggers BTE Explorer to take the requested action.
SetAuthenticationCallback	Sets the authentication callback for the specified device.
SetBluetoothState	Sets the Bluetooth device state to either on or off.

SetConnectionCallback	Sets the connection callback for the specified connection.
SetIncomingPIN	Sets or clears a static PIN to be used for any incoming Connections.
SetOutgoingPIN	Sets or clears a static PIN to be used for any outgoing Connections.
SetSCOConnectionState	Sets the SCO connection state to either connected or disconnected.
SetSecurityMode	Sets the authentication and encryption settings for future connections.

4.8.1 Close

Description

This function is responsible for cleaning up the module after it is no longer needed by the application.

Syntax

```
void BLUETOOTH_Close();
```

Parameters

None

Return Value

None

Remarks

None

See Also

Open

For C++

Library : BLUETOOTH.lib

Function : void BLUETOOTH_Close

Example

```
Bluetooth.Close();
```

4.8.2 Connect

Description

This function connects to a connection previously defined by a call to CreateConnection.

Syntax

```
Int32 BLUETOOTH_Connect(BT_Connection_ID ConnectionID);
```

Parameters

ConnectionID

Unique identifier for a connection which was previously defined by a call to CreateConnection and CreateConnectionEx.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values: BT_ERROR , BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

Disconnect

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_Connect(BTP_Connection_ID ConnectionID)

Example

```
hResult = Bluetooth.Connect(m_ConnectionInfo.ConnectionID);
```

4.8.3 CreateConnection

Description

This function defines a temporary or persistent (Favorite) connection, which may later be connected by a call to Connect (Supports SPP only).

Syntax

```
Int32 BLUETOOTH_CreateConnection(ref BT_Connection_Info_t ConnectionInfo);
```

Parameters

ConnectionInfo

Information defining the new connection. Also, the new connection ID is returned in this structure. This version supports SPP connections only.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values: BT_ERROR, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

DeleteConnection

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_CreateConnection(BTP_Connection_Info_t *ConnectionInfo)

Example

None

4.8.4 CreateConnectionEx

Description

This function defines a temporary or persistent (Favorite) connection, which may later be connected by a call to Connect (Supports SPP and HID).

Syntax

```
Int32 BLUETOOTH_CreateConnectionEx(ref BT_Connection_Info_Ex_t ConnectionInfo);
```

Parameters

ConnectionInfoEx

Information defining the new connection. Also, the new connection ID is returned in this structure. Currently supports connecting to HID and SPP.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

DeleteConnection

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_CreateConnectionEx(BTP_Connection_Info_Ex_t *ConnectionInfo)

Example

```
hResult = Bluetooth.CreateConnectionEx(ref m_ConnectionInfo);
```

4.8.5 DeleteConnection

Description

This function discards a previously defined connection. After deleting a connection, its connection ID is no longer valid.

Syntax

```
Int32 BLUETOOTH_DeleteConnection(BT_Connection_ID ConnectionID);
```

Parameters

ConnectionID

Unique identifier for a connection which was previously defined by a call to CreateConnection and CreateConnectionEx.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR , BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

CreateConnection, CreateConnectionEx

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_DeleteConnection(BTP_Connection_ID ConnectionID)

Example

```
hResult = Bluetooth.DeleteConnection(m_ConnectionInfo.ConnectionID);
```

4.8.6 Disconnect

Description

This function disconnects from an active connection. If the connection is not persistent, the connection is also deleted, invalidating its connection ID.

Syntax

```
Int32 BLUETOOTH_Disconnect(BT_Connection_ID ConnectionID);
```

Parameters

ConnectionID

Unique identifier for a connection which was previously defined by a call to CreateConnection and CreateConnectionEx.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

Connect

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_Disconnect(BTP_Connection_ID ConnectionID)

Example

```
hResult = Bluetooth.Disconnect(m_ConnectionInfo.ConnectionID);
```

4.8.7 FindConnectionClose

Description

This function deletes the remote list of connections and performs any additional cleanup.

Syntax

```
Int32 BLUETOOTH_FindConnectionClose(BT_Connection_Find ConnectionFind);
```

Parameters

ConnectionFind

Handle to the list connections, originally returned by a call to FindFirstConnection or FindFirstConnectionEx.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR , BT_ERROR_INVALID_HANDLE

Remarks

None

See Also

FindFirstConnection, FindFirstConnectionEx

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindConnectionClose(BTP_Connection_Find ConnectionFind)

Example

```
Bluetooth.FindConnectionClose(ConnectFind);
```

4.8.8 FindDeviceClose

Description

This function deletes the remote list of devices and performs any additional cleanup.

Syntax

```
Int32 BLUETOOTH_FindDeviceClose(BT_Device_Find DeviceFind);
```

Parameters

DeviceFind

Handle to the list of discovered devices, originally returned by a call to FindFirstDevice.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR , BT_ERROR_INVALID_HANDLE

Remarks

None

See Also

FindFirstDevice

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindDeviceClose(BTP_Device_Find DeviceFind)

Example

Bluetooth.FindDeviceClose(DeviceFind);

4.8.9 FindFirstConnection

Description

This function creates a list of active connections and Favorites to traverse, returning the first connection from the list.

Syntax

```
Int32 BLUETOOTH_FindFirstConnection(ref BT_Connection_Find ConnectionFind, ref  
BT_Connection_Info_t ConnectionInfo, ref BT_Connection_Query_t ConnectionQuery);
```

Parameters

ConnectionFind

Information defining the new connection. Also, the new connection ID is returned in this structure. This version supports SPP connections only.

ConnectionInfo

Information defining the first connection from the list.

ConnectionQuery

Provides filtering for the types of connection to be retrieved.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR , BT_ERROR_NO_MORE, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

FindConnectionClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindFirstConnection(BTP_Connection_Find *ConnectionFind,
BTP_Connection_Info_t *ConnectionInfo, const BTP_Connection_Query_t *ConnectionQuery)

Example

```
hResult = Bluetooth.FindFirstConnection(ref ConnectFind, ref ConnectionInfo, ref ConnectQuery);
```

4.8.10 FindFirstConnectionEx

Description

This function creates a list of active connections and Favorites to traverse, returning the first connection from the list.

Syntax

```
Int32 BLUETOOTH_FindFirstConnectionEx(ref BT_Connection_Find ConnectionFind, ref  
BT_Connection_Info_Ex_t ConnectionInfoEx, ref BT_Connection_Query_t ConnectionQuery);
```

Parameters

ConnectionFind

Handle to the list of connections, needed for subsequent calls to FindNextConnectionEx and FindConnectionClose.

ConnectionInfoEx

Information defining the first connection from the list. Supports SPP and HID connections.

ConnectionQuery

Provides filtering for the types of connection to be retrieved.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR , BT_ERROR_NO_MORE, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

FindConnectionClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindFirstConnectionEx(BTP_Connection_Find *ConnectionFind, BTP_Connection_Info_Ex_t *ConnectionInfoEx, const BTP_Connection_Query_t *ConnectionQuery)

Example

None

4.8.11 FindFirstDevice

Description

This function initializes a GAP inquiry, creating a list of discovered Bluetooth devices. The first device is returned by this function.

Syntax

```
Int32 BLUETOOTH_FindFirstDevice(ref BT_Device_Find DeviceFind, ref BT_Device_Info_t DeviceInfo,  
ref BT_Device_Query_t DeviceQuery);
```

Parameters

DeviceFind

Handle to the list of discovered devices, needed for subsequent calls to FindNextDevice and FindDeviceClose.

DeviceInfo

Information defining the first device.

DeviceQuery

Provides parameters for the GAP Inquiry and filtering for the devices to retrieve.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values: BT_ERROR , BT_ERROR_NO_MORE, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

FindDeviceClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindFirstDevice(BTP_Device_Find *DeviceFind, BTP_Device_Info_t *DeviceInfo, const BTP_Device_Query_t *DeviceQuery)

Example

None

4.8.12 FindFirstDeviceEx

Description

This function initializes a GAP inquiry, creating a list of discovered Bluetooth devices of a particular device type or all devices. In this version it supports searching for HID devices. The first device that matches the filter is returned by this function.

Syntax

```
Int32 BLUETOOTH_FindFirstDeviceEx(ref BT_Device_Find DeviceFind, ref BT_Device_Info_t DeviceInfo, ref BT_Device_Query_Ex_t DeviceQuery);
```

Parameters

DeviceFind

Handle to the list of discovered devices, needed for subsequent calls to FindNextDevice and FindDeviceClose.

DeviceInfo

Information defining the first device.

DeviceQueryEx

Provides parameters for the GAP Inquiry and filtering for the devices to retrieve.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR , BT_ERROR_NO_MORE, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

FindDeviceClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindFirstDeviceEx(BTP_Device_Find *DeviceFind, BTP_Device_Info_t *DeviceInfo, const BTP_Device_Query_Ex_t *DeviceQuery)

Example

```
hResult = Bluetooth.FindFirstDeviceEx(ref DeviceFind, ref DeviceInfo, ref DeviceQuery);
```

4.8.13 FindFirstService

Description

This function performs an SDP service discovery on the specified device, return the first service from the resulting list.

Syntax

```
Int32 BLUETOOTH_FindFirstService(ref BT_Service_Find ServiceFind, ref BT_Service_Info_t ServiceInfo, ref BT_Service_Query_t ServiceQuery);
```

Parameters

ServiceFind

Handle to the list of remote services, needed for subsequent calls to FindNextService and FindServiceClose.

ServiceInfo

Information defining the first remote service in the list.

ServiceQuery

Provides parameters for the SDP query.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_NO_MORE, BTP_ERROR_INVALID_HANDLE

Remarks

None

See Also

FindServiceClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindFirstService(BTP_Service_Find *ServiceFind, BTP_Service_Info_t *ServiceInfo, const BTP_Service_Query_t *ServiceQuery)

Example

None

4.8.14 FindFirstServiceEx

Description

This function performs an SDP service discovery on the specified device, return the first service from the resulting list.

Syntax

```
Int32 BLUETOOTH_FindFirstServiceEx(ref BT_Service_Find ServiceFind, ref BT_Service_Info_Ex_t ServiceInfo, ref BT_Service_Query_t ServiceQuery);
```

Parameters

ServiceFind

Handle to the list of remote services, needed for subsequent calls to FindNextServiceEx and FindServiceClose.

ServiceInfoEx

Information defining the first remote service in the list. This currently supports SPP and HID services.

ServiceQuery

Provides parameters for the SDP query. This currently supports SPP and HID searching by specifying their UUIDs.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_NO_MORE, BT_ERROR_INVALID_HANDLE

Remarks

None

See Also

FindServiceClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindFirstServiceEx(BTP_Service_Find *ServiceFind, BTP_Service_Info_Ex_t *ServiceInfo, const BTP_Service_Query_t *ServiceQuery)

Example

```
hResult = Bluetooth.FindFirstServiceEx(ref ServiceFind, ref ServiceInfo, ref ServiceQuery);
```

4.8.15 FindLocalDevice

Description

This function will return information about the local device. This will include the Bluetooth address, the friendly name, and the class of device.

Syntax

```
Int32 BLUETOOTH_FindLocalDevice(ref BT_Find_Local_Device_From_BTE_t DeviceInfo);
```

Parameters

DeviceInfo

Will receive information defining the local device.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values: BT_ERROR_INVALID_PARAMETER, BT_ERROR_MSG_SEND

Remarks

None

See Also

None

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindLocalDevice(BTP_Find_Local_Device_From_BTE_t *DeviceInfo)

Example

```
hResult = Bluetooth.FindLocalDevice(ref m_LocalInfo);
```

4.8.16 FindNextConnection

Description

This function retrieves the next connection from the list originally created by a call to FindFirstConnection.

Syntax

```
Int32 BLUETOOTH_FindNextConnection(BT_Connection_Find ConnectionFind, ref  
BT_Connection_Info_t ConnectionInfo);
```

Parameters

ConnectionFind

Handle to the list connections, originally returned by a call to FindFirstConnection.

ConnectionInfo

Information defining a connection from the list.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values: BT_ERROR, BT_ERROR_NO_MORE, BT_ERROR_INVALID_HANDLE

Remarks

None

See Also

FindConnectionClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindNextConnection(BTP_Connection_Find ConnectionFind, BTP_Connection_Info_t *ConnectionInfo)

Example

```
hResult = Bluetooth.FindNextConnection(ConnectFind, ref ConnectionInfo);
```

4.8.17 FindNextConnectionEx

Description

This function retrieves the next connection from the list originally created by a call to FindFirstConnection.

Syntax

```
Int32 BLUETOOTH_FindNextConnectionEx(ref BT_Connection_Find ConnectionFind, ref BT_Connection_Info_Ex_t ConnectionInfo);
```

Parameters

ConnectionFind

Handle to the list connections, originally returned by a call to FindFirstConnection.

ConnectionInfoEx

Information defining a connection from the list. Supports SPP and HID.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_NO_MORE, BT_ERROR_INVALID_HANDLE

Remarks

None

See Also

FindConnectionClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindNextConnectionEx(BTP_Connection_Find ConnectionFind, BTP_Connection_Info_Ex_t *ConnectionInfo)

Example

None

4.8.18 FindNextDevice

Description

This function returns the next device in the list of discovered devices created by a previous call to FindFirstDevice.

Syntax

```
Int32 BLUETOOTH_FindNextDevice(BT_Device_Find DeviceFind, ref BT_Device_Info_t DeviceInfo);
```

Parameters

DeviceFind

Handle to the list of discovered devices, originally returned by a call to FindFirstDevice.

DeviceInfo

Information defining a remote device.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_NO_MORE, BT_ERROR_INVALID_PARAMETER, BT_ERROR_INVALID_HANDLE

Remarks

None

See Also

FindDeviceClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindNextDevice(BTP_Device_Find DeviceFind, BTP_Device_Info_t *DeviceInfo)

Example

```
hResult = Bluetooth.FindNextDevice(DeviceFind, ref DeviceInfo);
```

4.8.19 FindNextService

Description

This function returns the next service in the list of services created by a previous call to FindFirstService.

Syntax

```
Int32 BLUETOOTH_FindNextService(BT_Service_Find ServiceFind, ref BT_Service_Info_t ServiceInfo);
```

Parameters

ServiceFind

Handle to the list of remote services, originally returned by a call to FindFirstService.

ServiceInfo

Information defining a remote service from the list.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_NO_MORE, BT_ERROR_INVALID_HANDLE

Remarks

None

See Also

FindServiceClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindNextService(BTP_Service_Find ServiceFind, BTP_Service_Info_t *ServiceInfo)

Example

None

4.8.20 FindNextServiceEx

Description

This function returns the next service in the list of services created by a previous call to FindFirstService.

Syntax

```
Int32 BLUETOOTH_FindNextServiceEx(BT_Service_Find ServiceFind, ref BT_Service_Info_Ex_t ServiceInfo);
```

Parameters

ServiceFind

Handle to the list of remote services, originally returned by a call to FindFirstService.

ServiceInfoEx

Information defining the first remote service in the list. This currently supports SPP and HID services.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_NO_MORE, BT_ERROR_INVALID_HANDLE

Remarks

None

See Also

FindServiceClose

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindNextService(BTP_Service_Find ServiceFind, BTP_Service_Info_t *ServiceInfo)

Example

```
hResult = Bluetooth.FindNextServiceEx(ServiceFind, ref ServiceInfo);
```

4.8.21 FindServiceClose

Description

This function deletes the remote list of services and performs any additional cleanup.

Syntax

```
Int32 BLUETOOTH_FindServiceClose(BT_Service_Find ServiceFind);
```

Parameters

ServiceFind

Handle to the list of remote services, originally returned by a call to FindFirstService or FindFirstServiceEx.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values: BT_ERROR, BT_ERROR_INVALID_HANDLE

Remarks

None

See Also

FindFirstService, FindFirstServiceEx, FindNextService, FindNextServiceEx

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_FindServiceClose(BTP_Service_Find ServiceFind)

Example

Bluetooth.FindServiceClose(ServiceFind);

4.8.22 GetBluetoothState

Description

This function gets the current Bluetooth device state (either BT_DEVICE_STATE_ON or BT_DEVICE_STATE_OFF).

Syntax

```
Int32 BLUETOOTH_GetBluetoothState(ref UInt32 BluetoothState);
```

Parameters

BluetoothState

Specifies the current Bluetooth state if the function returns BT_ERROR_SUCCESS

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_MSG_SEND

Remarks

None

See Also

SetBluetoothState

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_GetBluetoothState(DWord_t *BluetoothState)

Example

None

4.8.23 GetSCOConnectionState

Description

This function is used to get the SCO connection state for any open Hands-Free connection (either BT_SCO_STATE_CONNECTED or BT_SCO_STATE_DISCONNECTED). If SCO is connected, then the audio is being transferred to the Hands-Free device. If SCO is disconnected, then the audio is being played on the local device.

Syntax

```
Int32 BLUETOOTH_GetSCOConnectionState(ref BD_ADDR_t BD_ADDR, ref UInt32 ConnectionState);
```

Parameters

BD_ADDR

The Bluetooth address that BTEplorer has a Hands-Free connection with.

ConnectionState

Specifies the current SCO connection state if the function returns BT_ERROR_SUCCESS.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values: BT_ERROR, BT_ERROR_INVALID_PARAMETER, BT_ERROR_MSG_SEND

Remarks

None

See Also

SetSCOConnectionState

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_GetSCOConnectionState(BD_ADDR_t *BD_ADDR, DWord_t *ConnectionState)

Example

None

4.8.24 GetSecurityMode

Description

For Bluetooth 2.0 (or older) devices, this function is used to retrieve the current security mode. The security mode identifies whether Bluetooth 2.0-style "Mode 3 Security" should be used for authentication and encryption.

The current security mode will be one of the following: #BT_SECURITYMODE_NONE, BT_SECURITYMODE_AUTHENTICATE, BT_SECURITYMODE_AUTHENTICATE_AND_ENCRYPT

Syntax

```
Int32 BLUETOOTH_GetSecurityMode(ref BT_Security_Mode_Type SecurityMode);
```

Parameters

SecurityMode

A pointer to a BT_Security_Mode_Type that will store the current security mode.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_INVALID_PARAMETER, BT_ERROR_MSG_SEND

Remarks

None

See Also

SetSecurityMode

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_GetSecurityMode(BTP_Security_Mode_Type *SecurityMode)

Example

None

4.8.25 Open

Description

This function initializes the BTE Explorer API module, readying it for use by the application.

Syntax

```
bool BLUETOOTH_Open();
```

Parameters

None

Return Value

None

Remarks

None

See Also

Close

For C++

Library : BLUETOOTH.lib

Function : bool BLUETOOTH_Open(void)

Example

```
if (Bluetooth.Open())
{
    label_State_View.Text = "Open Success";
}
else
{
    label_State_View.Text = "Open Fail";
}
```

4.8.26 PerformAction

Description

This command is used to request that BTE Explorer take some particular action. This command can be used to launch BTE Explorer in some cases, or trigger specific actions within BTE Explorer in other cases. See the list of supported actions to determine the capabilities of this command.

Syntax

```
Int32 BLUETOOTH_PerformAction(UInt32 Action, UInt32 Param1, UInt32 Param2, UInt32 Return1, UInt32 Return2);
```

Parameters

Action

Determines the action that BTE Explorer is expected to take as a result of this command.

Param1

The first parameter for the selected action. **Parameters** are defined as needed for each action. Currently parameters are unused and should just be set to zero.

Param2

The second parameter for the selected action. **Parameters** are defined as needed for each action. Currently parameters are unused and should just be set to zero.

Return1

The first return value for the selected action. Currently return values are unused and can be set to NULL.

Return2

The second return value for the selected action. Currently return values are unused and can be set to NULL.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

None

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_PerformAction(DWord_t Action, DWord_t Param1, DWord_t Param2, DWord_t *Return1, DWord_t *Return2)

Example

```
hResult = Bluetooth.PerformAction(BlueToothNet.BT_ACTION_DELETE_ALL_DEVICES, 0, 0, 0, 0);
```

4.8.27 SetAuthenticationCallback

Description

This function registers an authentication callback with BTE Explorer. This function may be called multiple times to associate multiple devices with a callback, but each device may only be associated with a single callback. Calling this function also causes BTE Explorer to attempt to use automatic “JustWorks” pairing if both the local and remote device supports Secure Simple Pairing. “JustWorks” pairing results in a trusted relationship but does not require a PIN code or any user interaction. This callback will be called for “JustWorks” pairing, but the PIN provided will be ignored.

Syntax

```
Int32 BLUETOOTH_SetAuthenticationCallback(ref BD_ADDR_t BD_ADDR, ref  
BT_Authentication_Callback_t AuthenticationCallback, LPVOID CallbackParameter);
```

Parameters

BD_ADDR

Pointer to the unique identifier of the remote device associated with the callback.

AuthenticationCallback

The callback being registered.

CallbackParameter

User-defined parameter associated with the callback.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

None

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_SetAuthenticationCallback(const BD_ADDR_t *BD_ADDR,
BTP_Authentication_Callback_t AuthenticationCallback, LPVOID CallbackParameter)

Example

None

4.8.28 SetBluetoothState

Description

This function sets the Bluetooth device to either BT_DEVICE_STATE_ON or BT_DEVICE_STATE_OFF. Turning the device on and off will start and stop the BTE Explorer process.

Syntax

```
Int32 BLUETOOTH_SetBluetoothState(ref UInt32 BluetoothState);
```

Parameters

BluetoothState

Specifies the desired Bluetooth state.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_MSG_SEND

Remarks

None

See Also

GetBluetoothState

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_SetBluetoothState(DWord_t BluetoothState)

Example

None

4.8.29 SetConnectionCallback

Description

This function registers a connection callback with BTExplorer. This function may be called multiple times to associate multiple connections with a callback, or to associate multiple callbacks with a connection.

Syntax

```
Int32 BLUETOOTH_SetConnectionCallback(ref BT_Connection_ID ConnectionID, ref  
BT_Connection_Callback_t ConnectionCallback, ref LPVOID CallbackParameter);
```

Parameters

ConnectionID

Unique identifier for a connection which was previously defined by a call to CreateConnection.

ConnectionCallback

The callback being registered.

CallbackParameter

User-defined parameter associated with the callback.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values: BT_ERROR, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

None

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_SetConnectionCallback(BTP_Connection_ID ConnectionID, BTP_Connection_Callback_t ConnectionCallback, LPVOID CallbackParameter)

Example

None

4.8.30 SetIncomingPIN

Description

This sets or clears a PIN Code to be used for ANY incoming connections that require authentication. Calling this function also causes BTE Explorer to attempt to use automatic "JustWorks" pairing if both the local and remote device supports Secure Simple Pairing. "JustWorks" pairing results in a trusted relationship but does not require a PIN code or any user interaction. The PIN set is not used for "JustWorks" pairing, but can be used to enable automatic handling if desired.

Syntax

```
Int32 BLUETOOTH_SetIncomingPIN(ref BT_PIN_t NewPIN, ref BT_PIN_t OldPIN);
```

Parameters

IncomingPIN

A structure that identifies the new PIN and its length. Set the length of the PIN in this structure to 0 to clear the Incoming PIN.

OldPIN

An output structure that will have the length and value of the Old PIN.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

SetOutgoingPIN

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_SetIncomingPIN(BTP_PIN_t *NewPIN, BTP_PIN_t *OldPIN)

Example

None

4.8.31 SetOutgoingPIN

Description

This sets or clears a PIN Code to be used for ANY outgoing connections that require authentication. This takes precedence over any registered authentication callback. Calling this function also causes BTE Explorer to attempt to use automatic "JustWorks" pairing if both the local and remote device supports Secure Simple Pairing. "JustWorks" pairing results in a trusted relationship but does not require a PIN code or any user interaction. The PIN set is not used for "JustWorks" pairing, but can be used to enable automatic handling if desired.

Syntax

```
Int32 BLUETOOTH_SetOutgoingPIN(ref BT_PIN_t NewPIN, ref BT_PIN_t OldPIN);
```

Parameters

OutgoingPIN

A structure that identifies the new PIN and its length. Set the length of the PIN in this structure to 0 to clear the Outgoing PIN.

OldPIN

An output structure that will have the length and value of the Old PIN.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values: BT_ERROR, BT_ERROR_INVALID_PARAMETER

Remarks

None

See Also

SetIncomingPIN

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_SetOutgoingPIN(BT_PIN_t *NewPIN, BT_PIN_t *OldPIN)

Example

None

4.8.32 SetSCOConnectionState

Description

This function is used to set the SCO connection state for any open Hands-Free connection. Turning the SCO connection on and off will transfer audio to the Hands-Free device and return audio to the local device. The possible SCO connection states are BT_SCO_STATE_CONNECTED and BT_SCO_STATE_DISCONNECTED.

Syntax

```
Int32 BLUETOOTH_SetSCOConnectionState(ref BD_ADDR_t BD_ADDR, ref UInt32 ConnectionState);
```


Parameters

BD_ADDR

The Bluetooth address that BTE Explorer has a Hands-Free connection with.

ConnectionState

Specifies the desired SCO connection state.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values: BT_ERROR, BT_ERROR_INVALID_PARAMETER, BT_ERROR_MSG_SEND

Remarks

None

See Also

GetSCOConnectionState

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_SetSCOConnectionState(BD_ADDR_t *BD_ADDR, DWord_t ConnectionState)

Example

None

4.8.33 SetSecurityMode

Description

For Bluetooth 2.0 (or older) devices, this function is used to set the security mode for subsequent connections. The security mode dictates whether Bluetooth 2.0-style "Mode 3 Security" should be used for authentication and encryption.

One of three possible modes must be selected: None, Authenticate, or Authenticate and Encrypt.

Syntax

```
Int32 BLUETOOTH_SetSecurityMode(ref BT_Security_Mode_Type SecurityMode);
```

Parameters

SecurityMode

Specifies the desired security mode.

Return Value

BT_ERROR_SUCCESS if successful.

Error codes include the following values : BT_ERROR, BT_ERROR_INVALID_PARAMETER, BT_ERROR_MSG_SEND

Remarks

None

See Also

GetSecurityMode

For C++

Library : BLUETOOTH.lib

Function : HRESULT BLUETOOTH_SetSecurityMode(BTP_Security_Mode_Type SecurityMode)

Example

None

4.9 GPS

Status Return value & API Error Codes

Please refer to the below table for the status value definition.

Status Value Definition	Code	Meaning
TRUE	1	Success
FALSE	0	General Error

using
using HWND = System.IntPtr;
Event
public delegate void GetGpsInfoProc(GPS_ParseInfo Info);
Const
public const int WM_USER = 0x0400; public const int WM_USER_RECVDATA = WM_USER + 10000; public const int SATELLITE_COUNT = 12;
Enum
public enum GPS_MODULE_TYPE { MODULE_UNKNOWN, MODULE_UBLOX, MODULE_SIRF }; public enum GPS_START { GPS_HOT_START = 0, GPS_WARM_START, GPS_COLD_START }; public enum AGPS_DAY { AGPS_1DAY = 0, AGPS_2DAY, AGPS_3DAY, AGPS_5DAY, AGPS_7DAY, AGPS_10DAY, AGPS_14DAY };
Structure

```

[StructLayout(LayoutKind.Sequential)]
public struct GPS_Dop
{
    public double dPDop;        // Position dilution of precision
    public double dHDop;        // Horizontal dilution of precision
    public double dVDop;        // Vertical dilution of precision
};

[StructLayout(LayoutKind.Sequential)]
public struct GPS_Satellite
{
    public int nID;              // Satellite PRN number
    public int nElevation;       // Elevation, degrees
    public int nAzimuth;         // Azimuth, degrees
    public int nSNR;             // Signal to noise ration in dBHz
};

[StructLayout(LayoutKind.Sequential)]
public struct GPS_ParseInfo
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = SATELLITE_COUNT)]
    public GPS_Satellite[] mSat; // Satellite information structure
    public GPS_Dop mDop;         // Dilution of precision structure
    public int nSatInUse;        // Satellites being used (0 - 12)
    public int nSatNum;          // Satellites in view
    public bool bSatInfo;        // Check correct Satellites information
    public double dHeading;      // Course in degrees
    public double dVelocity;     // Speed over the ground in knots
    public bool bNorthLatitude;  // TRUE: North / FALSE: South
    public bool bEastLongitude;  // TRUE: East / FALSE: West
    public double dLatitude;     // Latitude: ddmm.mmmm
    public double dLongitude;    // Longitude: ddmm.mmmm
    public double dAltitude;     // Altitude in meters according to WGS-84 ellipsoid
    public double dUTCDate;      // UTC Date: DDMMYY
    public double dUTCTime;      // UTC Time: hhmmss.sss
    public int nPosFix;          // Position Fix: 0 = Invalid, 1 = Valid SPS, 2 = Valid DGPS, 3 = Valid PPS
    public int nGPSStatus;       // GPS Status: 1 = Valid, 0 = Invalid
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 128)]
    public string mNMEAMsg;     // Current NMEA message
};

```

Functions for GPS

Name	Description
AGPSDown	Downloads Almanac Data from AGPS Server.
AGPSRun	Applies Almanac Data to GPS Module.
Close	GPS can be closed through simply closing the opened Serial Port.
EnableStaticMode	Location determination when the receiver's antenna is presumed to be stationary on the earth.
ModuleRestart	The process of powering up a new GPS receiver for the first time and having it search out and lock onto the satellite by itself, without the benefit of initialization data.
Open	To open GPS, input the COM port number, Baud Rate for serial communication and Windows HWND for getting GPS data.

4.9.1 AGPSDown

Description

Downloads Almanac Data from AGPS Server.

Syntax

```
bool GPS_AGPSDown(AGPS_DAY day, string tzDestFilePath);
```

Parameters

day

Date of Almanac Data

tzDestFilePath

File path of Almanac Data

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

AGPSRun

For C++

Library : GPS.lib

Function : BOOL GPS_AGPSDown(AGPS_DAY day, const TCHAR* tzDestFilePath)

Example

```
Gps.AGPSDown(day, Full_Path);
```

4.9.2 AGPSRun

Description

Applies Almanac Data to GPS Module.

Syntax

```
bool GPS_AGPSRun(string tzInputFilePath);
```

Parameters

tzInputFilePath

Location that Almanac Data is stored.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

AGPSDown

For C++

Library : GPS.lib

Function : BOOL GPS_AGPSRun(const TCHAR* tzInputFilePath)

Example

```
Gps.AGPSRun(Path.GetFileName(Full_Path));
```

4.9.3 Close

Description

This function is responsible for cleaning up the module after it is no longer needed by the application.

Syntax

```
bool GPS_Close();
```

Parameters

None

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Open

For C++

Library : GPS.lib

Function : BOOL GPS_Close()

Example

```
if (Gps.Close())
{
    MessageBox.Show("Close");
}
else
{
    MessageBox.Show("Close Fail");
}
```

4.9.4 EnableStaticMode

Description

This function is location determination when the receiver's antenna is presumed to be stationary on the earth. This allows the use of various averaging techniques that improve accuracy by factors of over 1000.

Syntax

```
bool GPS_EnableStaticMode(bool bEnable);
```

Parameters

bEnable

TRUE : Static Mode Support

FALSE : Static Mode not supported

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : GPS.lib

Function : BOOL GPS_EnableStaticMode(BOOL bEnable)

Example

```
if (checkBox_Static.Checked == true)
{
    Gps.EnableStaticMode(true);
}
else if (checkBox_Static.Checked == false)
{
    Gps.EnableStaticMode(false);
}
```

4.9.5 ModuleRestart

Description

This function is the process of powering up a new GPS receiver for the first time and having it search out and lock onto the satellite by itself, without the benefit of initialization data.

Syntax


```
bool GPS_ModuleRestart(GPS_START StartType);
```

Parameters

StartType

GPS Restart Type.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : GPS.lib

Function : BOOL GPS_ModuleRestart(GPS_START StartType)

Example

```
switch (obj.SelectedIndex)
{
    case 0 :
        Gps.ModuleRestart(GPSNet.GPS_START.GPS_HOT_START);
        break;
    case 1 :
        Gps.ModuleRestart(GPSNet.GPS_START.GPS_WARM_START);
        break;
    case 2 :
        Gps.ModuleRestart(GPSNet.GPS_START.GPS_COLD_START);
        break;
    default:
        Gps.ModuleRestart(GPSNet.GPS_START.GPS_HOT_START);
        break;
}
```

4.9.6 Open

Description

This function initializes the GPS module, readying it for use by the application.

Syntax

```
bool GPS_Open(HWND hMainWnd, string tzComPort, [MarshalAs(UnmanagedType.FunctionPtr)]
GetGpsInfoProc Func);
```

Parameters

hMainWnd

Windows Handle of GPS Application.

tzComPort

COM Port for GPS Open.

GetFunc

Receiving GPS Data.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

Close

For C++

Library : GPS.lib

Function : BOOL GPS_Open(HWND hMainWnd, TCHAR* tzComPort, GetGpsInfoProc GetFunc = NULL)

Example

```
if (Gps.Open(MsgWin.Hwnd, strCom, fnGetGpsInfo))
{
    MessageBox.Show("Open");
}
else
{
    MessageBox.Show("Open Fail");
}
```

4.10 SYSTEM

Status Return value

Please refer to the below table for the status value definition.

Status Value Definition	Code	Meaning
TRUE	1	Success
FALSE	0	General Error

Enum
<pre>public enum DEVICE_INFO { DEVICE_M3SKY = 0, DEVICE_M3SKYSAM, DEVICE_M3ORANGE, DEVICE_M3SMARTCE, DEVICE_M3SMARTWM, DEVICE_M3T, DEVICE_M3ORANGEPLUS, DEVICE_Pos, DEVICE_MM3, DEVICE_M3ORANGEPLUS_CE };</pre>
Enum
<pre>public struct GSensor_PARAMS { public int GsensorEnable; public bool DisplayRotate; public bool MotionDisplayOff; public bool MotionDisplayWakeup; public bool MotionSleepOn; public bool MotionSleepWakeup; public bool MotionSleepOnPrevent; public bool MotionStateCheck; public bool DisplayMenuCheck; public bool SuspendMenuCheck; };</pre>

Functions for System

Name	Description
BacklightOn	Backlight On/Off
Cleanboot	Cleanboot
GetBacklightLevel	Gets the Backlight Level
GetBacklightTimeOut	Gets the BacklightTimeOut
GetBatteryLifePercent	Gets the Battery Life Percent
GetBatteryState	Gets the Battery State
GetBluetooth	Gets the Bluetooth
GetCpuClock	Gets the CPU Clock
GetDeviceInfo	Gets the Information of Device
GetGSensorValue	Gets the Gyro Sensor Value
GetOSVersionInfo	Gets the Information of OS Version
GetPhoneVolumeLevel	Gets the Phone Volume Level
GetPowerTimeOut	Gets the PowerTimeOut
GetSerialNumber	Gets the Serial Number
GetVolumeLevel	Gets the Volume Level
GetWlan	Gets the Wlan
KeypadLock	Kaypad Lock/Unlock
Reboot	Reboot
SetBacklightLevel	Sets the Backlight Level
SetBacklightTimeOut	Sets the BacklightTimeOut
SetBluetooth	Sets the Bluetooth On/Off
SetCpuClock	Sets the CPU Clock
SetGSensorValue	Sets the Gyro Sensor Value
SetPhoneVolumeLevel	Sets the Phone Volume Level
SetPowerTimeOut	Sets the PowerTimeOut
SetSleepMode	Sleep On
SetVolumeLevel	Sets the Volume Level
SetWlan	Sets the Wlan On/Off
Vibrate	Vibrate On/Off
VolumeMute	Volume Mute

[GetGuid](#)

Gets the Guid Number

4.10.1 BacklightOn

Description

Backlight On/Off.

Syntax

```
public bool BacklightOn(bool bOn);
```

Parameters

bOn

The state is either FALSE=Backlight Off, TRUE= Backlight On.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

This function cannot be used with M3 SMART CE.

See Also

None

For C++

Library : M3System.lib

Function : BOOL SYSTEM_BacklightOn(BOOL bOn)

Example

```
m_System.BacklightOn(true);
```

```
m_System.BacklightOn(false);
```

4.10.2 Cleanboot

Description

Cleanboot.

Syntax

```
public bool Cleanboot(bool bOn);
```

Parameters

bOn

The state is either TRUE=Cleanboot, FALSE=Reboot.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : BOOL SYSTEM_Cleanboot(BOOL bOn)

Example

```
bool m_bReboot = true;
m_System.Cleanboot(m_bReboot);
m_System.Reboot();
```

4.10.3 GetBacklightlevel

Description

Gets the Backlight Level.

Syntax

```
public int GetBacklightlevel();
```

Parameters

None

Return Value

Backlightlevel Value.

Remarks

None

See Also

SetBacklightlevel

For C++

Library : M3System.lib

Function : int SYSTEM_GetBacklightlevel()

Example

```
public int nBacklightlevel_Cur
nBacklightlevel_Cur = m_System.GetBacklightlevel();
TB_BACKLIGHTLEVEL.Value = nBacklightlevel_Cur;
switch(nBacklightlevel_Cur)
{
    case 8:
        LB_BACKLIGHTLEVEL.Text = "100%";
        break;
```

```

case 7:
    LB_BACKLIGHTLEVEL.Text = "90%";
    break;
case 6:
    LB_BACKLIGHTLEVEL.Text = "80%";
    break;
case 5:
    LB_BACKLIGHTLEVEL.Text = "70%";
    break;
case 4:
    LB_BACKLIGHTLEVEL.Text = "60%";
    break;
case 3:
    LB_BACKLIGHTLEVEL.Text = "40%";
    break;
case 2:
    LB_BACKLIGHTLEVEL.Text = "20%";
    break;
case 1:
    LB_BACKLIGHTLEVEL.Text = "10%";
    break;
case 0:
    LB_BACKLIGHTLEVEL.Text = "5%";
    break;
default:
    LB_BACKLIGHTLEVEL.Text = "5%";
    break;
}

```

4.10.4 GetBacklightTimeout

Description

Gets the BacklightTimeout.

Syntax

```
public int GetBackLightTimeout(int PowerType);
```


Parameters

type

The state is either 0=Battery Status, 1= AC Status.

Return Value

BacklightTimeOut Value.

Remarks

None

See Also

SetBackLightTimeOut

For C++

Library : M3System.lib

Function : BOOL SYSTEM_GetBacklightTimeOut(int PowerType)

Example

```
public int nBacklightTimeout_Cur;  
nBacklightTimeout_Cur = m_System.GetBackLightTimeOut(PowerType);  
CB_BACKLIGHTTIMEOUT.SelectedIndex = nBacklightTimeout_Cur;
```

4.10.5 GetBatteryLifePercent

Description

Gets the Battery Life Percent.

Syntax

```
public int GetBatteryLifePercent();
```

Parameters

None

Return Value

Current BatteryLife Value.

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : int SYSTEM_GetBatteryLifePercent()

Example

```
Public int bBatteryLife;  
bBatteryLife = m_System.GetBatteryLifePercent();
```

4.10.6 GetBatteryState

Description

Gets the Battery State.

Syntax

```
public bool GetBatteryState();
```

Parameters

None

Return Value

TRUE indicates AC Power. FALSE indicates Battery Power.

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : BOOL SYSTEM_GetBatteryState()

Example

```
if (m_System.GetBatteryState() == true)  
    LB_BATTERYSTATUS.Text = "Power : AC Power";  
else  
    LB_BATTERYSTATUS.Text = "Power : Battery Power";
```

4.10.7 GetBluetooth

Description

Gets the Bluetooth.

Syntax

```
public bool GetBluetooth();
```

Parameters

This function can only be used in M3 Mobile Device installing Windows Mobile OS.

Return Value

TRUE indicates Bluetooth ON. FALSE indicates Bluetooth OFF.

Remarks

None

See Also

SetBluetooth

For C++

Library : M3System.lib

Function : BOOL SYSTEM_GetBluetooth()

Example

```
if(m_System.GetBluetooth() == true)
    LB_BATTERYSTATUS.Text = " Bluetooth : ON";
Else
    LB_BATTERYSTATUS.Text = " Bluetooth : OFF";
```

4.10.8 GetCpuClock

Description

Gets the CPU Clock.

Syntax

```
public int GetCpuClock(out IntPtr getclock);
```

Parameters

getclock

Gets the Current CPU Clock Value.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

This function can't be used in M3 Smart, MM3 and M3 OX10.

See Also

SetCpuClock

For C++

Library : M3System.lib

Function : int SYSTEM_GetCpuClock(DWORD* getclock);

Example

```
public int nCurClock;
public IntPtr nCpuClock;
nCurClock = m_System.GetCpuClock(out nCpuClock);
```

4.10.9 GetDeviceInfo

Description

Gets the Information of Device

Syntax

```
public int GetDeviceInfo();
```

Parameters

getclock

Gets the Current CPU Clock Value.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetDeviceInfo

For C++

Library : M3System.lib

Function : int SYSTEM_GetCpuClock(DWORD* getclock);

Example

```
public int g_DeviceInfo;

g_DeviceInfo = m_System.GetDeviceInfo();

switch (g_DeviceInfo)
{
    case 0: //DEVICE_M3SKY
        M3Sky_Init();
        break;

    case 2: //DEVICE_M3ORANGE
        M3Orange_Init();
        break;

    case 4: //DEVICE_M3SMARTWM
        M3SmartWM_Init();
        break;

    case 5: //DEVICE_M3T
        M3T_Init();
```

```

        break;
    case 6: //DEVICE_M3ORANGEPLUS
        M3OrangePLUS_Init();
        break;
    case 7: //DEVICE_POS
        Pos_Init();
        break;
    case 8: //DEVICE_MM3
        MM3_Init();
        break;
}

```

4.10.10 GetGSensorValue

Description

Gets the Gyro Sensor Value

Syntax

```
public bool GetGSensorValue(out GSensor_PARAMS pGSensor_PARAMS);
```

Parameters

pGSensor_PARAMS

Gets the Gyro Sensor Value.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

This function can only be used with M3 SMART WM.

See Also

SetGSensorValue

For C++

Library : M3System.lib

Function : BOOL SYSTEM_GetGSensorValue(PGSensor_PARAMS pGSensor_Params);

Example

```

m_GSensor_Params = new GSenor_PARAMS();
m_System.GetGSensorValue(out m_GSensor_Params);
CB_MotionDisplayOff.Checked    = m_GSensor_Params.MotionDisplayOff;
CB_MotionDisplayWakeup.Checked = m_GSensor_Params.MotionDisplayWakeup;

```

```
CB_MotionSleepOn.Checked          = m_GSensor_Params.MotionSleepOn;  
CB_MotionSleepOnPrevent.Checked  = m_GSensor_Params.MotionSleepOnPrevent;  
CB_DisplayRotate.Checked         = m_GSensor_Params.DisplayRotate;
```

4.10.11 GetOSVersionInfo

Description

Gets the Information of OS version

Syntax

```
public string GetOSVersionInfo();
```

Parameters

None

Return Value

Return value is the information of OS Version

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : bool SYSTEM_GetOSVersionInfo(TCHAR* strVersionInfo);

Example

```
public string strOSVersion = m_System.GetOSVersionInfo();  
LB_OSVERSION.Text = "OS Version : " + strOSVersion;
```

4.10.12 GetPhoneVolumeLevel

Description

Gets the Phone Volume Level.

Syntax

```
public int GetVolumeLevel ();
```

Parameters

None

Return Value

The return value is PhoneVolumeLevel

Remarks

None

See Also

SetVolumeLevel

For C++

Library : M3System.lib

Function : int SYSTEM_GetVolumeLevel()

Example

```
Public int nVolumeLevel_Cur;
nVolumeLevel_Cur = GetVolumeLevel();
switch (nVolumeLevel_Cur)
{
case 0:
    nVolumeLevel_Cur = 5;
    break;
case 1:
    nVolumeLevel_Cur = 4;
    break;
case 2:
    nVolumeLevel_Cur = 3;
    break;
case 3:
    nVolumeLevel_Cur = 2;
    break;
case 4:
    nVolumeLevel_Cur = 1;
    break;
case 5:
    nVolumeLevel_Cur = 0;
    break;
default:
    nVolumeLevel_Cur = 3;
    break;
}
```

4.10.13 GetPowerTimeOut

Description

Gets the PowerTimeOut.

Syntax

```
public int GetPowerTimeOut(POWERTYPE type);
```

Parameters

type

The state is either 0=Battery Status, 1= AC Status.

Return Value

The return value is PowerTimeout.

Remarks

None

See Also

SetPowerTimeOut

For C++

Library : M3System.lib

Function : int SYSTEM_GetPowerTimeOut(int PowerType)

Example

```
PowerType = 0;           // Battery
nPowerTimeout_Cur = m_System.GetPowerTimeOut(PowerType);
CB_POWERTIMEOUT.SelectedIndex = nPowerTimeout_Cur;
```

4.10.14 GetSerialNumber

Description

Gets the Serial Number.

Syntax

```
public string GetSerialNumber();
```

Parameters

strSerial

Saving SerialNumber.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : BOOL SYSTEM_GetSerialNumber(TCHAR* szSerial);

Example

```
m_System = new SystemNet.SystemNet();  
g_DeviceInfo = m_System.GetDeviceInfo();  
strSerial = m_System.GetSerialNumber();  
LB_SERIAL.Text = "Serial Number : " + strSerial;
```

4.10.15 GetVolumeLevel

Description

Gets the Volume Level.

Syntax

```
public int GetVolumeLevel();
```

Parameters

None

Return Value

The return value is VolumeLevel.

Remarks

None

See Also

SetVolumeLevel

For C++

Library : M3System.lib

Function : int SYSTEM_GetVolumeLevel()

Example

```
nVolumeLevel_Cur = m_System.GetVolumeLevel();  
TB_MAINVOLUME.Value = nVolumeLevel_Cur;
```

4.10.16 GetWlan

Description

Gets the Wlan.

Syntax

```
public bool GetWlan();
```

Parameters

None

Return Value

The return value is Wlan Status.

Remarks

None

See Also

SetWlan

For C++

Library : M3System.lib

Function : bool SYSTEM_GetWlan()

Example

```
public bool bCurWlan;  
bCurWlan = m_System.GetWlan();  
if (bCurWlan == true)  
    RD_WLANON.Checked = true;  
else  
    RD_WLANOFF.Checked = true;
```

4.10.17 KeypadLock

Description

Keypad Lock/Unlock.

Syntax

```
public bool KeypadLock(bool bOn);
```

Parameters

bOn

The state is either TRUE=Lock, FALSE=Unlock.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : bool SYSTEM_KeypadLock(bool bOn)

Example

```
m_System.KeypadLock(TRUE);  
m_System.KeypadLock(FALSE);
```

4.10.18 Reboot

Description

Reboot.

Syntax

```
public void Reboot();
```

Parameters

None

Return Value

None

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : void SYSTEM_Reboot()

Example

```
m_System.Reboot();
```

4.10.19 SetBacklightlevel

Description

Sets the Backlight Level.

Syntax

```
public void SetBacklightlevel(DWORD m_dwBright);
```

Parameters

m_dwBright

Sets the Backlight Level.

Return Value

None

Remarks

None

See Also

GetBacklightlevel

For C++

Library : M3System.lib

Function : void SYSTEM_SetBacklightlevel(int m_nBright)

Example

```
m_System.SetBacklightlevel(1);
```

4.10.20 SetBacklightTimeOut

Description

Sets the BacklightTimeOut.

Syntax

```
public bool SetBackLightTimeOut(POWERTYPE type, bool bCheck, int nTimeOut);
```

Parameters

type

The state is either 0=Battery Status, 1= AC Status.

bCheck

The state is either FALSE=Not Set Time, TRUE= Set Time.

nTimeOut

Sets the Timeout Value.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetBackLightTimeOut

For C++

Library : M3System.lib

Function : bool SYSTEM_SetBackLightTimeOut(int PowerType, bool bCheck, int nTimeOut)

Example

```
Public int nBright;  
Public int PowerType;  
m_System.SetBackLightTimeOut(PowerType, fasle, nBright);
```

4.10.21 SetBluetooth

Description

Sets the Bluetooth On/Off.

Syntax

```
public bool SetBluetooth(bool bOn);
```

Parameters

bOn

The state is either 0=Bluetooth Off, 1= Bluetooth On.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

This function can only be used in M3 Mobile Device installing Windows Mobile OS.

See Also

GetBluetooth

For C++

Library : M3System.lib

Function : bool SYSTEM_SetBluetooth(bool bOn)

Example

```
m_System.SetBluetooth(true);  
m_System.SetBluetooth(false);
```

4.10.22 SetCpuClock

Description

Sets the CPU Clock.

Syntax

```
public bool SetCpuClock(int cpu);
```

Parameters

cpu

Sets the CPU Clock.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

This function can't be used in M3 Smart, MM3 and M3 OX10.

See Also

GetCpuClock

For C++

Library : M3System.lib

Function : bool SYSTEM_SetCpuClock(int cpu)

Example

```
m_System.SetCpuClock(0);
```

```
m_System.SetCpuClock(1);
```

```
m_System.SetCpuClock(2);
```

4.10.23 SetGSensorValue

Description

Sets the Gyro Sensor Value

Syntax

```
public bool SetGSensorValue(ref GSensor_PARAMS pGSensor_PARAMS);
```

Parameters

pGSensor_PARAMS

Sets the Gyro Sensor Value

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

This function can only be used with M3 SMART WM.

See Also

GetGSensorValue

For C++

Library : M3System.lib

Function : bool SYSTEM_SetGSensorValue(PGSensor_PARAMS pGSensor_Params)

Example

```
m_GSensor_Params = new GSenor_PARAMS();
```

```
m_GSensor_Params.MotionDisplayOff = CB_MotionDisplayOff.Checked;
```

```
m_GSensor_Params.MotionDisplayWakeup = CB_MotionDisplayWakeup.Checked;  
m_GSensor_Params.MotionSleepOn       = CB_MotionSleepOn.Checked;  
m_GSensor_Params.MotionSleepOnPrevent = CB_MotionSleepOnPrevent.Checked;  
m_GSensor_Params.DisplayRotate       = CB_DisplayRotate.Checked;
```

```
m_System.SetGSensorValue(ref m_GSensor_Params);
```

4.10.24 SetPhoneVolumeLevel

Description

Sets the Phone Volume Level.

Syntax

```
public void SetPhoneVolumeLevel(int m_nVolume);
```

Parameters

nVolume

Sets the PhoneVolume Level.

Return Value

None

Remarks

None

See Also

GetPhoneVolumeLevel

For C++

Library : M3System.lib

Function : void SYSTEM_SetPhoneVolumeLevel(int m_nVolume)

Example

```
m_System.SetPhoneVolumeLevel(5);
```

4.10.25 SetPowerTimeOut

Description

Sets the PowerTimeOut.

Syntax

```
public bool SetPowerTimeOut(int PowerType, int nTimeOut);
```

Parameters

powerType

The state is either 0=Battery Status, 1= AC Status.

nTimeout

Sets the Timeout Value.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetPowerTimeOut

For C++

Library : M3System.lib

Function : bool SYSTEM_SetPowerTimeOut(int PowerType, int nTimeout);

Example

```
Public int nBright;
```

```
Public int PowerType;
```

```
m_System.SetPowerTimeOut(PowerType, dwPower);
```

4.10.26 SetSleepMode

Description

Sleep On

Syntax

```
public void SetSleepMode();
```

Parameters

None

Return Value

None

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : void SYSTEM_SetSleepMode()

Example


```
m_System.SetSleepMode();
```

4.10.27 SetVolumeLevel

Description

Sets the Volume Level.

Syntax

```
public void SetVolumeLevel(int m_nVolume);
```

Parameters

m_nVolume

Sets the Volume Level.

Return Value

None

Remarks

None

See Also

GetVolumeLevel

For C++

Library : M3System.lib

Function : void SYSTEM_SetVolumeLevel(int m_nVolume)

Example

```
m_System.SetVolumeLevel(1);
```

4.10.28 SetWlan

Description

Sets the Wlan On/Off.

Syntax

```
public bool SetWlan(bool bOn);
```

Parameters

bOn

The state is either 0=Wlan Off, 1= Wlan On.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

GetWlan

For C++

Library : M3System.lib

Function : bool SYSTEM_SetWlan(bool bOn)

Example

```
m_System.SetWlan(true);
```

```
m_System.SetWlan(false);
```

4.10.29 Vibrate

Description.

Vibrate On/Off.

Syntax

```
public bool Vibrate(bool bOn);
```

Parameters

bOn

The state is either 0=Vibrate Off, 1= Vibrate On.

Return Value

Nonzero indicates success. Zero indicates failure.

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : bool SYSTEM_Vibrate(bool bOn)

Example

```
m_System.Vibrate(true);
```

```
m_System.Vibrate(false);
```

4.10.30 VolumeMute

Description

Volume Mute.

Syntax

```
public void VolumeMute();
```

Parameters

None

Return Value

None

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : void SYSTEM_VolumeMute()

Example

```
m_System.VolumeMute();
```

4.10.31 GetGuid

Description

Gets the Guid Number.

Syntax

```
public string GetGuid();
```

Parameters

None

Return Value

Guid Value.

Remarks

None

See Also

None

For C++

Library : M3System.lib

Function : BOOL SYSTEM_GetGuid(TCHAR* szSerial)

Example

```
String strGuid;
```

```
strGuid = m_System.GetSerialNumber();
```

```
LB_SERIAL.Text = "Serial Number : " + strGuid;
```

5 Demo Manual

This chapter is for Demo manuals for WLAN, Bluetooth and System. Please refer to the "Application Manual" for other application's demo manuals.

5.1 WLAN

Product	Model	Type	OS	Note
ALL	ALL	Summit	WM 6.1/6.5 CE 5.0	Applicable in Summit module, if 3rd party config is set

Precautions

- WLANTest.exe is usually located at \Flash Disk\WLAN.
- WLANTest.exe can be used on devices with Summit WLAN Module.
To use WlanTray instead of SCU (Summit Utility Client), the Active Profile on SCU must be set to ThirdPartyConfig.

1. First view of the program – AP List View



SSID : Service Set Identifier.

RSSI : Received Signal Strength Indication.

Security

true : Encryption

false : Open.

Power OFF : WLAN tern On/off.

Search : Scan available network.

Connect : Connect to selected network.

2. Connecting View for Encryption Type.

WLAN

Connect Info

SSID 9F_Motorola_30

RSSI -65

Encryption WEP

Password *****

Confirm *****

Connect to this AP?

Yes No

Connect Info : display View current status of WLAN.

Encryption : Access to an AP with stored settings.

Password : Write password.

Confirm : Confirm password.

Yes : Connect to apply AP info.

No : Cance..

3. State View

WLAN

ITEM	VALUE
Status	ASSOCIATED
SSID	9F_Motorola_30
Client IP	0.0.0.0
Client Mac	00:17:23:A0:99:BE
AP IP	0.0.0.0
AP Mac	00:15:70:E0:79:10
Beacon	100
DTIM	10
Channel	11
Bit Rate	1 Mbps
TX Power	63 mW

-62 dBm

AP List State Profile About

Status : Current status of WLAN

SSID : Name of connected SSID

Client IP

Client Mac

AP IP

AP Mac

Beacon : Shows periodic time of broad cast signal(beacon) that received from AP as Ksec.

DTIM : Shows how often the AP includes the DTIM(Delivery Traffic Indication Message) when sending signals. If the DTIM is 3, it means that DTIM is included in every third broadcast signal(beacon).

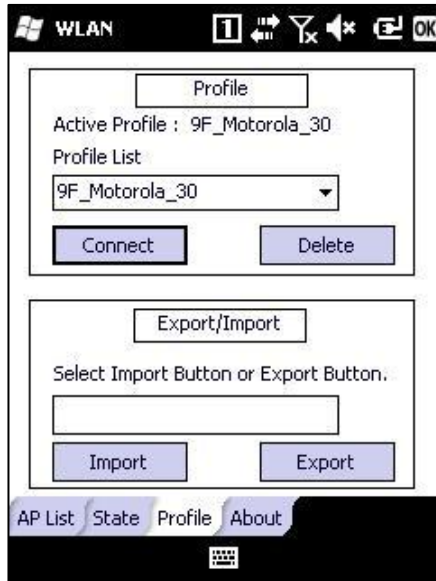
Channel : Connects to channel that AP sets and mostly channel 9 or 11 is used.

Bit Rate : Number of bits proceeded per 1 second.

TX Power

RSSI : Signal Strength,

4. Profile View



Active Profile : Profile of currently connected WLAN.

Profile List : List of stored profiles. Profile is produced automatically if it is connected at least once.

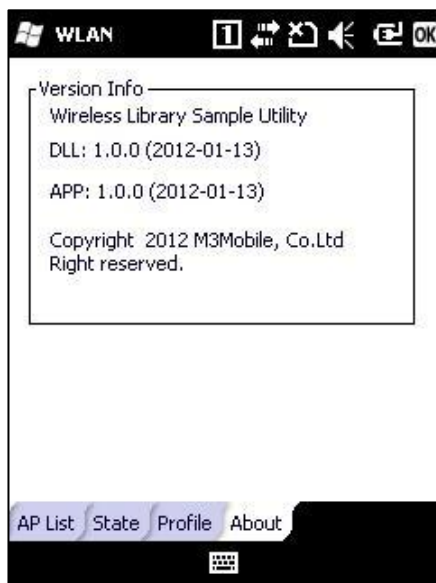
Connect : Connects to selected profile.

Delete : Deletes selected profile

Import : Restores the WLAN connection configuration.

Export : Back up of the WLAN connection configuration.

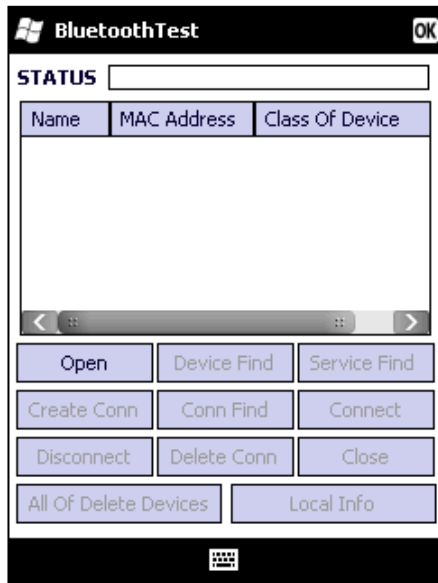
5. About View



Display Version Info

5.2 BLUETOOTH

1. Main Page



STATUS : Status of Bluetooth.

Open : Opens Bluetooth COM Port.

Device Find : Finds Bluetooth-connectable device.

Service Find : Finds service that Bluetooth can be connected.

Create Conn : Makes a Connection to connect.

Conn Find : Finds a Connection.

Connect : Connects Bluetooth devices to each other.

Disconnect : Disconnects the connected device.

Delete Conn : Deletes the Connection.

Close : Closes the Bluetooth COM Port.

All of Delete Devices : Deletes all searched devices.

Local Info : Provides the device information that runs program.



■ This demo program can be applied to upper level from StonestreetOne BTE Explorer Version 2.1.1 and Build Version 27460.

■ Performance procedure

1) If there is no Connection...

Open -> Device Find -> Service Find -> Create Connection -> Connection Find -> Connect -> Disconnect -> (Delete Connection) -> Close

2) If there is Connection...

Open -> Connection Find -> Connect -> Disconnect -> (Delete Connection) -> Close

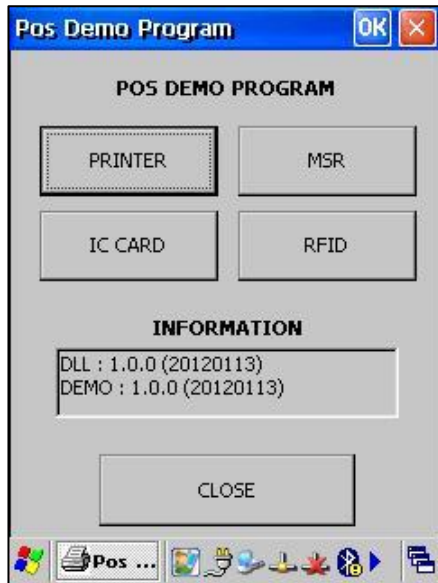
5.3 POS

Product	Model	Type	OS	Note
ALL	ALL	Software	ALL	Program for BT printer in M3 Mobile devices

Precautions

- PosTest.exe is usually located at \Flash Disk\POS.

1. First view of the program



PRINTER : Runs Printer Test Dialog.

MSR : Runs MSR Test Dialog.

IC CARD : Runs IC CARD Test Dialog.

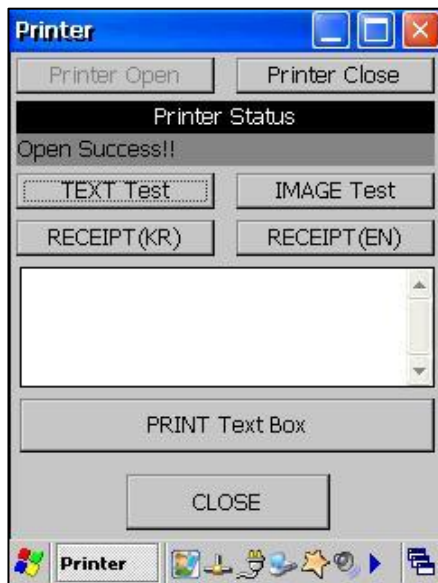
RFID : Runs RFID Test Dialog.

INFORMATION : DLL version info.

Ver : Test program version info

Close : Closes program

2. Printer Dialog.



Printer Open/Close : Printer Module On and Off.

Printer Status : Shows current status of printer.

TEXT Test : Font format can be adjusted but the font cannot.

IMAGE Test : Prints text in the text box.

RECEIPT : Prints receipt sample

PRINT Text Box : Prints text in the text box.

CLOSE : Returns to first Dialog.

3. MSR Dialog



MSR Open/Close : MSR module On and Off.

MSR Start/Stop : MSR Reading On and Off.

Card Status : Shows current status of MSR.

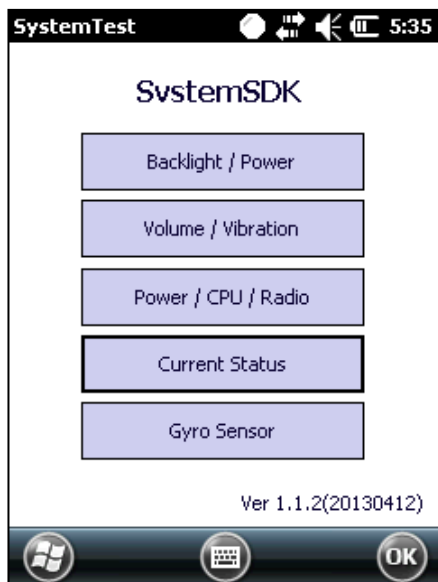
Close : Returns to first Dialog.

Card, Valid : No showing if Track 2 is not read.

5.4 SYSTEM

Product	Model	Type	OS	Note
M3 SKY	MC-7XX0S		WM 6.1/6.5	Summit Only
M3 T	MC-6700S/H	S/W Decoder	CE 5.0	
M3 SMART	M3 Smart	S/W, H/W Decoder	WM 6.5	Gyro Sensor Only
MM3	MC-8000S	S/W Decoder	WM 6.1	
M3 ORANGE	M3 Orange	S/W Decoder	WM 6.5	
M3 ORANGE +	M3 Orange +		WM 6.5	
POS	MC-8800S		CE 5.0	

1. Main Page



Backlight / Power

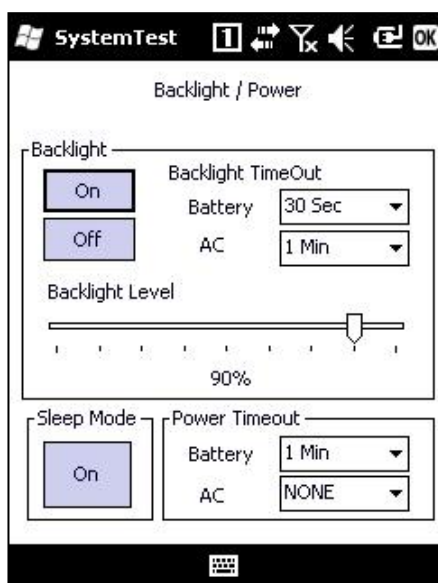
Volume / Vibration

Power / CPU / Radio

Current Status

Gyro Sensor

2. Backlight / Power



Backlight

On : Backlight On

Off : Backlight Off

Backlight TimeOut

Battery : Sets the Battery Backlight Timeout

AC : Sets the AC Backlight Timeout

Backlight Level : Backlight Level Control

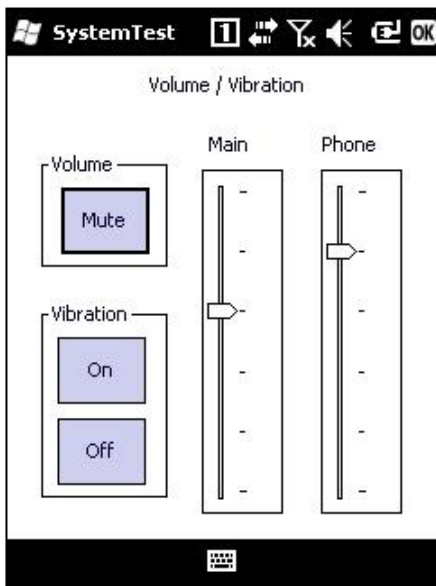
Sleep Mode : Sleep On

Power Timeout

Battery : Sets the Battery Backlight Timeout

AC : Sets the AC Backlight Timeout

3. Volume / Vibration



Volume

Mute : Volume Mute

Vibration

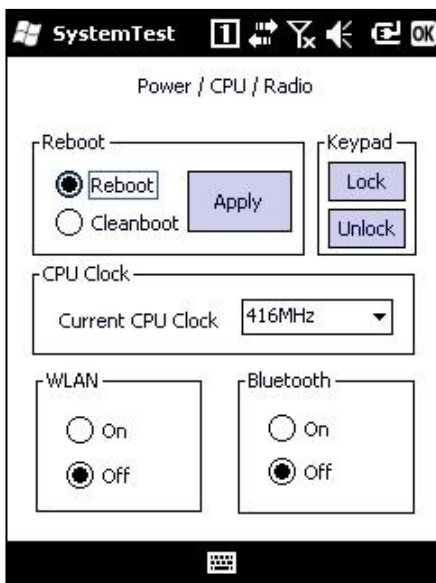
On : Vibration On

Off : Vibration Off

Main Volume : Main Volume Control

Phone Volume : Phone Volume Control

4. Power / CPU / Radio



Reboot

Reboot + Apply : Reboot

Cleanboot + Apply : Cleanboot

Keypad

Lock : Keypad Lock

Unlock : Keypad Unlock

CPU Clock : Sets the Current CPU Clock

WLAN

On : Wlan On

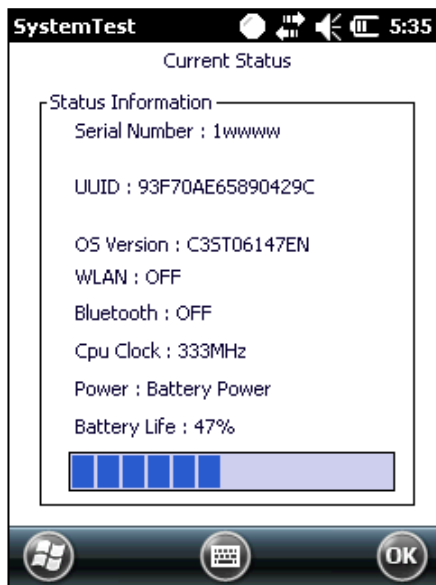
Off : Wlan Off

Bluetooth

On : Bluetooth On

Off : Bluetooth Off

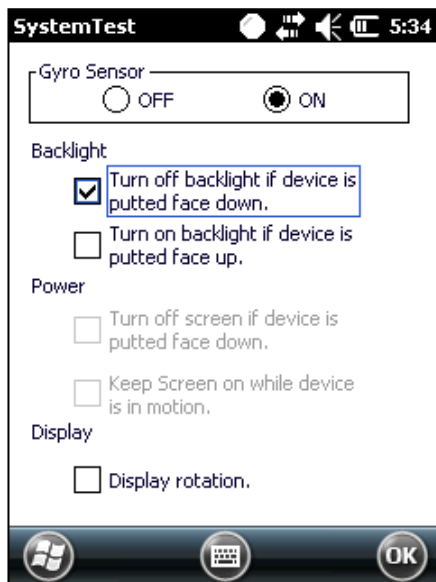
5. Current Status



Status Information

- Serial Number : Gets the Serial Number
- UUID : Gets the UUID
- OS Version : Gets the OS Version
- WLAN : Gets the Wlan Status
- Bluetooth : Gets the Bluetooth Status
- CPU Clock : Gets the Current CPU Clock
- Power : Gets the Power Status(Battery / AC)
- Battery Life : Gets the Current BatteryLife

6. Gyro Sensors



Gyro Sensor

- ON : Gyro Sensor On
- OFF : Gyro Sensor Off

Backlight

- Turn off backlight
- Turn on backlight

Power

- Turn off Screen
- Keep Screen

Display

- Display rotation

Services

If you experience any trouble while using our product, you can visit **M3 Service center** or send enquires to our **online support web page** (<http://itc.m3mobile.net>), we will do our best to solve your trouble as soon as we can.

M3 FAQ document can help you with troubleshooting.

For any enquires about business program, please contact program provider for faster service.

Contact Details

Headquarter

M3 Bldg., 735-45, Yeoksam-Dong, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82 2 574 0037 Fax: +82 2 558 1253

www.m3mobile.net, sales@m3mobile.co.kr

Factory / Service Center

Chun-ui Techno Park 201-610, 202, Chunui-Dong, WonMi-Gu, Buchoen, Gyeonggi-Do, 420-857, Korea
Tel: +82 32 623 0030, Fax: +82 32 623 0035

Online Support Web page

<http://itc.m3mobile.net>