

# Programmer's guide

UNI SDK Version 1.2.0 Manual for Microsoft® .NET CF

---

## **Abstract**

This document describes the methods and properties of the M3 MOBILE UNI SDK version 1.2.0. UNI SDK supports M3 Mobile products unless written separately.

Modules covered:

1D Scanner, 2D Imager, Long-range Scanner, Camera, RFID (LF/HF), UHF Gun, WLAN, Bluetooth, GPS, System SDK, POS peripheral (Printer, MSR, IC Card, RFID)

## 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 / 10
- 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

### UNI SDK Version 1.2.0    Date: March 2016

- Scanner (1D)
  - Added M3 UL10, M3 BK10 to supported model
  - Added SE965 Module of Zebra to supported model
  - Added N4313 Module of Honeywell to supported model
  - Minor bug fixes
- Imager (2D)
  - Added M3 UL10, M3 BK10 to supported model
  - Added N5600 Module of Honeywell to supported model
  - Added SE4500 Module of Zebra to supported model
  - Minor bug fixes
- LR Scanner (Long Range)
  - Added M3 UL10 CE to supported model
  - Minor bug fixes
- Wlan
  - Added M3 UL10, M3 BK10 to supported model
  - Added 45N Module of Wireless lan to supported model
  - Minor bug fixes
- System SDK
  - Added M3 UL10, M3 BK10 to supported model
  - Minor bug fixes
- Added UHF Gun SDK
- Remove the GPS SDK and recommend using the MS GPS API(Application programming interface)
  - Please contact our online support web page to receive our GPS 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) .....	3
3.2	IMAGER (2D) .....	5
3.3	LRSCANNER (Long-Range) .....	7
3.4	CAMERA (CE) .....	9
3.5	CAMERA (WM) .....	11
3.6	RFID (LF/HF) .....	13
3.7	RFID Gun SDK.....	15
3.8	WLAN .....	18
3.9	BLUETOOTH.....	21
3.10	RFID (HF) .....	32
3.11	SYSTEM SDK .....	37
4	References.....	41
4.1	SCANNER (1D) .....	42
4.1.1	CLOSE .....	49
4.1.2	GetAdaptiveScanning .....	49
4.1.3	GetCODABAR .....	50
4.1.4	GetCODE11 .....	51
4.1.5	GetCODE128.....	52
4.1.6	GetCODE25.....	52
4.1.7	GetCODE39 .....	53
4.1.8	GetCODE93.....	55
4.1.9	GetDeviceType .....	55
4.1.10	GetEAN13 .....	56
4.1.11	GetEAN8 .....	57
4.1.12	GetEngineType .....	57
4.1.13	GetGS1 .....	58
4.1.14	GetKOREAPOST .....	59
4.1.15	GetMSI.....	59
4.1.16	GetOption .....	60
4.1.17	GetScanData.....	61
4.1.18	GetSymbology.....	62
4.1.19	GetTELEPEN.....	63
4.1.20	GetUPCA .....	63
4.1.21	GetUPCE .....	64
4.1.22	GetVersionInfo .....	65

4.1.23	Open .....	66
4.1.24	Read .....	66
4.1.25	ReadCancel .....	67
4.1.26	RegHotKey.....	67
4.1.27	SetAdaptiveScanning .....	68
4.1.28	SetCODABAR .....	69
4.1.29	SetCODE11 .....	70
4.1.30	SetCODE128 .....	71
4.1.31	SetCODE25 .....	71
4.1.32	SetCODE39 .....	72
4.1.33	SetCODE93 .....	73
4.1.34	SetEAN13.....	74
4.1.35	SetEAN8.....	75
4.1.36	SetGS1 .....	76
4.1.37	SetKOREAPOST.....	76
4.1.38	SetMSI .....	77
4.1.39	SetOption.....	78
4.1.40	SetSymbology .....	79
4.1.41	SetSymbologyAll.....	80
4.1.42	SetSymbologyDefault.....	80
4.1.43	SetTELEPEN.....	81
4.1.44	SetUPCA.....	82
4.1.45	SetUPCE.....	82
4.1.46	UnRegHotKey .....	83
4.2	IMAGER (2D) .....	85
4.2.1	CAMCapture .....	99
4.2.2	CAMGetOption .....	99
4.2.3	CAMInit.....	100
4.2.4	CAMPreviewStart .....	101
4.2.5	CAMPreviewStop .....	101
4.2.6	CAMSetOption.....	102
4.2.7	CAMUnInit.....	103
4.2.8	Close .....	103
4.2.9	GetAZTEC .....	104
4.2.10	GetCenteringWindow.....	105
4.2.11	GetCHINAPOST.....	105
4.2.12	GetCODABAR.....	106
4.2.13	GetCODABLOCK .....	107
4.2.14	GetCODE11 .....	108
4.2.15	GetCODE128 .....	108

4.2.16	GetCODE16K .....	109
4.2.17	GetCODE39.....	110
4.2.18	GetCODE49.....	111
4.2.19	GetCODE93.....	111
4.2.20	GetCOMPOSITE .....	112
4.2.21	GetDATAMATRIX.....	113
4.2.22	GetDecOption.....	114
4.2.23	GetDeviceType .....	114
4.2.24	GetEAN13 .....	115
4.2.25	GetEAN8 .....	115
4.2.26	GetIATA25.....	116
4.2.27	GetImagerMode.....	117
4.2.28	GetINT25.....	117
4.2.29	GetKOREAPOST .....	118
4.2.30	GetMATRIX25 .....	119
4.2.31	GetMAXICODE .....	120
4.2.32	GetMICROPDF .....	120
4.2.33	GetMSI.....	121
4.2.34	GetOCR .....	122
4.2.35	GetOption .....	123
4.2.36	GetPDF417.....	124
4.2.37	GetPLANET.....	124
4.2.38	GetPLESSEY .....	125
4.2.39	GetPOSICODE.....	126
4.2.40	GetPOSTNET.....	126
4.2.41	GetQR .....	127
4.2.42	GetRSS .....	128
4.2.43	GetScanData.....	129
4.2.44	GetScanDataBytes .....	129
4.2.45	GetSTRT25 .....	130
4.2.46	GetSymbology.....	131
4.2.47	GetTELEPEN.....	133
4.2.48	GetUPCA .....	133
4.2.49	GetUPCE .....	134
4.2.50	GetVersionInfo .....	135
4.2.51	IQGetBarcodeData.....	135
4.2.52	IQGetOption .....	136
4.2.53	IQImagingStart.....	137
4.2.54	IQImagingStop.....	137
4.2.55	IQInit.....	138

4.2.56	IQSetOption.....	138
4.2.57	IQUnInit .....	139
4.2.58	Open .....	140
4.2.59	Read .....	140
4.2.60	ReadCancel .....	141
4.2.61	RegHotKey.....	142
4.2.62	SetAZTEC .....	143
4.2.63	SetCenteringWindow .....	143
4.2.64	SetCHINAPOST .....	144
4.2.65	SetCODABAR .....	145
4.2.66	SetCODABLOCK .....	146
4.2.67	SetCODE11 .....	146
4.2.68	SetCODE128 .....	147
4.2.69	SetCODE16K.....	148
4.2.70	SetCODE39 .....	148
4.2.71	SetCODE49 .....	149
4.2.72	SetCODE93 .....	150
4.2.73	SetCOMPOSITE.....	151
4.2.74	SetDATAMATRIX .....	152
4.2.75	SetDecOption .....	152
4.2.76	SetEAN13.....	153
4.2.77	SetEAN8.....	154
4.2.78	SetIATA25 .....	154
4.2.79	SetINT25.....	155
4.2.80	SetKOREAPOST.....	156
4.2.81	SetMATRIX25.....	156
4.2.82	SetMAXICODE.....	157
4.2.83	SetMICROPDF.....	158
4.2.84	SetMSI .....	159
4.2.85	SetOCR.....	159
4.2.86	SetOption.....	160
4.2.87	SetPDF417 .....	161
4.2.88	SetPLANET .....	162
4.2.89	SetPLESSEY .....	162
4.2.90	SetPOSICODE .....	163
4.2.91	SetPOSTNET .....	164
4.2.92	SetQR.....	165
4.2.93	SetRSS .....	165
4.2.94	SetSTR25.....	166
4.2.95	SetSymbology .....	167



4.2.96	SetSymbologyAll .....	169
4.2.97	SetSymbologyDefault .....	169
4.2.98	SetTELEPEN .....	170
4.2.99	SetUPCA .....	171
4.2.100	SetUPCE .....	171
4.2.101	UnRegHotKey .....	172
4.3	LRSCANNER (Long-Range) .....	174
4.3.1	Close .....	182
4.3.2	GetCODABAR .....	182
4.3.3	GetCODABLOCK .....	183
4.3.4	GetCODE11 .....	184
4.3.5	GetCODE128 .....	184
4.3.6	GetCODE39 .....	185
4.3.7	GetDeviceType .....	186
4.3.8	GetEAN13 .....	186
4.3.9	GetEAN8 .....	187
4.3.10	GetGS1COMPOSITE .....	188
4.3.11	GetGS1DATABAR .....	189
4.3.12	GetINT25 .....	189
4.3.13	GetMSI .....	190
4.3.14	GetOption .....	191
4.3.15	GetPLESSEY .....	192
4.3.16	GetPOSTNET .....	192
4.3.17	GetScanData .....	193
4.3.18	GetSTANDARD2OF5 .....	194
4.3.19	GetSymbology .....	195
4.3.20	GetTELEPEN .....	196
4.3.21	GetUPCA .....	197
4.3.22	GetUPCE .....	198
4.3.23	GetVersionInfo .....	198
4.3.24	Open .....	199
4.3.25	Read .....	199
4.3.26	ReadCancel .....	200
4.3.27	RegHotKey .....	201
4.3.28	SetCODABAR .....	202
4.3.29	SetCODABLOCK .....	202
4.3.30	SetCODE11 .....	203
4.3.31	SetCODE128 .....	204
4.3.32	SetCODE39 .....	205
4.3.33	SetEAN13 .....	205

4.3.34	SetEAN8.....	206
4.3.35	SetGS1COMPOSITE .....	207
4.3.36	SetGS1DATABAR .....	207
4.3.37	SetINT25.....	208
4.3.38	SetMSI .....	209
4.3.39	SetOption.....	210
4.3.40	SetPLESSEY .....	211
4.3.41	SetPOSTNET .....	211
4.3.42	SetSTANDARD2OF5 .....	212
4.3.43	SetSymbology .....	213
4.3.44	SetSymbologyAll.....	214
4.3.45	SetSymbologyDefault.....	215
4.3.46	SetTELEPEN.....	215
4.3.47	SetUPCA.....	216
4.3.48	SetUPCE.....	217
4.3.49	UnRegHotKey .....	218
4.4	CAMERA (CE) .....	219
4.4.1	Open.....	223
4.4.2	Close .....	223
4.4.3	PreviewStart.....	224
4.4.4	PreviewStop.....	224
4.4.5	Capture.....	225
4.4.6	GetLastSaveFilePath.....	226
4.4.7	AutoFocus.....	226
4.4.8	EnableAutoAF.....	227
4.4.9	Zoom .....	227
4.4.10	SetCameraOption .....	228
4.4.11	GetCameraOption .....	229
4.4.12	Brightness .....	230
4.4.13	FlashOn.....	230
4.4.14	FlashOff .....	231
4.4.15	RawData.....	231
4.4.16	InsertExifInformation.....	232
4.4.17	UseGPSExifData .....	233
4.4.18	GetScannerType .....	233
4.4.19	GetVersion.....	234
4.5	CAMERA (WM) .....	235
4.5.1	Open.....	239
4.5.2	Close .....	239
4.5.3	SetPreviewWindow .....	240

4.5.4	PreviewStart.....	241
4.5.5	PreviewStop.....	241
4.5.6	GetRegCapturePath.....	242
4.5.7	SetRegCapturePathEx.....	243
4.5.8	Capture.....	244
4.5.9	CaptureEx.....	245
4.5.10	VideoStart.....	245
4.5.11	VideoStop.....	246
4.5.12	VideoStopEx.....	247
4.5.13	GetFlashState.....	247
4.5.14	AlwaysFlash.....	248
4.5.15	CaptureFlash.....	248
4.5.16	AutoFocus.....	249
4.5.17	SetAfMode.....	250
4.5.18	Zoom.....	250
4.5.19	SetResolution.....	251
4.5.20	GetResolution.....	251
4.5.21	SetQuality.....	252
4.5.22	GetQuality.....	253
4.5.23	SetBrightness.....	253
4.5.24	GetBrightness.....	254
4.5.25	SetWhiteBalance.....	254
4.5.26	GetWhiteBalance.....	255
4.5.27	SetContrast.....	255
4.5.28	GetContrast.....	256
4.5.29	SetSharpness.....	257
4.5.30	GetSharpness.....	257
4.5.31	SetHistoEqual.....	258
4.5.32	GetHistoEqual.....	258
4.5.33	RegisterPreview.....	259
4.5.34	InsertExifInformation.....	260
4.5.35	UseGPSExifData.....	260
4.5.36	GetModelType.....	261
4.5.37	GetScannerType.....	262
4.5.38	GetVersion.....	263
4.6	RFID (LF/HF).....	264
4.6.1	Open.....	267
4.6.2	Close.....	268
4.6.3	PowerSupply.....	268
4.6.4	GetRadioType.....	269

4.6.5	SelectTag .....	270
4.6.6	HighSelectTag .....	270
4.6.7	LoginTag .....	271
4.6.8	ReadBlock .....	272
4.6.9	WriteBlock .....	273
4.6.10	ReadMultiBlock .....	273
4.6.11	WriteMultiBlock .....	274
4.6.12	SetAntenna .....	275
4.6.13	GetVersion .....	276
4.6.14	EnableMultiTag .....	277
4.6.15	SendReadMultiTag .....	278
4.6.16	SendTransferCommand .....	278
4.6.17	SendContinuousRead .....	279
4.6.18	SendCommand .....	280
4.6.19	SendCommandGetData .....	280
4.6.20	GetData .....	281
4.6.21	CheckResult .....	282
4.6.22	SoundPlay .....	283
4.6.23	SetTagType .....	284
4.6.24	GetTagType .....	284
4.6.25	GetTagTypeToString .....	285
4.6.26	TagItInventory .....	285
4.6.27	TagItReadBlock .....	286
4.6.28	TagItWriteBlock .....	287
4.7	UHF GUN READER .....	288
4.7.1	GetError .....	294
4.7.2	IsReady .....	294
4.7.3	Init .....	295
4.7.4	Close .....	296
4.7.5	Inventory .....	297
4.7.6	InventoryStop .....	297
4.7.7	Read .....	298
4.7.8	Write .....	299
4.7.9	Lock .....	300
4.7.10	Kill .....	301
4.7.11	GetData .....	302
4.7.12	SetRegionFrequency .....	303
4.7.13	GetRegionalFrequency .....	304
4.7.14	ReadBattery .....	304
4.7.15	ReadBatteryStatus .....	305

4.7.16	Version.....	305
4.8	WLAN .....	307
4.8.1	ActivateConfig .....	310
4.8.2	Close .....	310
4.8.3	ConnectAP .....	311
4.8.4	ConnectAPEX .....	312
4.8.5	DeleteConfig.....	313
4.8.6	ExportConfig.....	314
4.8.7	ImportConfig.....	314
4.8.8	Init .....	315
4.8.9	GetAllConfigName .....	316
4.8.10	GetCurrentAPIInfo .....	316
4.8.11	GetBssidList .....	317
4.8.12	GetPowerStatus.....	318
4.8.13	PowerOn .....	318
4.8.14	PowerOff .....	319
4.9	BLUETOOTH.....	320
4.9.1	Close .....	335
4.9.2	Connect .....	335
4.9.3	CreateConnection .....	336
4.9.4	CreateConnectionEx .....	336
4.9.5	DeleteConnection .....	337
4.9.6	Disconnect .....	338
4.9.7	FindConnectionClose.....	338
4.9.8	FindDeviceClose .....	339
4.9.9	FindFirstConnection .....	340
4.9.10	FindFirstConnectionEx .....	340
4.9.11	FindFirstDevice .....	341
4.9.12	FindFirstDeviceEx .....	342
4.9.13	FindFirstService .....	343
4.9.14	FindFirstServiceEx.....	344
4.9.15	FindLocalDevice .....	345
4.9.16	FindNextConnection .....	345
4.9.17	FindNextConnectionEx .....	346
4.9.18	FindNextDevice .....	347
4.9.19	FindNextService .....	347
4.9.20	FindNextServiceEx .....	348
4.9.21	FindServiceClose .....	349
4.9.22	GetBluetoothState .....	350
4.9.23	GetSCOConnectionState .....	350

4.9.24	GetSecurityMode .....	351
4.9.25	Open .....	352
4.9.26	PerformAction .....	352
4.9.27	SetAuthenticationCallback .....	353
4.9.28	SetBluetoothState .....	354
4.9.29	SetConnectionCallback .....	355
4.9.30	SetIncomingPIN .....	356
4.9.31	SetOutgoingPIN .....	357
4.9.32	SetSCOConnectionState .....	357
4.9.33	SetSecurityMode .....	358
4.10	SYSTEM .....	360
4.10.1	BacklightOn .....	363
4.10.2	Cleanboot .....	363
4.10.3	GetBacklightLevel .....	364
4.10.4	GetBacklightTimeOut .....	365
4.10.5	GetBatteryLifePercent .....	366
4.10.6	GetBatteryState .....	367
4.10.7	GetBluetooth .....	367
4.10.8	GetCpuClock .....	368
4.10.9	GetDeviceInfo .....	369
4.10.10	GetGSensorValue .....	370
4.10.11	GetOSVersionInfo .....	371
4.10.12	GetPhoneVolumeLevel .....	371
4.10.13	GetPowerTimeOut .....	373
4.10.14	GetSerialNumber .....	373
4.10.15	GetVersionInfo .....	374
4.10.16	GetVolumeLevel .....	374
4.10.17	GetWlan .....	375
4.10.18	KeypadLock .....	376
4.10.19	Reboot .....	376
4.10.20	SetBacklightLevel .....	377
4.10.21	SetBacklightTimeOut .....	378
4.10.22	SetBluetooth .....	378
4.10.23	SetCpuClock .....	379
4.10.24	SetGSensorValue .....	380
4.10.25	SetPhoneVolumeLevel .....	380
4.10.26	SetPowerTimeOut .....	381
4.10.27	SetSleepMode .....	382
4.10.28	SetVolumeLevel .....	382
4.10.29	SetWlan .....	383

4.10.30	Vibrate.....	383
4.10.31	VolumeMute .....	384
4.10.32	GetGuid .....	385
5	Demo Manual .....	386
5.1	WLAN .....	386
5.2	BLUETOOTH.....	389
5.3	POS .....	390
5.4	SYSTEM .....	392

# 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. Need to have UGR Gun
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	1.0.2 1.3.1.2 1.0.6 (1009)	2016-11-30 2016-11-23 2016-11-30
IMAGER	ImagerTestNet.exe Imager. dll ImagerNet.dll	IMAGER_TEST	1.1.0 1.2.2.1 1.1.9	2014-01-15 2016-04-18 2015-03-17
LRSCANNER	LRScanTestNet.exe LRScanner.dll LRScannetNet.dll	LRSCANNER_TEST	1.0.1 1.0.4 1.0.1	2014-11-07 2016-11-28 2014-11-06
CAMERA_CE	CamNetTest.exe CAM.dll CamNet.dll	CAMERA_TEST_NET	1.0.1 1.5.3 1.0.1	2014-11-07 2016-11-28 2014-11-06
CAMERA_WM	CamTestNet.exe CAM.dll CamNet.dll	CAMERA_TEST	1.4.1 1.3.2 1.4.1	2016-12-16 2016-09-27 2016-02-06
RFID	RfidTestNet.exe RFID.dll RFIDNet.dll	RFID_TEST_NET	1.2.0 1.1.0	2016-08-04 2012-09-20
RFID UHF	RFID_UHF_Net.exe CAENRFIDLibraryPocketPC.dll	RFID_TEST_NET	1.1.0 3.2.1	2013-10-21
UHF GUN READER	UHF_Gun_Net_Test.exe RFID_UHF.dll RFID_UHF Net.dll	UHF_Gun_NET_ TEST	1.0.1	2014-04-30
WLAN	WlanTestNet.exe WLAN.dll WLANNET.dll	WLAN_TEST	1.0.2 1.2.2 1.0.2	2013-11-26 2016-09-28 2014-10-22
BLUETOOTH	BluetoothTestNet.exe BLUETOOTH.dll BluetoothNet.dll	BLUETOOTH_TEST	1.0.1 1.0.0 1.0.0	
SYSTEM	SystemTestNet.exe M3System.dll M3SystemNet.dll	SYSTEM_TEST	1.1.6 1.3.1 1.2.5	2015-02-05 2015-10-02 2016-09-27

### 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.

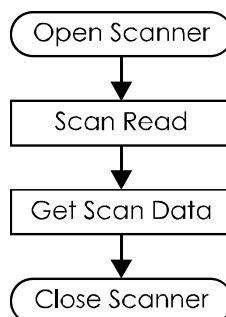
Section	Topic
SCANNER	Initialization ScanRead Get Window Message for getting Scan Data Close Scanner
IMAGER	Initialization ScanRead Get Window Message for getting Scan Data Close Scanner
CAMERA_CE	Open Close Capture Still Shot Preview Insert EXIF Information
CAMERA_WM	Open Close Capture Still Preview 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
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 BK10	No restriction
M3 MT10	No restriction
M3 OX10	No restriction
M3 PS10	No restriction
M3 ST10	No restriction
M3 UL10	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_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3ORANGEPLUS_CE) ||
(m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3UL10_CE) || (m_DeviceType ==
SCAN_DEVICE_TYPE.DEVICE_M3TPLUS_CE) || (m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3BLACK_CE) )
```

```
    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);
```

#### Scan Read

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

```
{
```

```
    m_Scanner.Read();
```

```
}
```

#### Get ScanData

```
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;
```

```
    }
```

```
}
```

#### Close Scanner

```
m_Scanner.UnRegHotKey(1);
```

```
for (int i = 0; i < 3; i++)
```

```
{
```

```
    m_bResult = m_Scanner.Close();
```

```
    Thread.Sleep(300);
```

```
    if (m_bResult == true)
```

```
        break;
```

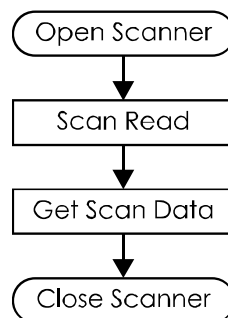
```
}
```

## 3.2 IMAGER (2D)

Supported PDA:

Brand	Restriction
M3 BK10	No restriction
M3 MT10	OS higher than 5050xx
M3 OX10	No restriction
M3 PS10	Not supported
M3 ST10 WM	No restriction
M3 ST10 CE	Not supported
M3 UL10	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_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3ORANGEPLUS_CE) ||
(m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3UL10_CE) || (m_DeviceType ==
SCAN_DEVICE_TYPE.DEVICE_M3TPLUS_CE) || (m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3BLACK_CE) )

    m_bWinCE = true;

//Imager 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);
m_Imager.SetSymbologyAll();

```

#### Scan Read

```

private void BN_SCAN_Click(object sender, EventArgs e)
{
    m_Imager.Read();
}

```

#### Get ScanData

```

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;
    }
}

```

#### Close Scanner

```

m_Imager.UnRegHotKey(1);
for (int i = 0; i < 3; i++)
{
    m_bResult = m_Imager.Close();
    Thread.Sleep(300);
    if (m_bResult == true)
        break;
}

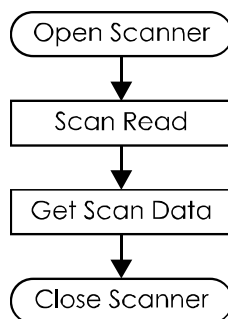
```

### 3.3 LRSCANNER (Long-Range)

Supported PDA:

Brand	Restriction
M3-BK10	Long-range is not supported
M3-MT10	Long-range is not supported
M3-OX10	Long-range is not supported
M3-PS10	Long-range is not supported
M3-ST10	Long-range is not supported
M3-UL10	No restriction

Basic long-range 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 LRScanEmul is running.
- II. LRScanner, H/W Decoder Scanner, takes longer than S/W Decoder when opening and closing scanner. It is recommended that program starts with opening LRScanner and ends with closing it.
- III. Its Virtual Key is VK\_F14 for Windows Mobile OS and VK\_22 for CE 6.0 OS.
- IV. Scan data can be simply obtained by adding Scanner event with APIs related HotKey

#### Open Scanner

```
private LRScanner m_LRScanner;
public const int m_nHotKeyWM = 125; // VK_F14(WM)
public const int m_nHotKeyCE = 133; // VK_F22(WinCE)
        :
m_LRScanner = new LRScanner();
m_LRScanner.LRScannerDataEvent += new LRScannerNet.LRScannerDataDelegate(OnScanRead);
//Scanner Open
m_LRScanner.Open();
m_DeviceType = m_LRScanner.GetDeviceType();
```

```

if (m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_M3UL10_CE)
{
    m_bWinCE = true;
    m_LRScanner.RegHotKey(1, m_nHotKeyCE, m_bSyncMode);
}
else if(m_DeviceType == SCAN_DEVICE_TYPE.DEVICE_MM3)
{
    m_bWinCE = false;
    m_LRScanner.RegHotKey(1, m_nHotKeyWM, m_bSyncMode);
}

```

#### Scan Read

```

private void BN_SCAN_Click(object sender, EventArgs e)
{
    m_LRScanner.Read();
}

```

#### Get ScanData

```

m_LRScanner.LRScannerDataEvent += new LRScannerNet.LRScannerDataDelegate(OnScanRead);
:
public void OnScanRead(object sender, LRScannerDataArgs e)
{
    if (e.ScanData != "")
    {
        LB_TYPE.Text = e.ScanType;
        TB_DATA.Text = e.ScanData;
    }
}

```

#### Close Scanner

```

m_LRScanner.UnRegHotKey(1);
for (int i = 0; i < 3; i++)
{
    m_bResult = m_LRScanner.Close();
    Thread.Sleep(300);
    if (m_bResult == true)
        break;
}

```

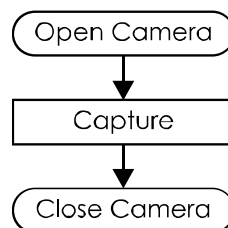


### 3.4 CAMERA (CE)

Supported PDA:

Brand	Restriction
M3-BK10-WM	Windows Mobile base brand
M3 BK10 CE	No restriction
M3 MT10	No restriction
M3-OX10-WM	Windows Mobile base brand
M3 OX10 CE	No restriction
M3 PS10	No restriction
M3-ST10-WM	Windows Mobile base brand
M3 ST10 CE	No restriction
M3 UL10	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);</pre>

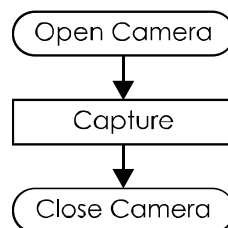
<pre>if (model == Cam.MODEL_TYPE.MODEL_UNKNOWN) {     MessageBox.Show("Camera can not open");     return; }</pre>
Capture Still
<pre>String strSavePath = ""; m_Cam.Capture(strSavePath);</pre>
Preview
<pre>bool bPreview = true; if(bPreview) {     m_Cam.PreviewStart(); } else {     m_Cam.PreviewStop(); }</pre>
Insert Exif Information
<pre>string pathTmp; m_Cam.GetLastSaveFilePath(out pathTmp); int cnt = 0; for (int i = 0; i &lt; 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 = "M3Green"; m_Cam.InsertExifInformation(FilePath, info);</pre>
Close Camra
<pre>M_Cam.Close();</pre>

### 3.5 CAMERA (WM)

Supported PDA:

Brand	Restriction
M3 BK10 WM	No restriction
M3 BK10 CE	Windows CE base brand
M3 MT10	Windows CE base brand
M3 OX10 WM	No restriction
M3 OX10 CE	Windows CE base brand
M3 PS10	Windows CE base brand
M3 ST10 WM	No restriction
M3 ST10 CE	Windows CE base brand
M3 UL10	Windows CE base brand

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

```
using CamNet;  
Cam m_Cam = new Cam();
```

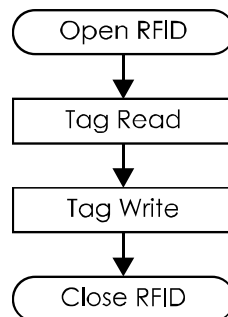
m_Cam.Open(this.Handle, Cam.CAMERA_MODE.STILL_MODE, Cam.VIDEO_TYPE.VIDEO_WMV);
<b>Capture Still</b>
char[] strOutFile = new char[260]; m_Cam.Capture("Flash Disk\\Camera\\", strOutFile, true);
<b>Preview</b>
void PreviewStart(bool bStart) { if(bStart) { m_Cam.PreviewStart(); } else { m_Cam.PreviewStop(); } } }
<b>Insert Exif Information</b>
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);
<b>Close Camra</b>
m_Cam.Close();

### 3.6 RFID (LF/HF)

Supported PDA:

Brand	Restriction
M3-BK10	Not supported
M3-MT10	Not supported
M3-O10	HF RFID modules is installed. UHF is not supported
M3-PS10	Not supported
M3-ST10	Not supported
M3-UL10	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

```
if(bAntennaOn == TRUE)  
{  
    RfidCommon.SetAntenna(TRUE);  
}  
else  
{  
    RfidCommon .SetAntenna(FALSE);  
}
```

#### Tag Select

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

#### Data Read

```
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);
```

#### Data Write

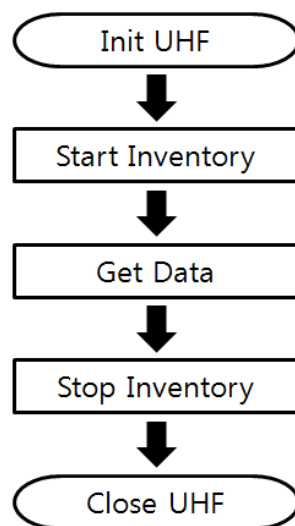
```
ngBuilder();  
RfidCommon.SelectTag(strSerial);  
const int nMaxData = 1024;  
StringBuilder strData = new StringBuilder(nMaxData);  
StringBuilder strOutData = new StringBuilder(nMaxData);  
m_Rfid.WriteBlock("01", "00112233", strOutData);
```

### 3.7 RFID Gun SDK

Supported PDA:

Brand	Restriction
M3-BK10	Not supported
M3-MT10	Not supported
M3-O10	Need to have UGR Gun
M3-PS10	Not supported
M3-ST10	Not supported
M3-UL10	Not supported

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);

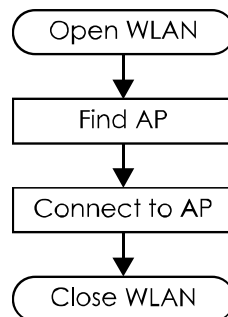
        return false;
    }
    label_Result.Text = "Close UHF";
    return true;
}
```

## 3.8 WLAN

Supported PDA:

Brand	Restriction
M3 BK10	WLAN must be SUMMIT
M3 MT10	WLAN must be SUMMIT
M3 OX10	WLAN must be SUMMIT
M3 PS10	WLAN must be SUMMIT
M3 ST10	WLAN must be SUMMIT
M3 UL10	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 result=Wlan.ConnectAPEX("SSID", "1234", 1, 2); if(result==false)     MessageBox.Show("Fail to ConnectAPEX()"); </pre>
<b>Delete Config</b>
<pre> bool result= Wlan.DeleteConfig("SSID"); if(!result)     MessageBox.Show("Activate config can not removed!!"); </pre>
<b>Change Config</b>
<pre> bool result=Wlan.ActivateConfig("SSID"); if(!result)     MessageBox.Show("Fail To ActivateConfig()"); </pre>
<b>Export Configuration</b>
<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>
<b>WLAN Power On</b>
<pre> bool result = Wlan.PowerOn(); if(!result)     MessageBox.Show("Fail To PowerOn()"); </pre>
<b>WLAN Power Off</b>
<pre> bool result = Wlan.PowerOff(); if(!result)     MessageBox.Show("Fail To PowerOff()"); </pre>
<b>Import Configuration</b>
<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

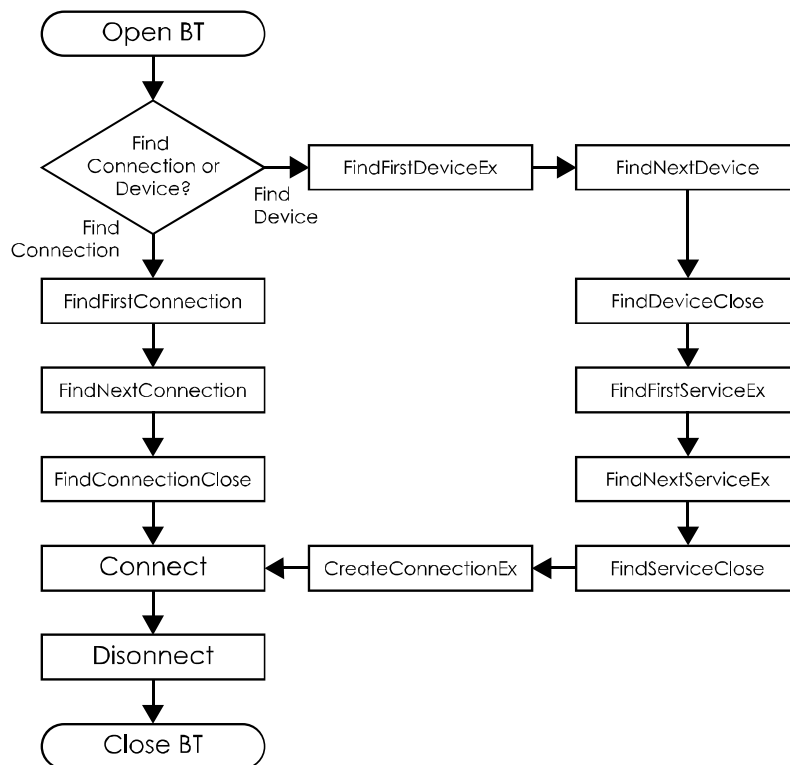
```
bool result=Wlan.Close();  
if(!result)  
    MessageBox.Show("Fail To Close (");
```

## 3.9 BLUETOOTH

Supported PDA:

Brand	Restriction
M3 BK10	BTE Explorer version 2.1.1 and Build version 27460 or higher
M3 MT10	BTE Explorer version 2.1.1 and Build version 27460 or higher
M3 OX10	BTE Explorer version 2.1.1 and Build version 27460 or higher
M3 PS10	BTE Explorer version 2.1.1 and Build version 27460 or higher
M3 ST10 WM	BTE Explorer version 2.1.1 and Build version 27460 or higher
M3 ST10 CE	Not supported
M3 UL10	BTE Explorer 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 BTE Explorer 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

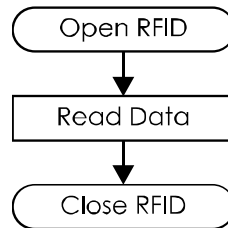
```
// Bluetooth Close
private void button_Close_Click(object sender, EventArgs e)
{
    listView_Info.Items.Clear();
    if (g_bBluetoothOpen == true)
    {
        Bluetooth.Close();
        label_State_View.Text = "Close Success";
        button_Open.Enabled = true;
        button_Device_Find.Enabled = 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;
}
```

### 3.10 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

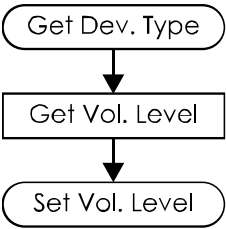
```
private void btn_rfid_close_Click(object sender, EventArgs e)
{
    if (g_OnOffStatus)
    {
        if (POS_RFID.RFID_Close())
        {
            label_rfid_status.Text = "RFID Close Success!!";
            g_OnOffStatus = false;
        }
        else
            label_rfid_status.Text = "RFID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

### 3.11 SYSTEM SDK

Supported PDA:

Brand	Restriction
M3 BK10	No restriction
M3 MT10	No restriction
M3 OX10	No restriction
M3 PS10	No restriction
M3 ST10	No restriction
M3 UL10	No restriction

Basic system SDK flow chart:



Get Device Type
<pre>m_System = new SystemNet.SystemNet(); m_DeviceInfo = m_System.GetDeviceInfo();  if (m_DeviceInfo == SystemNet.MODEL_TYPE.DEVICE_M3SMARTWM    m_DeviceInfo == SystemNet.MODEL_TYPE.DEVICE_M3BLACK) //DEVICE_M3SMARTWM: {     BN_GyroSensor.Visible = true; } else {     BN_GyroSensor.Visible = false; }</pre>
Get Volume Level
<pre>if (m_DeviceInfo == SystemNet.MODEL_TYPE.DEVICE_M3BLACK) {     TB_PHONEVOLUME.Maximum = 5;     TB_PHONEVOLUME.Minimum = 0;     nPhoneVolumeLevel_Cur = m_System.GetPhoneVolumeLevel();     TB_PHONEVOLUME.Value = nPhoneVolumeLevel_Cur; } else {</pre>

```

TB_PHONEVOLUME.Maximum = 5;
TB_PHONEVOLUME.Minimum = 0;
nPhoneVolumeLevel_Cur = m_System.GetPhoneVolumeLevel();

switch (nPhoneVolumeLevel_Cur)
{
    case 0:
        nPhoneVolumeLevel_Cur = 5;
        break;
    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 == SystemNet.MODEL_TYPE.DEVICE_M3T)
{
    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 <a href="#">IMAGER tutorial</a> for restrictions
LRSCANNER	LRScanner.h LRScanner.lib	MM3, M3 UL10 only
CAMERA CE	Cam.h Cam.lib	All M3 CE units * Please see <a href="#">CAMERA (CE) tutorial</a> for more details
CAMERA WM	Cam.h Cam.lib	All M3 WM units * Please see <a href="#">CAMERA (WM) tutorial</a> 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 <a href="#">BLUETOOTH tutorial</a> for more details
SYSTEM	System.h M3System.lib	All brands except M3 GREEN *Please see <a href="#">SYSTEM SDK tutorial</a> 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_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,     DEVICE_M3BLACK,     DEVICE_M3UL10_CE,     DEVICE_M3TPLUS_CE,     DEVICE_M3BLACK_CE,     DEVICE_UNKNOWN, };  public enum SCAN_ENGINE_TYPE {     OPTICON = 0,     SYMBOL,     INTERMEC,     SYMBOL_HW,     CINO,     ZEBRA_HW_SE965,     HONEYWELL_N4313,     UNKNOWN,     NOTYET };</pre>

```

public enum SCAN_SOUND
{
    SOUND_DEFAULT = 0,
    SOUND_BEEP,
    SOUND_NONE
};

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
<a href="#">Close</a>	Closes an open scanner
<a href="#">GetAdaptiveScanning</a>	Get the information of adaptive Scanning in SE 965 engine
<a href="#">GetCODABAR</a>	Gets the option of CODABAR Barcode
<a href="#">GetCODE11</a>	Gets the option of CODE11 Barcode
<a href="#">GetCODE128</a>	Gets the option of CODE128 Barcode
<a href="#">GetCODE25</a>	Gets the option of CODE25 Barcode
<a href="#">GetCODE39</a>	Gets the option of CODE39 Barcode
<a href="#">GetCODE93</a>	Gets the option of CODE93 Barcode
<a href="#">GetDeviceType</a>	Gets the type of device
<a href="#">GetEAN13</a>	Gets the option of EAN-13 Barcode
<a href="#">GetEAN8</a>	Gets the option of EAN-8 Barcode
<a href="#">GetEngineType</a>	Gets the type of scanner engine
<a href="#">GetGS1</a>	Gets the option of GS1 Barcode
<a href="#">GetKOREAPOST</a>	Gets the option of KOREAPOST Barcode
<a href="#">GetMSI</a>	Gets the option of MSI Barcode
<a href="#">GetOption</a>	Gets the option of Scanner
<a href="#">GetScanData</a>	Gets the ScanData which is read by scanner
<a href="#">GetSymbology</a>	Gets Enable/Disable of Symbologies
<a href="#">GetTELEPEN</a>	Gets the option of TELEPEN Barcode
<a href="#">GetUPCA</a>	Gets the option of UPC-A Barcode
<a href="#">GetUPCE</a>	Gets the option of UPC-E Barcode
<a href="#">GetVersionInfo</a>	Gets the information of scanner engine and dll version
<a href="#">Open</a>	Opens a scanner
<a href="#">Read</a>	Starts the beaming of scanner
<a href="#">ReadCancel</a>	Stops the beaming of scanner
<a href="#">RegHotKey</a>	Registers Scanner Button as a hot key
<a href="#">SetAdaptiveScanning</a>	Set the adaptive scanning function in SE 965 engine
<a href="#">SetCODABAR</a>	Sets the option of CODABAR Barcode
<a href="#">SetCODE11</a>	Sets the option of CODE11 Barcode
<a href="#">SetCODE128</a>	Sets the option of CODE128 Barcode

<a href="#"><u>SetCODE25</u></a>	Sets the option of CODE25 Barcode
<a href="#"><u>SetCODE39</u></a>	Sets the option of CODE39 Barcode
<a href="#"><u>SetCODE93</u></a>	Sets the option of CODE93 Barcode
<a href="#"><u>SetEAN13</u></a>	Sets the option of EAN-13 Barcode
<a href="#"><u>SetEAN8</u></a>	Sets the option of EAN-8 Barcode
<a href="#"><u>SetGS1</u></a>	Sets the option of GS1 Barcode
<a href="#"><u>SetKOREAPOST</u></a>	Sets the option of KOREAPOST Barcode
<a href="#"><u>SetMSI</u></a>	Sets the option of MSI Barcode
<a href="#"><u>SetOption</u></a>	Sets the option of Scanner
<a href="#"><u>SetSymbology</u></a>	Sets the Enable/Disable of Symbologies
<a href="#"><u>SetSymbologyAll</u></a>	Enables all of Symbologies
<a href="#"><u>SetSymbologyDefault</u></a>	Initializes all option of scanner
<a href="#"><u>SetTELEPEN</u></a>	Sets the option of TELEPEN Barcode
<a href="#"><u>SetUPCA</u></a>	Sets the option of UPC-A Barcode
<a href="#"><u>SetUPCE</u></a>	Sets the option of UPC-E Barcode
<a href="#"><u>UnRegHotKey</u></a>	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 GetAdaptiveScanning

#### Description

Gets the information of adaptive Scanning in SE 965 engine.

#### Syntax

```
public bool GetAdaptiveScanning ();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

Set AdaptiveScanning

**For C++**

Library : Scanner.lib

Function : BOOL SCAN\_SetAdaptiveScanning()

**Example**

```
bool bResult;  
bResult = SetAdaptiveScanning();
```

### 4.1.3 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();
```

#### 4.1.4 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();
```

### 4.1.5 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();
```

### 4.1.6 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();
```

### 4.1.7 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();
```

### 4.1.8 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();
```

### 4.1.9 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

### 4.1.10 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;
```



### 4.1.11 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(EAN8\_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;
```

### 4.1.12 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

### 4.1.13 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;
```

#### 4.1.14 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;
```

#### 4.1.15 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();
```

## 4.1.16 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;
```

### **4.1.17 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

### 4.1.18 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;
```

#### 4.1.19 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;
```

#### 4.1.20 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;
```

## 4.1.21 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;
```

## 4.1.22 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

## **4.1.23 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

## **4.1.24 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

### **4.1.25 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

### **4.1.26 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);
```

## **4.1.27 SetAdaptiveScanning**

### **Description**

Sets the adaptive scanning function in SE 965 engine.

**Syntax**

```
public bool SetAdaptiveScanning(bool bEnable);
```

**Parameters**

*bEnable*

Whether to enable adaptive scanning function or not.

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

GetAdaptiveScanning

**For C++**

Library : Scanner.lib

Function : BOOL SCAN\_SetAdaptiveScanning (bool bEnable)

**Example**

```
SetAdaptiveScanning (true);
```

## 4.1.28 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);
```

## 4.1.29 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);

```

### 4.1.30 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);

```

### 4.1.31 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);
```

## 4.1.32 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);
```

## 4.1.33 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**

GetCODE39

**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);
```

## 4.1.34 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(PEAN13\_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);
```

### 4.1.35 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(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.bEAN8_AS_EAN13 = CB_CONVERT.Checked;
```

```
m_Scanner.SetEAN8(ref m_Ean8);
```

### 4.1.36 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);
```

### 4.1.37 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);
```

## 4.1.38 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);

```

## 4.1.39 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);

```

#### 4.1.40 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);
```

#### 4.1.41 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

#### 4.1.42 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

## 4.1.43 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);
```

#### 4.1.44 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);
```

#### 4.1.45 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);
```

## 4.1.46 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
<pre>public event ImagerDataDelegate ImagerDataEvent; public event ImagerByteDataDelegate ImagerByteDataEvent;</pre>
Enum
<pre>public enum SCAN_DEVICE_TYPE {     DEVICE_M3SKY = 0,     DEVICE_M3SKYSAM = 1,     DEVICE_MM3 = 2,     DEVICE_M3ORANGE = 3,     DEVICE_M3SMART_CE = 4,     DEVICE_M3SMART_WM = 5,     DEVICE_M3GREEN = 6,     DEVICE_M3T = 7,     DEVICE_M3POS = 8,     DEVICE_M3ORANGEPLUS = 9,     DEVICE_M3ORANGEPLUS_CE = 10,     DEVICE_M3BLACK = 11,     DEVICE_M3UL10_CE = 12,     DEVICE_M3TPLUS_CE = 13,     DEVICE_M3BLACK_CE = 14,     DEVICE_UNKNOWN = 15, };  public enum SCAN_SOUND {     SOUND_DEFAULT = 0,     SOUND_BEEP,     SOUND_NONE };  public enum CODE39_FORMAT {     CODE39_STANDARD = 0,     CODE39_CODE32,</pre>

```

CODE39_TRIOPTIC
};

public enum OCR_MODE
{
    MODE_OCR_DISABLED = 0,
    MODE_OCR_A,
    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 bSTRT25, 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 nIType, 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
<a href="#">CAMCapture</a>	Captures the preview of imaging
<a href="#">CAMGetOption</a>	Gets the option of imaging
<a href="#">CAMInit</a>	Initializes imaging device
<a href="#">CAMPreviewStart</a>	Starts the preview of imaging
<a href="#">CAMPreviewStop</a>	Stops the preview of imaging
<a href="#">CAMSetOption</a>	Sets the option of imaging
<a href="#">CAMUninit</a>	Uninitializes imaging device
<a href="#">Close</a>	Closes an open imager
<a href="#">GetAZTEC</a>	Gets the option of AZTEC Barcode
<a href="#">GetCenteringWindow</a>	Gets Enable/Disable of centering window function
<a href="#">GetCHINAPOST</a>	Gets the option of CHINAPOST Barcode
<a href="#">GetCODABAR</a>	Gets the option of CODABAR Barcode
<a href="#">GetCODABLOCK</a>	Gets the option of CODABLOCK Barcode
<a href="#">GetCODE11</a>	Gets the option of CODE11 Barcode
<a href="#">GetCODE128</a>	Gets the option of CODE128 Barcode
<a href="#">GetCODE16K</a>	Gets the option of CODE16K Barcode
<a href="#">GetCODE39</a>	Gets the option of CODE39 Barcode
<a href="#">GetCODE49</a>	Gets the option of CODE49 Barcode
<a href="#">GetCODE93</a>	Gets the option of CODE93 Barcode
<a href="#">GetCOMPOSITE</a>	Gets the option of COMPOSITE Barcode
<a href="#">GetDATAMATRIX</a>	Gets the option of DATAMATRIX Barcode
<a href="#">GetDecOption</a>	Gets the option of Decoder
<a href="#">GetDeviceType</a>	Gets the type of device
<a href="#">GetEAN13</a>	Gets the option of EAN-13 Barcode
<a href="#">GetEAN8</a>	Gets the option of EAN-8 Barcode
<a href="#">GetIATA25</a>	Gets the option of IATA25 Barcode
<a href="#">GetImagerMode</a>	Gets the Imager Mode
<a href="#">GetINT25</a>	Gets the option of INT25 Barcode
<a href="#">GetKOREAPOST</a>	Gets the option of KOREAPOST Barcode
<a href="#">GetMATRIX25</a>	Gets the option of MATRIX25 Barcode

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



<a href="#">SetCenteringWindow</a>	Enables centering window function
<a href="#">SetCHINAPOST</a>	Sets the option of CHINAPOST Barcode
<a href="#">SetCODABAR</a>	Sets the option of CODABAR Barcode
<a href="#">SetCODABLOCK</a>	Sets the option of CODABLOCK Barcode
<a href="#">SetCODE11</a>	Sets the option of CODE11 Barcode
<a href="#">SetCODE128</a>	Sets the option of CODE128 Barcode
<a href="#">SetCODE16K</a>	Sets the option of CODE16K Barcode
<a href="#">SetCODE39</a>	Sets the option of CODE39 Barcode
<a href="#">SetCODE49</a>	Sets the option of CODE49 Barcode
<a href="#">SetCODE93</a>	Sets the option of CODE93 Barcode
<a href="#">SetCOMPOSITE</a>	Sets the option of COMPOSITE Barcode
<a href="#">SetDATAMATRIX</a>	Sets the option of DATAMATRIX Barcode
<a href="#">SetDecOption</a>	Sets the option of Decoder
<a href="#">SetEAN13</a>	Sets the option of EAN-13 Barcode
<a href="#">SetEAN8</a>	Sets the option of EAN-8 Barcode
<a href="#">SetIATA25</a>	Sets the option of IATA25 Barcode
<a href="#">SetINT25</a>	Sets the option of INT25 Barcode
<a href="#">SetKOREAPOST</a>	Sets the option of KOREAPOST Barcode
<a href="#">SetMATRIX25</a>	Sets the option of MATRIX25 Barcode
<a href="#">SetMAXICODE</a>	Sets the option of MAXICODE Barcode
<a href="#">SetMICROPDF</a>	Sets the option of MICROPDF Barcode
<a href="#">SetMSI</a>	Sets the option of MSI Barcode
<a href="#">SetOCR</a>	Sets the option of OCR Barcode
<a href="#">SetOption</a>	Sets the option of imager
<a href="#">SetPDF417</a>	Sets the option of PDF417 Barcode
<a href="#">SetPLANET</a>	Sets the option of PLANET Barcode
<a href="#">SetPLESSEY</a>	Sets the option of PLESSEY Barcode
<a href="#">SetPOSCODE</a>	Sets the option of POSICODE Barcode
<a href="#">SetPOSTNET</a>	Sets the option of POSTNET Barcode
<a href="#">SetQR</a>	Sets the option of QR Barcode
<a href="#">SetRSS</a>	Sets the option of RSS Barcode
<a href="#">SetSTRT25</a>	Sets the option of STRT25 Barcode

<a href="#">SetSymbology</a>	Sets the Enable/Disable of Symbologies
<a href="#">SetSymbologyAll</a>	Enables all of Symbologies
<a href="#">SetSymbologyDefault</a>	Initializes all option of scanner
<a href="#">SetTELEPEN</a>	Sets the option of TELEPEN Barcode
<a href="#">SetUPCA</a>	Sets the option of UPC-A Barcode
<a href="#">SetUPCE</a>	Sets the option of UPC-E Barcode
<a href="#">UnRegHotKey</a>	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 GetScanDataBytes

#### Description

Gets the ScanData which is read by imager.

#### Syntax

```
public bool GetScanDataBytes(ref byte[] pbyBarType, ref byte [] pbyBarData, ref int pnLength);
```

#### Parameters

*szBarType*

Pointer to barcode type.

*szBarData*

Pointer to barcode data.

*pnLength*

Pointer to Length of barcode data.

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

None

**For C++**

Library : Imager.lib

Function : `BOOL IMAGER_GetScanByteData(BYTE *pbyBarType, BYTE *pbyBarData, int *pnLength);`

**Example**

None

## 4.2.45 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();
```

## 4.2.46 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.bPOSICODE;
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;

## 4.2.47 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();
```

## 4.2.48 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;
```

## 4.2.49 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;
```

### **4.2.50 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

### **4.2.51 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

## 4.2.52 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;
```

## 4.2.53 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

## 4.2.54 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

## 4.2.55 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

## 4.2.56 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);
```

## 4.2.57 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

## 4.2.58 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

## 4.2.59 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

## 4.2.60 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

## 4.2.61 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);
```

## 4.2.62 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);
```

## 4.2.63 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);
```

## 4.2.64 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);
```

## **4.2.65 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);
```

## 4.2.66 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);
```

## 4.2.67 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);
```

## 4.2.68 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 CODE128common 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);
```

## 4.2.69 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);
```

## 4.2.70 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);
```

## 4.2.71 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);
```

## 4.2.72 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);
```

## **4.2.73 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);
```

## 4.2.74 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);
```

## 4.2.75 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

## 4.2.76 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);
```

## 4.2.77 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);
```

## 4.2.78 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_Iata25 = new IATA25_PARAMS();  
m_Iata25.bEnable = CB_ENABLE.Checked;  
m_Iata25.nMinLen = Convert.ToInt32(TB_MINLEN.Text);  
m_Iata25.nMaxLen = Convert.ToInt32(TB_MAXLEN.Text);  
m_Imager.SetIATA25(ref m_Iata25);
```

## 4.2.79 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 plnt25);`

**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);
```

## 4.2.80 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);
```

## 4.2.81 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);
```

## 4.2.82 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);
```

## 4.2.83 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);
```

## 4.2.84 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);
```

## 4.2.85 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);
```

## 4.2.86 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);
```

## 4.2.87 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);
```

## **4.2.88 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);
```

## **4.2.89 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);
```

## 4.2.90 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);
```

## 4.2.91 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);
```

## 4.2.92 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);
```

## 4.2.93 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);
```

## 4.2.94 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);
```

## **4.2.95 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);
```

## 4.2.96 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

## 4.2.97 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

## 4.2.98 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);
```

## 4.2.99 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);
```

## 4.2.100 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);
```

### **4.2.101 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 LRSCANNER (Long-Range)

#### 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 LRScannerDataDelegate LRScannerDataEvent;
Enum
<pre>public enum BARCODE_TYPE {     TYPE_AUSPOST = 0,     TYPE_AZTEC,     TYPE_BPO,     TYPE_CANADAPOST,     TYPE_CODABAR,     TYPE_CODABLOCK_A,     TYPE_CODABLOCK_F,     TYPE_CODE11,     TYPE_CODE39,     TYPE_CODE93,     TYPE_CODE128,     TYPE_GS1_128,     TYPE_DATAMATRIX,     TYPE_DUTCHPOST,     TYPE_UPCA,     TYPE_UPCE,     TYPE_EAN8,     TYPE_EAN13,     TYPE_ISXN,     TYPE_CC_AB,     TYPE_CC_C,     TYPE_GS1_DATABAR,     TYPE_GS1_LIM,     TYPE_GS1_EXP,     TYPE_INFOMAIL,     TYPE_INT25,     TYPE_JAPANPOST,     TYPE_MATRIX2OF5,     TYPE_MAXICODE,</pre>

```

TYPE_MSI,
TYPE_PDF417,
TYPE_MICRO_PDF417,
TYPE_PLANET,
TYPE_PLESSEY,
TYPE_POSTNET,
TYPE_QR,
TYPE_STANDARD2OF5,
TYPE_SWEDENPOST,
TYPE_TELEPEN,
TYPE_TLC39,
TYPE_UNKNOWN
};

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,
    DEVICE_M3BLACK,
    DEVICE_M3UL10_CE,
    DEVICE_UNKNOWN
};

```

## Structure

```

public struct DECODER
{
    public bool bAUSPOST;
    public bool bAZTEC;
    public bool bBPO;
    public bool bCANADAPOST;
    public bool bCODABAR;
    public bool bCODABLOCK;
    public bool bCODE11;
    public bool bCODE39;
    public bool bCODE93;
    public bool bCODE128;
    public bool bDATAMATRIX;
    public bool bDUTCHPOST;
}

```

```

public bool bUPCA;
public bool bUPCE;
public bool bEAN8;
public bool bEAN13;
public bool bGS1_COMPOSITE;
public bool bGS1_DATABAR;
public bool bINFOMAIL;
public bool bINT25;
public bool bJAPANPOST;
public bool bMATRIX2OF5;
public bool bMAXICODE;
public bool bMSI;
public bool bPDF417;
public bool bMICRO_PDF417;
public bool bPLANET;
public bool bPLESSEY;
public bool bPOSTNET;
public bool bQRCODE;
public bool bSTANDARD2OF5;
public bool bSWEDENPOST;
public bool bTELEPEN;
public bool bTLC39;

public DECODER(bool bAUSPOST, bool bAZTEC, bool bBPO, bool bCANADAPOST, bool bCODABAR, bool bCODABLOCK,
bool bCODE11, bool bCODE39, bool bCODE93, bool bCODE128,
    bool bDATAMATRIX, bool bDUTCHPOST, bool bUPCA, bool bUPCE, bool bEAN8, bool bEAN13, bool
bGS1_COMPOSITE, bool bGS1_DATABAR, bool bINFOMAIL, bool bINT25, bool bJAPANPOST,
    bool bMATRIX2OF5, bool bMAXICODE, bool bMSI, bool bPDF417, bool bMICRO_PDF417, bool bPLANET, bool
bPLESSEY, bool bPOSTNET, bool bQRCODE, bool bSTANDARD2OF5, bool bSWEDENPOST,
    bool bTELEPEN, bool bTLC39);
}

public struct DECODER_PARAMS
{
    public bool bContinueMode;
    public bool bXmitAimID;
    public bool bVibrate;
    public bool b1DDecodeMode;
    public bool bCenterDecode;
    public int nSound;
    public int nTimeOut;
    public int nSecurityLevel;

    public DECODER_PARAMS(bool bContinueMode, bool bXmitAimID, bool bVibrate, bool b1DDecodeMode, bool
bCenterDecode, int nSound, int nTimeOut, int nSecurityLevel);
}

public struct CODABAR_PARAMS
{
    public bool bEnable;
    public bool bXSS;

```



```

    public bool bCDV;
    public bool bXCD;
    public CODABAR_PARAMS(bool bEnable, bool bXSS, bool bCDV, bool bXCD);
}

public struct CODABLOCK_PARAMS
{
    public bool bEnable;
    public bool bCodaBlock_F;

    public CODABLOCK_PARAMS(bool bEnable, bool bCodaBlock_F);
}

public struct CODE11_PARAMS
{
    public bool bEnable;
    public bool bCDV;
    public bool bXCD;
    public CODE11_PARAMS(bool bEnable, bool bCDV, bool bXCD);
}

public struct CODE39_PARAMS
{
    public bool bEnable;
    public bool bCDV;
    public bool bXCD;
    public bool bFullASCII;
    public CODE39_PARAMS(bool bEnable, bool bCDV, bool bXCD, bool bFullASCII);
}

public struct CODE128_PARAMS
{
    public bool bEnable;
    public bool bGs1_128;
    public bool bIsbt128;
    public CODE128_PARAMS(bool bEnable, bool bGs1_128, bool bIsbt128);
}

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

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

```

```

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

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

public struct GS1COMPOSITE_PARAMS
{
    public bool bEnable;
    public bool bCC_C;
    public GS1COMPOSITE_PARAMS(bool bEnable, bool bCC_C);
}

public struct GS1DATABAR_PARAMS
{
    public bool bEnable;
    public bool bGS1LIM;
    public bool bGS1EXP;
    public GS1DATABAR_PARAMS(bool bEnable, bool bGS1LIM, bool bGS1EXP);
}

public struct INT25_PARAMS
{
    public bool bEnable;
    public bool bCDV;
    public bool bXCD;
    public INT25_PARAMS(bool bEnable, bool bCDV, bool bXCD);
}

public struct MSI_PARAMS
{
    public bool bEnable;
    public bool bCDV;
    public bool bXCD;
    public MSI_PARAMS(bool bEnable, bool bCDV, bool bXCD);
}

```

```
}  
public struct PLESSEY_PARAMS  
{  
    public bool bEnable;  
    public bool bXSS;  
    public bool bCDV;  
    public bool bXCD;  
    public PLESSEY_PARAMS(bool bEnable, bool bXSS, bool bCDV, bool bXCD);  
}  
public struct POSTNET_PARAMS  
{  
    public bool bEnable;  
    public bool bXCD;  
    public POSTNET_PARAMS(bool bEnable, bool bXCD);  
}  
public struct STANDARD2OF5_PARAMS  
{  
    public bool bEnable;  
    public bool bCDV;  
    public bool bXCD;  
    public STANDARD2OF5_PARAMS(bool bEnable, bool bCDV, bool bXCD);  
}  
public struct TELEPEN_PARAMS  
{  
    public bool bEnable;  
    public bool bNumeric;  
    public TELEPEN_PARAMS(bool bEnable, bool bNumeric);  
}
```

## Functions for LRScanner

Name	Description
<a href="#">Close</a>	Closes an open scanner
<a href="#">GetCODABAR</a>	Gets the option of CODABAR Barcode
<a href="#">GetCODABLOCK</a>	Gets the option of CODABLOCK Barcode
<a href="#">GetCODE11</a>	Gets the option of CODE11 Barcode
<a href="#">GetCODE128</a>	Gets the option of CODE128 Barcode
<a href="#">GetDeviceType</a>	Gets the type of device
<a href="#">GetDevice_CODE39</a>	Gets the option of CODE39 Barcode
<a href="#">GetEAN13</a>	Gets the option of EAN-13 Barcode
<a href="#">GetEAN8</a>	Gets the option of EAN-8 Barcode
<a href="#">GetGS1COMPOSITE</a>	Gets the option of GS1COMPOSITE Barcode
<a href="#">GetGS1DATABAR</a>	Gets the option of GS1DATABAR Barcode
<a href="#">GetINT25</a>	Gets the option of INT25 Barcode
<a href="#">GetMSI</a>	Gets the option of MSI Barcode
<a href="#">GetOption</a>	Gets the option of Scanner
<a href="#">GetPLESSEY</a>	Gets the option of PLESSEY Barcode
<a href="#">GetPOSTNET</a>	Gets the option of POSTNET Barcode
<a href="#">GetScanData</a>	Gets the ScanData which is read by scanner
<a href="#">GetSTANDARD2OF5</a>	Gets the option of STANDARD2OF5 Barcode
<a href="#">GetSymbology</a>	Gets Enable/Disable of Symbologies
<a href="#">GetTELEPEN</a>	Gets the option of TELEPEN Barcode
<a href="#">GetUPCA</a>	Gets the option of UPC-A Barcode
<a href="#">GetUPCE</a>	Gets the option of UPC-E Barcode
<a href="#">GetVersionInfo</a>	Gets the information of scanner engine and dll version
<a href="#">Open</a>	Opens a scanner
<a href="#">Read</a>	Starts the beaming of scanner
<a href="#">ReadCancel</a>	Stops the beaming of scanner
<a href="#">RegHotKey</a>	Registers Scanner Button as a hot key
<a href="#">SetCODABAR</a>	Gets the option of CODABAR Barcode
<a href="#">SetCODABLOCK</a>	Gets the option of CODABLOCK Barcode
<a href="#">SetCODE11</a>	Sets the option of CODE11 Barcode

<a href="#">SetCODE128</a>	Sets the option of CODE128 Barcode
<a href="#">SetCODE39</a>	Gets the option of CODE39 Barcode
<a href="#">SetEAN13</a>	Gets the option of EAN-13 Barcode
<a href="#">SetEAN8</a>	Gets the option of EAN-8 Barcode
<a href="#">SetGS1COMPOSITE</a>	Gets the option of GS1COMPOSITE Barcode
<a href="#">SetGS1DATABAR</a>	Gets the option of GS1DATABAR Barcode
<a href="#">SetINT25</a>	Gets the option of INT25 Barcode
<a href="#">SetMSI</a>	Sets the option of Scanner
<a href="#">SetOption</a>	Sets the option of Scanner
<a href="#">SetPLESSEY</a>	Gets the option of PLESSEY Barcode
<a href="#">SetPOSTNET</a>	Gets the option of POSTNET Barcode
<a href="#">SetSTANDARD2OF5</a>	Gets the option of STANDARD2OF5 Barcode
<a href="#">SetSymbology</a>	Sets the Enable/Disable of Symbologies
<a href="#">SetSymbologyAll</a>	Enables all of Symbologies
<a href="#">SetSymbologyDefault</a>	Initializes all option of scanner
<a href="#">SetTELEPEN</a>	Sets the option of TELEPEN Barcode
<a href="#">SetUPCA</a>	Sets the option of UPC-A Barcode
<a href="#">SetUPCE</a>	Sets the option of UPC-E Barcode
<a href="#">UnRegHotKey</a>	UnRegHotKey Free Scanner Button as a hot key

### 4.3.1 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 : LRScanner.lib

Function : BOOL LRSCAN\_Close()

#### Example

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

### 4.3.2 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetCODABAR(PCODABAR\_PARAMS pCodabar)

#### Example

```
private CODABAR_PARAMS m_Codabar = new CODABAR_PARAMS();  
m_LRScanner.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;
```

### 4.3.3 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetCODABLOCK(PCODABLOCK\_PARAMS pCodablock)

#### Example

```
private CODABLOCK_PARAMS m_Codablock = new CODABLOCK_PARAMS();
m_LRScanner.GetCODABLOCK(out m_Codablock);
CB_ENABLE_CODABLOCKA.Checked = m_Codablock.bEnable;
CB_ENABLE_CODABLOCKF.Checked = m_Codablock.bCodaBlock_F;
```

#### 4.3.4 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetCODE11(PCODE11\_PARAMS pCode11)

##### Example

```
private CODE11_PARAMS m_Code11 = new CODE11_PARAMS();
m_LRScanner.GetCODE11(out m_Code11);
CB_ENABLE.Checked = m_Code11.bEnable;
CB_CDV.Checked = m_Code11.bCDV;
if (m_Code11.bXCD == false)
    RD_CDV1.Checked = true;
else if (m_Code11.bXCD == true)
    RD_CDV2.Checked = true;
```

#### 4.3.5 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetCODE128(PCODE128\_PARAMS pCode128)

#### **Example**

```
private CODE128_PARAMS m_Code128 = new CODE128_PARAMS();  
m_LRScanner.GetCODE128(out m_Code128);  
CB_ENABLE.Checked = m_Code128.bEnable;  
CB_GS1128.Checked = m_Code128.bGs1_128;  
CB_ISBT128.Checked = m_Code128.bIsbt128;
```

### **4.3.6 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetCODE39(PCODE39\_PARAMS pCode39)

### Example

```
private CODE39_PARAMS m_Code39 = new CODE39_PARAMS();  
m_LRScanner.GetCODE39(out m_Code39);  
CB_ENABLE.Checked = m_Code39.bEnable;  
CB_CDV.Checked = m_Code39.bCDV;  
CB_XCD.Checked = m_Code39.bXCD;  
CB_FULLASCII.Checked = m_Code39.bFullASCII;
```

## 4.3.7 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

LRSCAN\_GetDeviceType is only used after LRSCAN\_OPEN.

### See Also

None

### For C++

Library : LRScanner.lib

Function : SCAN\_DEVICE\_TYPE LRSCAN\_GetDeviceType()

### Example

None

## 4.3.8 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetEAN13(PEAN13\_PARAMS pEan13)

#### **Example**

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

### **4.3.9 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetEAN8(PEAN8\_PARAMS pEan8)

#### Example

```
private EAN8_PARAMS m_Ean8 = new EAN8_PARAMS();  
m_LRScanner.GetEAN8(out m_Ean8);  
CB_ENABLE.Checked = m_Ean8.bEnable;  
CB_XCD.Checked = m_Ean8.bXCD;  
CB_EAN8TOEAN13.Checked = m_Ean8.bEAN8_AS_EAN13;
```

### 4.3.10 GetGS1COMPOSITE

#### Description

Gets the option of GS1COMPOSITE Barcode.

#### Syntax

```
public bool GetGS1COMPOSITE(out GS1COMPOSITE_PARAMS pGs1Composite);
```

#### Parameters

*pGs1Composite*

Pointer to a GS1COMPOSITE\_PARAMS structure to be filled in with the GS1COMPOSITE common parameters.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

SetGS1COMPOSITE

#### For C++

Library : LRScanner.lib

Function : BOOL LRSCAN\_GetGS1COMPOSITE(PGS1COMPOSITE\_PARAMS pGs1Composite)

#### Example

```
private GS1COMPOSITE_PARAMS m_Gs1composite = new GS1COMPOSITE_PARAMS();  
m_LRScanner.GetGS1COMPOSITE(out m_Gs1composite);  
CB_ENABLE.Checked = m_Gs1composite.bEnable;
```

CB\_ENABLE\_CC.Checked = m\_Gs1composite.bCC\_C;

### 4.3.11 GetGS1DATABAR

#### Description

Gets the option of GS1DATABAR Barcode.

#### Syntax

```
public bool GetGS1DATABAR(out GS1DATABAR_PARAMS pGs1Databar);
```

#### Parameters

*pGs1Databar*

Pointer to a GS1DATABAR\_PARAMS structure to be filled in with the GS1DATABAR common parameters.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

SetGS1DATABAR

#### For C++

Library : LRScanner.lib

Function : BOOL LRSCAN\_GetGS1DATABAR(PGS1DATABAR\_PARAMS pGs1Databar)

#### Example

```
private GS1DATABAR_PARAMS m_Gs1databar = new GS1DATABAR_PARAMS();  
m_LRScanner.GetGS1DATABAR(out m_Gs1databar);  
CB_ENABLE.Checked = m_Gs1databar.bEnable;  
CB_GS1LIM.Checked = m_Gs1databar.bGS1LIM;  
CB_GS1EXP.Checked = m_Gs1databar.bGS1EXP;
```

### 4.3.12 GetINT25

#### Description

Gets 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetINT25(PINT25\_PARAMS pInt25)

**Example**

```
private INT25_PARAMS m_Int25 = new INT25_PARAMS();  
m_LRScanner.GetINT25(out m_Int25);  
CB_ENABLE.Checked = m_Int25.bEnable;  
CB_CDV.Checked = m_Int25.bCDV;  
CB_XCD.Checked = m_Int25.bXCD;
```

### 4.3.13 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetMSI(PMSI\_PARAMS pMsi)

**Example**

```
private MSI_PARAMS m_Msi = new MSI_PARAMS();
m_LRScanner.GetMSI(out m_Msi);
CB_ENABLE.Checked = m_Msi.bEnable;
CB_CDV.Checked = m_Msi.bCDV;
if (m_Msi.bXCD == false)
    RD_MOD10.Checked = true;
else if (m_Msi.bXCD == true)
    RD_MOD10_10.Checked = true;
```

### 4.3.14 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetOption(PDECODER\_PARAMS pOption)

#### Example

```
private DECODER_PARAMS m_DecoderParams = new DECODER_PARAMS();
m_LRScanner.GetOption(out m_DecoderParams);
if (m_DecoderParams.nSound == 0)
    RD_SOUNDDEFAULT.Checked = true;
else if (m_DecoderParams.nSound == 1)
    RD_BEEP.Checked = true;
else
```

```

RD_NOSOUND.Checked = true;
CB_TIMEOUT.SelectedIndex = m_DecoderParams.nTimeOut - 1;
CB_SECURITY.SelectedIndex = m_DecoderParams.nSecurityLevel - 1;
CB_CONTINUE.Checked = m_DecoderParams.bContinueMode;
CB_AIMID.Checked = m_DecoderParams.bXmitAimID;
CB_VIBRATE.Checked = m_DecoderParams.bVibrate;
CB_1DDECODER.Checked = m_DecoderParams.b1DDecodeMode;
CB_CENTERDECODER.Checked = m_DecoderParams.bCenterDecode;

```

### 4.3.15 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetPLESSEY(PPLESSEY\_PARAMS pPlessey)

#### Example

```

private PLESSEY_PARAMS m_Plessey = new PLESSEY_PARAMS();
m_LRScanner.GetPLESSEY(out m_Plessey);
CB_ENABLE.Checked = m_Plessey.bEnable;
CB_CDV.Checked = m_Plessey.bCDV;

```

### 4.3.16 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetPOSTNET(PPOSTNET\_PARAMS pPostNet)

#### **Example**

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

### **4.3.17 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetScanData(TCHAR \*pszBarType, TCHAR \*pszBarData)

#### Example

None

### 4.3.18 GetSTANDARD2OF5

#### Description

Gets the option of STANDARD2OF5 Barcode.

#### Syntax

```
public bool GetSTANDARD2OF5(out STANDARD2OF5_PARAMS pStandard2of5);
```

#### Parameters

*pStandard2of5*

Pointer to a STANDARD2OF5\_PARAMS structure to be filled in with the STANDARD2OF5 common parameters.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

SetSTANDARD2OF5

#### For C++

Library : LRScanner.lib

Function : BOOL LRSCAN\_GetSTANDARD2OF5(PSTANDARD2OF5\_PARAMS pStandard2of5)

#### Example

```
private STANDARD2OF5_PARAMS m_Standard2of5 = new STANDARD2OF5_PARAMS();  
m_LRScanner.GetSTANDARD2OF5(out m_Standard2of5);  
CB_ENABLE.Checked = m_Standard2of5.bEnable;  
CB_CDV.Checked = m_Standard2of5.bCDV;  
CB_XCD.Checked = m_Standard2of5.bXCD;
```

### 4.3.19 GetSymbology

#### Description

Gets Enable/Disable of Symbologies.

#### Syntax

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

#### 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetSymbology(PDECODER pSym)

#### Example

```
public DECODER m_Decoder = new DECODER();
m_LRScanner.GetSymbology(out m_Decoder);
BARCODE_ENABLE[0] = m_Decoder.bAUSPOST;
BARCODE_ENABLE[1] = m_Decoder.bAZTEC;
BARCODE_ENABLE[2] = m_Decoder.bBPO;
BARCODE_ENABLE[3] = m_Decoder.bCANADAPOST;
BARCODE_ENABLE[4] = m_Decoder.bCODABAR;
BARCODE_ENABLE[5] = m_Decoder.bCODABLOCK;
BARCODE_ENABLE[6] = m_Decoder.bCODE11;
BARCODE_ENABLE[7] = m_Decoder.bCODE39;
BARCODE_ENABLE[8] = m_Decoder.bCODE93;
BARCODE_ENABLE[9] = m_Decoder.bCODE128;
BARCODE_ENABLE[10] = m_Decoder.bDATAMATRIX;
BARCODE_ENABLE[11] = m_Decoder.bDUTCHPOST;
BARCODE_ENABLE[12] = m_Decoder.bUPCA;
BARCODE_ENABLE[13] = m_Decoder.bUPCE;
```

```
BARCODE_ENABLE[14] = m_Decoder.bEAN8;
BARCODE_ENABLE[15] = m_Decoder.bEAN13;
BARCODE_ENABLE[16] = m_Decoder.bGS1_COMPOSITE;
BARCODE_ENABLE[17] = m_Decoder.bGS1_DATABAR;
BARCODE_ENABLE[18] = m_Decoder.bINFOMAIL;
BARCODE_ENABLE[19] = m_Decoder.bINT25;
BARCODE_ENABLE[20] = m_Decoder.bJAPANPOST;
BARCODE_ENABLE[21] = m_Decoder.bMATRIX2OF5;
BARCODE_ENABLE[22] = m_Decoder.bMAXICODE;
BARCODE_ENABLE[23] = m_Decoder.bMSI;
BARCODE_ENABLE[24] = m_Decoder.bPDF417;
BARCODE_ENABLE[25] = m_Decoder.bMICRO_PDF417;
BARCODE_ENABLE[26] = m_Decoder.bPLANET;
BARCODE_ENABLE[27] = m_Decoder.bPLESSEY;
BARCODE_ENABLE[28] = m_Decoder.bPOSTNET;
BARCODE_ENABLE[29] = m_Decoder.bQRCODE;
BARCODE_ENABLE[30] = m_Decoder.bSTANDARD2OF5;
BARCODE_ENABLE[31] = m_Decoder.bSWEDENPOST;
BARCODE_ENABLE[32] = m_Decoder.bTELEPEN;
BARCODE_ENABLE[33] = m_Decoder.bTLC39;
```

### 4.3.20 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetTELEPEN(PTELEPEN\_PARAMS pTelepen)

#### Example

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

### 4.3.21 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetUPCA(PUPCA\_PARAMS pUpca)

#### Example

```
private UPCA_PARAMS m_Upca = new UPCA_PARAMS();  
m_LRScanner.GetUPCA(out m_Upca);  
CB_ENABLE.Checked = m_Upca.bEnable;  
CB_XNUM.Checked = m_Upca.bXNum;  
CB_XCD.Checked = m_Upca.bXCD;  
CB_UPCATOEAN13.Checked = m_Upca.bUPCA_AS_EAN13;
```

CB\_ADDON.Checked = m\_Upca.bAddOn;

### 4.3.22 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetUPCE(PUPCE\_PARAMS pUpce)

#### Example

```
private UPCE_PARAMS m_Upce = new UPCE_PARAMS();  
m_LRScanner.GetUPCE(out m_Upce);  
CB_ENABLE.Checked = m_Upce.bEnable;  
CB_XNUM.Checked = m_Upce.bXNum;  
CB_XCD.Checked = m_Upce.bXCD;  
CB_UPCATOUPCE.Checked = m_Upce.bUPCE_AS_UPCA;
```

### 4.3.23 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 : LRScanner.lib

Function : BOOL LRSCAN\_GetVersionInfo(TCHAR\* pszVersion)

#### **Example**

None

### **4.3.24 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 : LRScanner.lib

Function : BOOL LRSCAN\_Open()

#### **Example**

None

### **4.3.25 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 : LRScanner.lib

Function : BOOL LRSCAN\_Read()

#### **Example**

None

### **4.3.26 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 : LRScanner.lib

Function : BOOL LRSCAN\_ReadCancel()

#### **Example**



None

### 4.3.27 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_LRScanner.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_LRScanner.Open();
```

```

if (m_bWinCE == true)
    m_LRScanner.RegHotKey(1, m_nHotKeyCE, m_bSyncMode);
else
    m_LRScanner.RegHotKey(1, m_nHotKeyWM, m_bSyncMode);

```

### 4.3.28 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 : LRScanner.lib

Function : BOOL LRSCAN\_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_LRScanner.SetCODABAR(ref m_Codabar);

```

### 4.3.29 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 : LRScanner.lib

Function : BOOL LRSCAN\_SetCODABLOCK(PCODABLOCK\_PARAMS pCodablock)

**Example**

```
private CODABLOCK_PARAMS m_Codablock = new CODABLOCK_PARAMS();  
m_Codablock.bEnable = CB_ENABLE_CODABLOCKA.Checked;  
m_Codablock.bCodaBlock_F = CB_ENABLE_CODABLOCKF.Checked;  
m_LRScanner.SetCODABLOCK(ref m_Codablock);
```

### 4.3.30 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 : LRScanner.lib

Function : BOOL LRSCAN\_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;
if (RD_CDV1.Checked)
    m_Code11.bXCD = false;
else if (RD_CDV2.Checked)
    m_Code11.bXCD = true;
m_LRScanner.SetCODE11(ref m_Code11);
```

### 4.3.31 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 : LRScanner.lib

Function : BOOL LRSCAN\_SetCODE128(PCODE128\_PARAMS pCode128)

#### Example

```
private CODE128_PARAMS m_Code128 = new CODE128_PARAMS();
m_Code128.bEnable = CB_ENABLE.Checked;
m_Code128.bGs1_128 = CB_GS1128.Checked;
m_Code128.bIsbt128 = CB_ISBT128.Checked;
```

```
m_LRScanner.SetCODE128(ref m_Code128);
```

### 4.3.32 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 : LRScanner.lib

Function : BOOL LRSCAN\_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;  
m_Code39.bFullASCII = CB_FULLASCII.Checked;  
m_LRScanner.SetCODE39(ref m_Code39);
```

### 4.3.33 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 EAN13 common parameters.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

GetEAN13

#### **For C++**

Library : LRScanner.lib

Function : BOOL LRSCAN\_SetEAN13(PEAN13\_PARAMS pEan13)

#### **Example**

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

### **4.3.34 SetEAN8**

#### **Description**

Sets the option of EAN-8 Barcode.

#### **Syntax**

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

#### **Parameters**

*pEan8*

Pointer to an EAN8\_PARAMS structure holding the EAN8 common parameters.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

GetEAN8

#### **For C++**

Library : LRScanner.lib

Function : `BOOL LRSCAN_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.bEAN8_AS_EAN13 = CB_EAN8TOEAN13.Checked;  
m_LRScanner.SetEAN8(ref m_Ean8);
```

### **4.3.35 SetGS1COMPOSITE**

#### **Description**

Sets the option of GS1COMPOSITE Barcode.

#### **Syntax**

```
public bool SetGS1COMPOSITE(ref GS1COMPOSITE_PARAMS pGs1Composite);
```

#### **Parameters**

*pGs1Composite*

Pointer to a GS1COMPOSITE\_PARAMS structure holding the GS1COMPOSITE common parameters.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

GetGS1COMPOSITE

#### **For C++**

Library : LRScanner.lib

Function : `BOOL LRSCAN_SetGS1COMPOSITE(PGS1COMPOSITE_PARAMS pGs1Composite)`

#### **Example**

```
private GS1COMPOSITE_PARAMS m_Gs1composite = new GS1COMPOSITE_PARAMS();  
m_Gs1composite.bEnable = CB_ENABLE.Checked;  
m_Gs1composite.bCC_C = CB_ENABLE_CC.Checked;  
m_LRScanner.SetGS1COMPOSITE(ref m_Gs1composite);
```

### **4.3.36 SetGS1DATABAR**

#### **Description**

Sets the option of GS1DATABAR Barcode.

**Syntax**

```
public bool SetGS1DATABAR(ref GS1DATABAR_PARAMS pGs1Databar);
```

**Parameters**

*pGs1Databar*

Pointer to a GS1DATABAR\_PARAMS structure holding the GS1DATABAR common parameters.

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

GetGS1DATABAR

**For C++**

Library : LRScanner.lib

Function : BOOL LRSCAN\_SetGS1DATABAR(PGS1DATABAR\_PARAMS pGs1Databar)

**Example**

```
private GS1DATABAR_PARAMS m_Gs1databar = new GS1DATABAR_PARAMS();  
m_Gs1databar.bEnable = CB_ENABLE.Checked;  
m_Gs1databar.bGS1LIM = CB_GS1LIM.Checked;  
m_Gs1databar.bGS1EXP = CB_GS1EXP.Checked;  
m_LRScanner.SetGS1DATABAR(ref m_Gs1databar);
```

### 4.3.37 SetINT25

**Description**

Sets the option of INT25 Barcode.

**Syntax**

```
public bool SetINT25(ref INT25_PARAMS pInt25);
```

**Parameters**

*pInt25*

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 : LRScanner.lib

Function : BOOL LRSCAN\_SetINT25(PINT25\_PARAMS pInt25)

#### Example

```
private INT25_PARAMS m_Int25 = new INT25_PARAMS();  
m_Standard2of5.bEnable = CB_ENABLE.Checked;  
m_Standard2of5.bCDV = CB_CDV.Checked;  
m_Standard2of5.bXCD = CB_XCD.Checked;  
m_LRScanner.SetSTANDARD2OF5(ref m_Standard2of5);
```

### 4.3.38 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 : LRScanner.lib

Function : BOOL LRSCAN\_SetMSI(PMSI\_PARAMS pMsi)

#### Example

```
private MSI_PARAMS m_Msi = new MSI_PARAMS();  
m_Msi.bEnable = CB_ENABLE.Checked;  
m_Msi.bCDV = CB_CDV.Checked;  
if (RD_MOD10.Checked)  
    m_Msi.bXCD = false;
```

```
else if (RD_MOD10_10.Checked)
    m_Msi.bXCD = true;
m_LRScanner.SetMSI(ref m_Msi);
```

### 4.3.39 SetOption

#### Description

Sets the option of Scanner.

#### Syntax

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

#### Parameters

*pOption*

Pointer to a DECODER\_PARAMS structure holding the scanner parameters.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

GetOption

#### For C++

Library : LRScanner.lib

Function : BOOL LRSCAN\_SetOption(PDECODER\_PARAMS pOption)

#### Example

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

```
m_DecoderParams.b1DDecodeMode = CB_1DDECODER.Checked;  
m_DecoderParams.bCenterDecode = CB_CENTERDECODER.Checked;  
m_LRScanner.SetOption(ref m_DecoderParams);
```

#### 4.3.40 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 PPLESSEY common parameters.

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

GetPLESSEY

##### For C++

Library : LRScanner.lib

Function : BOOL LRSCAN\_SetPLESSEY(PPLESSEY\_PARAMS pPlessey)

##### Example

```
private PLESSEY_PARAMS m_Plessey = new PLESSEY_PARAMS();  
m_Plessey.bEnable = CB_ENABLE.Checked;  
m_Plessey.bCDV = CB_CDV.Checked;  
m_LRScanner.SetPLESSEY(ref m_Plessey);
```

#### 4.3.41 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 : LRScanner.lib

Function : BOOL LRSCAN\_SetPOSTNET(PPOSTNET\_PARAMS pPostNet)

#### **Example**

```
private POSTNET_PARAMS m_Postnet = new POSTNET_PARAMS();  
m_Postnet.bEnable = CB_ENABLE.Checked;  
m_Postnet.bXCD = CB_XCD.Checked;  
m_LRScanner.SetPOSTNET(ref m_Postnet);
```

### **4.3.42 SetSTANDARD2OF5**

#### **Description**

Sets the option of STANDARD2OF5 Barcode.

#### **Syntax**

```
public bool SetSTANDARD2OF5(ref STANDARD2OF5_PARAMS pStandard2of5);
```

#### **Parameters**

*pStandard2of5*

Pointer to a STANDARD2OF5\_PARAMS structure holding the STANDARD2OF5 common parameters.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

GetSTANDARD2OF5

#### **For C++**

Namespace : LRScannerNet.LRScanner

Function : bool LRSCAN\_SetSTANDARD2OF5(PSTANDARD2OF5\_PARAMS pStandard2of5)

#### **Example**

```
private STANDARD2OF5_PARAMS m_Standard2of5 = new STANDARD2OF5_PARAMS();
m_Standard2of5.bEnable = CB_ENABLE.Checked;
m_Standard2of5.bCDV = CB_CDV.Checked;
m_Standard2of5.bXCD = CB_XCD.Checked;
m_LRScanner.SetSTANDARD2OF5(ref m_Standard2of5);
```

### 4.3.43 SetSymbology

#### Description

Sets Enable/Disable of Symbologies.

#### Syntax

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

#### 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 : LRScanner.lib

Function : BOOL LRSCAN\_SetSymbology(PDECODER pSym)

#### Example

```
public DECODER m_Decoder = new DECODER();
m_Decoder.bAUSPOST = LV_SYMLIST.Items[0].Checked;
m_Decoder.bAZTEC = LV_SYMLIST.Items[1].Checked;
m_Decoder.bBPO = LV_SYMLIST.Items[2].Checked;
m_Decoder.bCANADAPOST = LV_SYMLIST.Items[3].Checked;
m_Decoder.bCODABAR = LV_SYMLIST.Items[4].Checked;
m_Decoder.bCODABLOCK = LV_SYMLIST.Items[5].Checked;
m_Decoder.bCODE11 = LV_SYMLIST.Items[6].Checked;
m_Decoder.bCODE128 = LV_SYMLIST.Items[7].Checked;
m_Decoder.bCODE39 = LV_SYMLIST.Items[8].Checked;
```

```
m_Decoder.bCODE93 = LV_SYMLIST.Items[9].Checked;
m_Decoder.bDATAMATRIX = LV_SYMLIST.Items[10].Checked;
m_Decoder.bDUTCHPOST = LV_SYMLIST.Items[11].Checked;
m_Decoder.bEAN13 = LV_SYMLIST.Items[12].Checked;
m_Decoder.bEAN8 = LV_SYMLIST.Items[13].Checked;
m_Decoder.bGS1_COMPOSITE = LV_SYMLIST.Items[14].Checked;
m_Decoder.bGS1_DATABAR = LV_SYMLIST.Items[15].Checked;
m_Decoder.bINFOMAIL = LV_SYMLIST.Items[16].Checked;
m_Decoder.bINT25 = LV_SYMLIST.Items[17].Checked;
m_Decoder.bJAPANPOST = LV_SYMLIST.Items[18].Checked;
m_Decoder.bMATRIX2OF5 = LV_SYMLIST.Items[19].Checked;
m_Decoder.bMAXICODE = LV_SYMLIST.Items[20].Checked;
m_Decoder.bMICRO_PDF417 = LV_SYMLIST.Items[21].Checked;
m_Decoder.bMSI = LV_SYMLIST.Items[22].Checked;
m_Decoder.bPDF417 = LV_SYMLIST.Items[23].Checked;
m_Decoder.bPLANET = LV_SYMLIST.Items[24].Checked;
m_Decoder.bPLESSEY = LV_SYMLIST.Items[25].Checked;
m_Decoder.bPOSTNET = LV_SYMLIST.Items[26].Checked;
m_Decoder.bQRCODE = LV_SYMLIST.Items[27].Checked;
m_Decoder.bSTANDARD2OF5 = LV_SYMLIST.Items[28].Checked;
m_Decoder.bSWEDENPOST = LV_SYMLIST.Items[29].Checked;
m_Decoder.bTELEPEN = LV_SYMLIST.Items[30].Checked;
m_Decoder.bTLC39 = LV_SYMLIST.Items[31].Checked;
m_Decoder.bUPCA = LV_SYMLIST.Items[32].Checked;
m_Decoder.bUPCE = LV_SYMLIST.Items[33].Checked;
m_LRScanner.SetSymbology(ref m_Decoder);
```

#### 4.3.44 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 : LRScanner.lib

Function : BOOL LRSCAN\_SetSymbologyAll()

**Example**

None

### 4.3.45 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 : LRScanner.lib

Function : BOOL LRSCAN\_SetSymbologyDefault()

**Example**

None

### 4.3.46 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 : LRScanner.lib

Function : BOOL LRSCAN\_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_LRScanner.SetTELEPEN(ref m_Telepen);
```

## 4.3.47 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 : LRScanner.lib

Function : BOOL LRSCAN\_SetUPCA(PUPCA\_PARAMS pUpca)

#### Example

```
private UPCA_PARAMS m_Upca = new UPCA_PARAMS();
m_Upca.bEnable = CB_ENABLE.Checked;
m_Upca.bXNum = CB_XNUM.Checked;
m_Upca.bXCD = CB_XCD.Checked;
m_Upca.bUPCA_AS_EAN13 = CB_UPCATOEAN13.Checked;
m_Upca.bAddOn = CB_ADDON.Checked;
m_LRScanner.SetUPCA(ref m_Upca);
```

### 4.3.48 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 : LRScanner.lib

Function : BOOL LRSCAN\_SetUPCE(PUPCE\_PARAMS pUpce)

#### Example

```
private UPCE_PARAMS m_Upce = new UPCE_PARAMS();
m_Upce.bEnable = CB_ENABLE.Checked;
m_Upce.bXNum = CB_XNUM.Checked;
m_Upce.bXCD = CB_XCD.Checked;
```

```
m_Upce.bUPCE_AS_UPCA = CB_UPCATOUPCE.Checked;  
m_LRScanner.SetUPCE(ref m_Upce);
```

### 4.3.49 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

## 4.4 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_13M,
    MODEL_GREEN,
    MODEL_T,
    MODEL_POS,
    MODEL_SMART,
    MODEL_ORANGE_CE,
    MODEL_UL10,
    MODEL_BLACK_CE,
    MODEL_UL10_QVGA,
    MODEL_BLACK_QVGA,
    MODEL_ORANGE_CE_VGA,
    MODEL_UNKNOWN
}

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

### 4.4.1 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;
}
```

### 4.4.2 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 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();
```

### **4.4.4 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();
```

### **4.4.5 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\\");
}
```

## 4.4.6 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);
```

## 4.4.7 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();
```

## 4.4.8 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);
```

## 4.4.9 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);
```

## 4.4.10 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\\");
```

## 4.4.11 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);
```

### 4.4.12 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);
```

### 4.4.13 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();
```

### **4.4.14 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();
```

### **4.4.15 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

### 4.4.16 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);
```

#### 4.4.17 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);
```

#### 4.4.18 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();
```

### 4.4.19 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

## 4.5 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_BLACK =5,     MODEL_UNKONWN = 99, }  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
<a href="#">Open</a>	Opens the Camera.
<a href="#">Close</a>	Closes the Camera.
<a href="#">SetPreviewWindow</a>	Sets the size of Preview screen.
<a href="#">PreviewStart</a>	Starts viewing the Preview screen.
<a href="#">PreviewStop</a>	Stops viewing the Preview screen.
<a href="#">GetRegCapturePath</a>	Get save filename.(old version)
<a href="#">GetRegCapturePathEx</a>	Get save filename.
<a href="#">SetRegCapturePathEx</a>	Set save filename.
<a href="#">Capture</a>	Captures the Still Picture. (old version)
<a href="#">CaptureEx</a>	Captures the Still Picture.
<a href="#">VideoStart</a>	Starts Videoing.
<a href="#">VideoStop</a>	Stops Videoing. (old version)
<a href="#">VideoStopEx</a>	Stops Videoing.
<a href="#">GetFlashState</a>	Flash On/Off check
<a href="#">AlwaysFlash</a>	Turns on the Flash all the time.
<a href="#">CaptureFlash</a>	Sets whether the flash is on or not when capturing picture.
<a href="#">AutoFocus</a>	Focuses automatically.
<a href="#">SetAfMode</a>	Sets AF Mode.
<a href="#">Zoom</a>	Zooms in and/or out.
<a href="#">SetResolution</a>	Sets Resolution of picture.
<a href="#">GetResolution</a>	Gets setting value of resolution.
<a href="#">SetQuality</a>	Sets quality of Jpeg Still Shot.
<a href="#">GetQuality</a>	Gets quality value of Jpeg Still shot.
<a href="#">SetBrightness</a>	Sets brightness of camera.
<a href="#">GetBrightness</a>	Gets brightness of camera.
<a href="#">SetWhiteBalance</a>	Sets white balance of camera.
<a href="#">GetWhiteBalance</a>	Gets white balance of camera.
<a href="#">SetContrast</a>	Sets contrast of camera.
<a href="#">GetContrast</a>	Gets contrast of camera.
<a href="#">SetSharpness</a>	Sets sharpness of camera.

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

## 4.5.1 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;
}
```

## 4.5.2 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();
```

## **4.5.3 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
```

### **4.5.4 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();
```

### **4.5.5 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();
```

### **4.5.6 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)
```

## 4.5.7 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)
```

## 4.5.8 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);
```

### 4.5.9 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);
```

### 4.5.10 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();
```

### **4.5.11 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);
```

### 4.5.12 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);
```

### 4.5.13 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();
```

### 4.5.14 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);
```

### 4.5.15 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);
```

## 4.5.16 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();
```

### 4.5.17 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);
```

### 4.5.18 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);
```

### 4.5.19 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);
```

### 4.5.20 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();
```

## 4.5.21 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);
```

## 4.5.22 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();
```

## 4.5.23 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);
```

## 4.5.24 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();
```

## 4.5.25 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);
```

## 4.5.26 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();
```

## 4.5.27 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);
```

## **4.5.28 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();
```

## 4.5.29 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);
```

## 4.5.30 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();
```

## 4.5.31 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);
```

## 4.5.32 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

### **4.5.33 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

### 4.5.34 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);
```

### 4.5.35 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

## 4.5.36 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();
```

## 4.5.37 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();
```

### 4.5.38 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

## 4.6 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
<a href="#">Open</a>	Opens RFID.
<a href="#">Close</a>	Closes RFID
<a href="#">PowerSupply</a>	Supplies power to RFID module.
<a href="#">GetRadioType</a>	Gets RFID Radio Type of device.
<a href="#">SelectTag</a>	Searches and Selects one tag within Antenna field.
<a href="#">HighSelectTag</a>	Searches one tag within Antenna field and selects as High baudrate.
<a href="#">LoginTag</a>	Login the tag if necessary.
<a href="#">ReadBlock</a>	Reads memory block of the tag.
<a href="#">WriteBlock</a>	Writes data in memory block of the tag.
<a href="#">ReadMultiBlock</a>	Reads multiple data blocks on a card.
<a href="#">WriteMultiBlock</a>	Writes multiple data blocks on a card.
<a href="#">SetAntenna</a>	Controls antenna power of the RFID Module.
<a href="#">GetVersion</a>	Gets DLL information and FW information of reader.
<a href="#">EnableMultiTag</a>	Enables Anti-Collision function so that multi tags can be read.
<a href="#">SendReadMultiTag</a>	Sends commander reading multi tags to reader.
<a href="#">SendTransferCommand</a>	Allows card-specific communication.
<a href="#">SendContinuousRead</a>	Reads and displays serial numbers continuously while one or more tags remain in the field.
<a href="#">SendCommand</a>	Sends a command to a reader specified by its handle
<a href="#">SendCommandGetData</a>	Sends a command to a reader specified by its handle and receives data.
<a href="#">GetData</a>	Gets the result that performs commander stored in reader.
<a href="#">CheckResult</a>	Checks obtained result from reader if it is valid.
<a href="#">SoundPlay</a>	Plays sound and vibration.
<a href="#">SetTagType</a>	Sets up the reader for a specific tag type.
<a href="#">GetTagType</a>	Gets tag type.
<a href="#">GetTagTypeToString</a>	Gets tag type as string.
<a href="#">TagItInventory</a>	Performs Inventory commander of TagIt tag.
<a href="#">TagItReadBlock</a>	Reads memory block of TagIt tag.
<a href="#">TagItWriteBlock</a>	Writes memory block of TagIt tag.

## 4.6.1 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;
}
```

## 4.6.2 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();
```

## 4.6.3 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);
```

#### 4.6.4 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;
}
```

### 4.6.5 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

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

### 4.6.6 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);
```

## 4.6.7 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);
    }
}

```

### 4.6.8 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  
}
```

#### 4.6.9 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);
```

#### 4.6.10 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  
}
```

## 4.6.11 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);
```

## 4.6.12 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);
}
```

### **4.6.13 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);
```

## 4.6.14 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);
```

#### 4.6.15 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

#### 4.6.16 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

## 4.6.17 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);
```

## 4.6.18 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);  
}
```

## 4.6.19 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);
```

## 4.6.20 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);
```

## 4.6.21 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
}
```

## 4.6.22 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);.
```

### 4.6.23 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);
```

### 4.6.24 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();.
```

### 4.6.25 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);.
```

### 4.6.26 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);
```

## 4.6.27 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);
```

## 4.6.28 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);
```

## 4.7 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
<a href="#">GetError</a>	Check the last error
<a href="#">IsReady</a>	Check UHF GUN READER's availability by checking the power
<a href="#">Init</a>	Initialize RFID
<a href="#">Close</a>	Close RFID
<a href="#">Inventory</a>	Start Inventory by reading EPC data
<a href="#">InventoryStop</a>	Stop Inventory
<a href="#">Read</a>	Read memory from tag
<a href="#">Write</a>	Write data to tag
<a href="#">Lock</a>	Limit access to tag
<a href="#">Kill</a>	Kill the tag (disabling the tag)
<a href="#">GetData</a>	Get tag data from Inventory and Read
<a href="#">SetRegionFrequency</a>	Set RFID frequency to suit regional regulation
<a href="#">GetRegionFrequency</a>	Check current regional frequency setting
<a href="#">ReadBattery</a>	Check battery voltage and ADC value
<a href="#">ReadBatteryStatus</a>	Check battery level (0~4 level)
<a href="#">Version</a>	Check DLL and F/W version

### 4.7.1 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());
}
```

### 4.7.2 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;
```

### 4.7.3 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);  
}
```

## 4.7.4 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);  
}
```



### 4.7.5 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();
```

### 4.7.6 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();
```

### 4.7.7 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!");
}
```

### 4.7.8 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!");  
}
```

### 4.7.9 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!");
```

```
}
```

## 4.7.10 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!");  
}
```

### 4.7.11 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);
```

## 4.7.12 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;  
}
```

### 4.7.13 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();
```

### 4.7.14 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

### 4.7.15 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

### 4.7.16 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);
```

## 4.8 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
<a href="#">ActivateConfig</a>	Activate the configuration with the given name.
<a href="#">Close</a>	Free WLAN.dll.
<a href="#">ConnectAP</a>	Connecting to AP.
<a href="#">DeleteConfig</a>	This function deletes the configuration matching 'name'.
<a href="#">ExportConfig</a>	This function exports configurations.
<a href="#">ImportConfig</a>	This function imports configurations.
<a href="#">Init</a>	WLAN.dll initiation
<a href="#">GetAllConfigName</a>	This function retrieves all of the configurations.
<a href="#">GetCurrentAPInfo</a>	Get current AP information
<a href="#">GetBssidList</a>	Scans and lists connectable APs.
<a href="#">GetPowerStatus</a>	Get power status.
<a href="#">PowerOn</a>	Inserts WLAN card.
<a href="#">PowerOff</a>	Removes WLAN card.

## 4.8.1 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()");
```

## 4.8.2 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()");
}
```

### 4.8.3 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()");

```

#### 4.8.4 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()");
```

## 4.8.5 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!!");
```

## 4.8.6 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!!!";
```

## 4.8.7 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!!!";
```

### 4.8.8 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());
```

### 4.8.9 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);
```

### 4.8.10 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 ()");
```

### 4.8.11 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 == "");
```

### 4.8.12 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!"));
```

### 4.8.13 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()");
```

### **4.8.14 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()");
```

## 4.9 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
<a href="#">Close</a>	Performs cleanup after the API is no longer needed.
<a href="#">Connect</a>	Activates the specified connection in the connection list.
<a href="#">CreateConnection</a>	Defines a new connection, adding it to a list of connections.
<a href="#">CreateConnectionEx</a>	Defines a new connection, adding it to a list of connections.
<a href="#">DeleteConnection</a>	Deletes a connection from the list of connections.
<a href="#">Disconnect</a>	Disconnects from the specified connection.
<a href="#">FindConnectionClose</a>	Frees remote resources.
<a href="#">FindDeviceClose</a>	Frees remote resources.
<a href="#">FindFirstConnection</a>	Finds the first connection meeting the specified criteria.
<a href="#">FindFirstConnectionEx</a>	Finds the first connection meeting the specified criteria.
<a href="#">FindFirstDevice</a>	Performs a GAP inquiry to discover devices, returning the first device meeting the specified criteria.
<a href="#">FindFirstDeviceEx</a>	Performs a GAP inquiry to discover devices, returning the first device meeting the specified criteria. This supports searching for a device of specific type.
<a href="#">FindFirstService</a>	Performs an SDP query to discover remote services for a specified device, returning the first service in the list.
<a href="#">FindFirstServiceEx</a>	Performs an SDP query to discover remote services for a specified device, returning the first service in the list.
<a href="#">FindLocalDevice</a>	Returns device information for the local device.
<a href="#">FindNextConnection</a>	Returns the next connection meeting the criteria specified in the call to FindFirstConnection.
<a href="#">FindNextConnectionEx</a>	Returns the next connection meeting the criteria specified in the call to FindFirstConnection.
<a href="#">FindNextDevice</a>	Returns the next device in the list meeting the initial criteria, specified in the call to FindFirstDevice.
<a href="#">FindNextService</a>	Returns the next service in the list.
<a href="#">FindNextServiceEx</a>	Returns the next service in the list.
<a href="#">FindServiceClose</a>	Frees remote resources.
<a href="#">GetBluetoothState</a>	Gets the Bluetooth device state.
<a href="#">GetSCOConnectionState</a>	Gets the SCO connection state.
<a href="#">GetSecurityMode</a>	Gets the current authentication and encryption settings.
<a href="#">Open</a>	Initializes the BTE Explorer API module.
<a href="#">PerformAction</a>	Triggers BTE Explorer to take the requested action.
<a href="#">SetAuthenticationCallback</a>	Sets the authentication callback for the specified device.
<a href="#">SetBluetoothState</a>	Sets the Bluetooth device state to either on or off.

<a href="#">SetConnectionCallback</a>	Sets the connection callback for the specified connection.
<a href="#">SetIncomingPIN</a>	Sets or clears a static PIN to be used for any incoming Connections.
<a href="#">SetOutgoingPIN</a>	Sets or clears a static PIN to be used for any outgoing Connections.
<a href="#">SetSCOConnectionState</a>	Sets the SCO connection state to either connected or disconnected.
<a href="#">SetSecurityMode</a>	Sets the authentication and encryption settings for future connections.

### 4.9.1 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();
```

### 4.9.2 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);
```

### 4.9.3 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

### 4.9.4 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);
```

## 4.9.5 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);
```

### 4.9.6 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);
```

### 4.9.7 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);
```

## **4.9.8 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);

### 4.9.9 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);
```

### 4.9.10 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

## 4.9.11 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

## **4.9.12 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);
```

### 4.9.13 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

### 4.9.14 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);
```

### 4.9.15 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);
```

### 4.9.16 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);
```

## 4.9.17 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

## 4.9.18 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);
```

## 4.9.19 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

## 4.9.20 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);
```

## 4.9.21 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);
```

## 4.9.22 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

## 4.9.23 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

### 4.9.24 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

## 4.9.25 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";
}
```

## 4.9.26 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);
```

## 4.9.27 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

## 4.9.28 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

## **4.9.29 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

## 4.9.30 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

## 4.9.31 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

## 4.9.32 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

## 4.9.33 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

## 4.10 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     DEVICE_M3BLACK,     DEVICE_M3UL10_CE,     DEVICE_M3BLACK_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
<a href="#">BacklightOn</a>	Backlight On/Off
<a href="#">Cleanboot</a>	Cleanboot
<a href="#">GetBacklightLevel</a>	Gets the Backlight Level
<a href="#">GetBacklightTimeOut</a>	Gets the BacklightTimeOut
<a href="#">GetBatteryLifePercent</a>	Gets the Battery Life Percent
<a href="#">GetBatteryState</a>	Gets the Battery State
<a href="#">GetBluetooth</a>	Gets the Bluetooth
<a href="#">GetCpuClock</a>	Gets the CPU Clock
<a href="#">GetDeviceInfo</a>	Gets the Information of Device
<a href="#">GetGSensorValue</a>	Gets the Gyro Sensor Value
<a href="#">GetOSVersionInfo</a>	Gets the Information of OS Version
<a href="#">GetPhoneVolumeLevel</a>	Gets the Phone Volume Level
<a href="#">GetPowerTimeOut</a>	Gets the PowerTimeOut
<a href="#">GetSerialNumber</a>	Gets the Serial Number
<a href="#">GetVolumeLevel</a>	Gets the Volume Level
<a href="#">GetWlan</a>	Gets the Wlan
<a href="#">KeypadLock</a>	Kaypad Lock/Unlock
<a href="#">Reboot</a>	Reboot
<a href="#">SetBacklightLevel</a>	Sets the Backlight Level
<a href="#">SetBacklightTimeOut</a>	Sets the BacklightTimeOut
<a href="#">SetBluetooth</a>	Sets the Bluetooth On/Off
<a href="#">SetCpuClock</a>	Sets the CPU Clock
<a href="#">SetGSensorValue</a>	Sets the Gyro Sensor Value
<a href="#">SetPhoneVolumeLevel</a>	Sets the Phone Volume Level
<a href="#">SetPowerTimeOut</a>	Sets the PowerTimeOut
<a href="#">SetSleepMode</a>	Sleep On
<a href="#">SetVolumeLevel</a>	Sets the Volume Level
<a href="#">SetWlan</a>	Sets the Wlan On/Off
<a href="#">Vibrate</a>	Vibrate On/Off
<a href="#">VolumeMute</a>	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 ST10 CE, M3 OX10 CE and M3 UL10.

#### 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 St10, M3 OX10 WM, M3 OX10 CE and M3 UL10.

**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 SystemNet.MODEL_TYPE.DEVICE_M3SKY:
        M3Sky_Init();
        break;

    case SystemNet.MODEL_TYPE.DEVICE_M3ORANGE:
        M3Orange_Init();
        break;

    case SystemNet.MODEL_TYPE.DEVICE_M3SMARTWM:
        M3SmartWM_Init();
        break;

    case SystemNet.MODEL_TYPE.DEVICE_M3T:
        M3T_Init();
```

```

        break;

    case 6: //DEVICE_M3ORANGEPLUS
        M3OrangePLUS_Init();

        break;

    case SystemNet.MODEL_TYPE.DEVICE_Pos:
        Pos_Init();

        break;

    case SystemNet.MODEL_TYPE.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 ST10 WM.

##### See Also

SetGSensorValue

##### For C++

Library : M3System.lib

Function : BOOL SYSTEM\_GetGSensorValue(PGSensor\_PARAMS pGSensor\_Params);

##### Example

```

m_GSenor_Params = new GSenor_PARAMS();
m_System.GetGSensorValue(out m_GSenor_Params);

CB_MotionDisplayOff.Checked    = m_GSenor_Params.MotionDisplayOff;
CB_MotionDisplayWakeup.Checked = m_GSenor_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 GetPhoneVolumeLevel();
```

##### Parameters

None

##### Return Value

The return value is PhoneVolumeLevel

##### Remarks

None

### See Also

SetPhoneVolumeLevel

### For C++

Library : M3System.lib

Function : int SYSTEM\_GetPhoneVolumeLevel( )

### Example

```
Public int nVolumeLevel_Cur;
nVolumeLevel_Cur = GetPhoneVolumeLevel ();
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 GetVersionInfo

**Description**

Gets the Information DLL version.

**Syntax**

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

**Parameters**

None

**Return Value**

The return value is DLL Version information.

**Remarks**

None

**See Also**

None

**For C++**

Library : M3System.lib

Function : int SYSTEM\_GetVersionInfo(TCHAR\* pszVersion)

**Example**

```
Public string strVersion = "";  
strVersion = m_System.GetVersionInfo();
```

### 4.10.16 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;
```

## 4.10.17 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;
```

#### 4.10.18 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);
```

#### 4.10.19 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();
```

### **4.10.20 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);
```

### 4.10.21 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);
```

### 4.10.22 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);
```

## 4.10.23 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 ST10, M3 OX10 and UL10 CE.

**See Also**

GetCpuClock

**For C++**

Library : M3System.lib

Function : bool SYSTEM\_SetCpuClock(int cpu)

**Example**

```
m_System.SetCpuClock((int)SystemNet.SKY_CPUTYPE.CLOCK_208MHz);
```

```
m_System.SetCpuClock((int)SystemNet.SKY_CPUTYPE.CLOCK_312MHz);
```

```
m_System.SetCpuClock((int)SystemNet.SKY_CPUTYPE.CLOCK_416MHz);
```

#### 4.10.24 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 ST10 WM.

##### See Also

GetGSensorValue

##### For C++

Library : M3System.lib

Function : bool SYSTEM\_SetGSensorValue(PGSensor\_PARAMS pGSensor\_Params)

##### Example

```
m_GSensor_Params = new GSensor_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);
```

#### 4.10.25 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);
```

## 4.10.26 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);
```

#### 4.10.27 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();
```

#### 4.10.28 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);
```

### 4.10.29 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);
```

### 4.10.30 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);
```

### 4.10.31 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();
```

### 4.10.32 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.

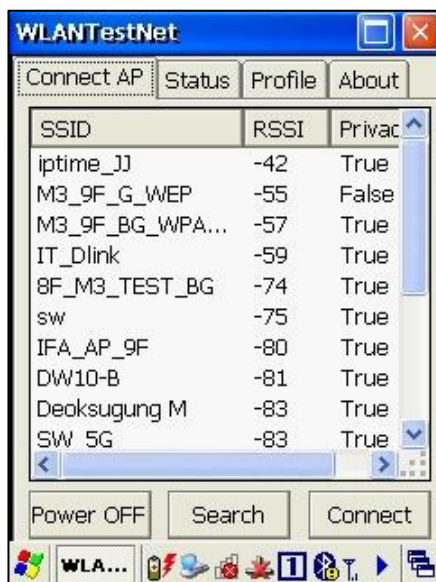
### 5.1 WLAN

Product	Model	Type	OS	Note
ALL	ALL	Summit	WM 6.1/6.5 CE 5.0/6.0	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 turn On/off.

Search : Scan available network.

Connect : Connect to selected network.

## 2. Connecting View for Encryption Type.



Connect Info : display View current status of WLAN.

Encryption : Access to an AP with stored settings.

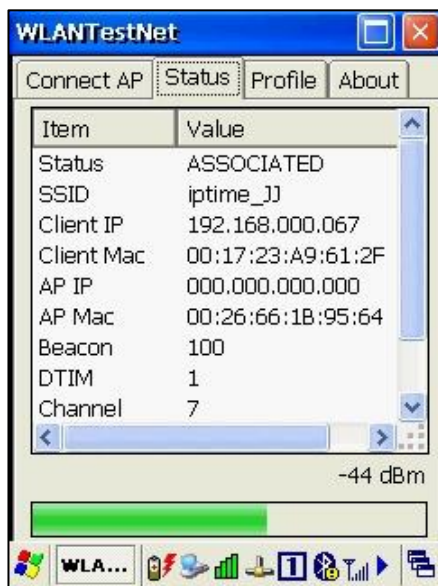
Password : Write password.

Confirm : Confirm password.

Yes : Connect to apply AP info.

No : Cance..

## 3. State View



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.

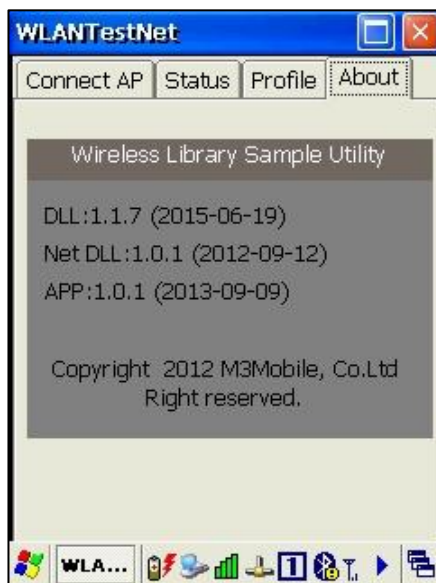
Connect : Connects to selected profile.

Delete : Deletes selected profile

Import : Restores the WLAN connection configuration.

Export : Back up of the WLAN connection configuration.

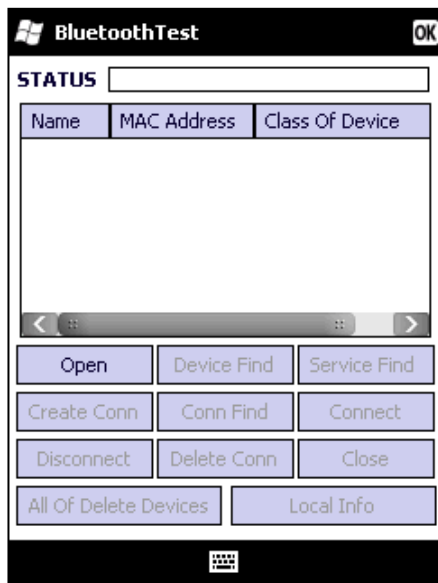
#### 5. About View



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 BTExplorer 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

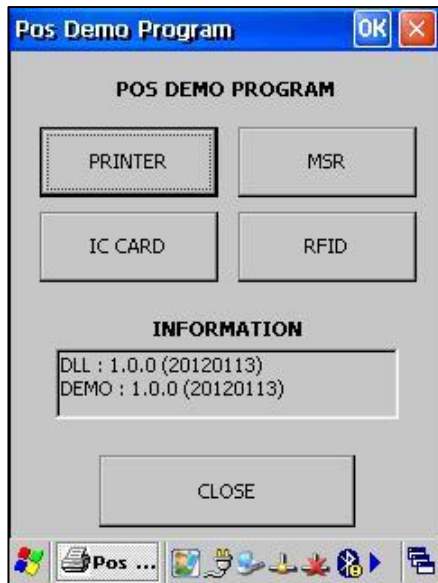
## 5.3 POS

Product	Model	Type	OS	Note
M3 PS10	M3 PS10	Software	CE 5.0	

### 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.

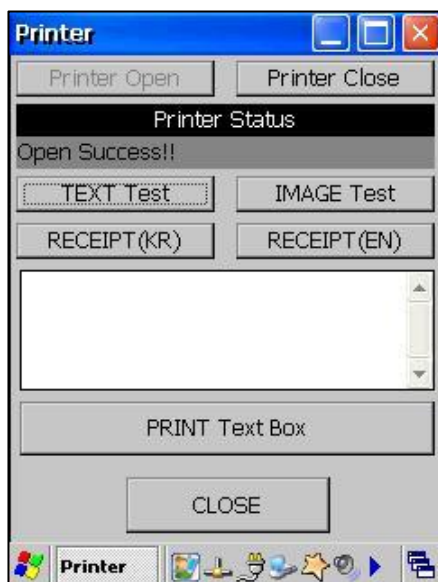
RFID : Runs RFID Test Dialog.

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.

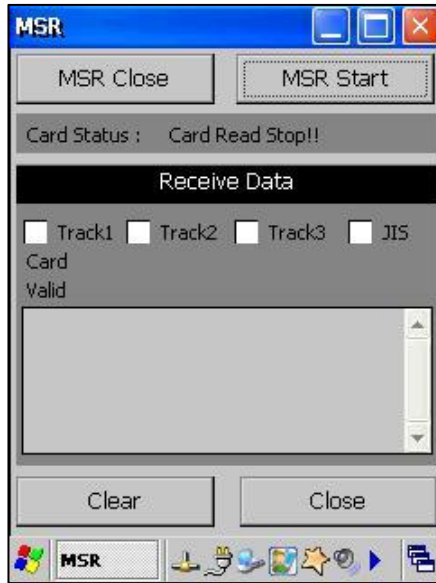
IMAGE Test : Prints text in the text box.

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.

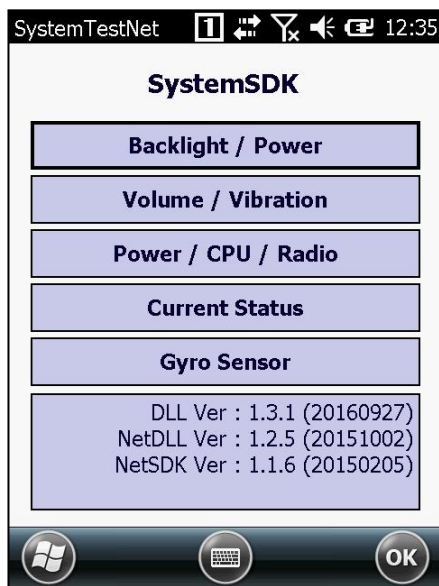
Close : Returns to first Dialog.

Card, Valid : No showing if Track 2 is not read.

## 5.4 SYSTEM

Product	Model	Type	OS	Note
M3 BK10	M3 BK10		WM 6.5 / CE 6.0	Gyro Sensor
M3 MT10	M3 MT10		CE 5.0	
M3 OX10	M3 OX10		WM 6.5 / CE 6.0	
M3 PS10	M3 PS10		CE 5.0	
M3 ST10	M3 ST10		WM 6.5 / CE 6.0	Gyro Sensor
M3 UL10	M3 UL10		WM 6.5	

### 1. Main Page



Backlight / Power

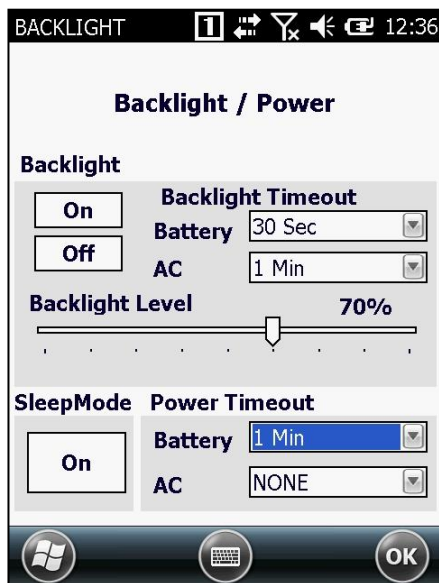
Volume / Vibration

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

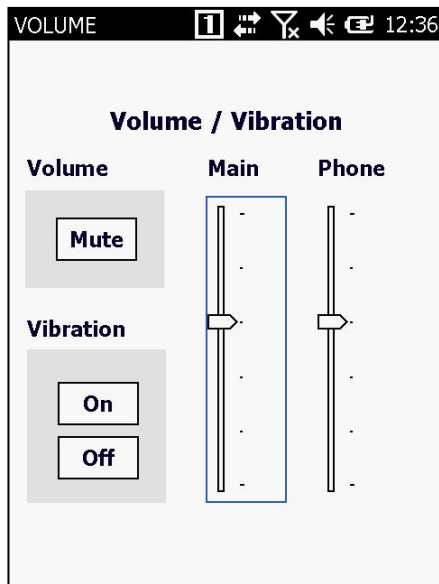
Power Timeout

Battery : Sets the Battery Backlight Timeout

AC : Sets the AC Backlight Timeout



### 3. Volume / Vibration



Volume

Mute : Volume Mute

Vibration

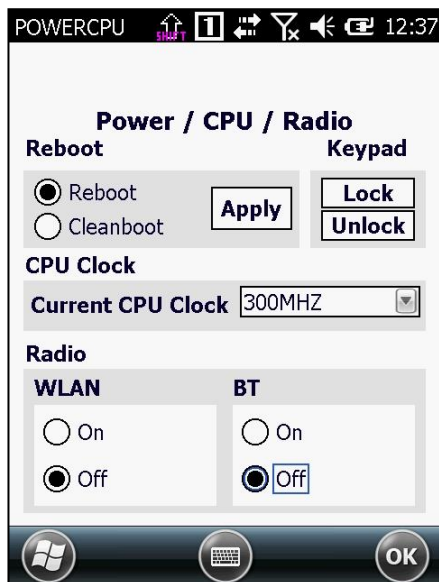
On : Vibration On

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

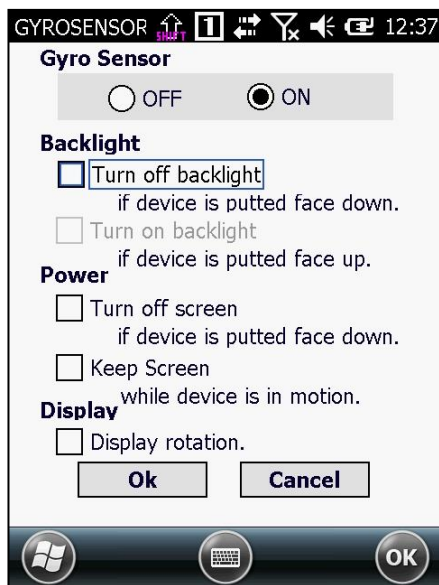
## 5. Current Status



### 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](http://www.m3mobile.net), [sales@m3mobile.co.kr](mailto: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>