

# Programmer's guide

M3 OX10 SDK Version 1.3 Manual

---

## **Abstract**

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

Modules covered:

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

## Copyright and Agreement

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

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

## Development Tools and Requirements

### Supported Development Tools

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

### Development System Requirements

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

### Development Platform Requirements

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

## Release History

### **M3 OX10 SDK Version 1.3      Date: July 2016**

- Added 45N Module of Wireless lan to supported model

### **M3 OX10 SDK Version 1.2      Date: March 2016**

- Add UHF Gun SDK
- Remove the POS SDK and Long Rang Scanner

### **UNI SDK Version 1.1.1      Date: November 2013**

- Added M3 OX10 CE
- System SDK
  - DLL names have been changed
    - System.DLL => M3system.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 .....	13
2	Samples.....	14
3	Tutorial.....	15
3.1	SCANNER (1D).....	17
3.2	IMAGER (2D) .....	19
3.3	CAMERA (CE) .....	21
3.4	CAMERA (WM).....	25
3.5	RFID (LF/HF) .....	28
3.6	UHF Gun Reader .....	30
3.7	WLAN .....	34
3.8	BLUETOOTH .....	37
3.9	GPS.....	48
3.10	SYSTEM SDK.....	54
4	References .....	56
4.1	SCANNER (1D).....	57
4.1.1	SCAN_Close .....	63
4.1.2	SCAN_GetCODABAR .....	63
4.1.3	SCAN_GetCODE11 .....	64
4.1.4	SCAN_GetCODE128 .....	65
4.1.5	SCAN_GetCODE25 .....	66
4.1.6	SCAN_GetCODE39 .....	67
4.1.7	SCAN_GetCODE93 .....	68
4.1.8	SCAN_GetDeviceType.....	68
4.1.9	SCAN_GetEAN13 .....	69
4.1.10	SCAN_GetEAN8 .....	70
4.1.11	SCAN_GetEngineType.....	70
4.1.12	SCAN_GetGS1 .....	71
4.1.13	SCAN_GetKOREAPOST .....	72
4.1.14	SCAN_GetMSI.....	72
4.1.15	SCAN_GetOption .....	73
4.1.16	SCAN_GetScanData .....	74
4.1.17	SCAN_GetSymbology.....	75
4.1.18	SCAN_GetTELEPEN .....	76
4.1.19	SCAN_GetUPCA .....	77
4.1.20	SCAN_GetUPCE .....	78
4.1.21	SCAN_GetVersionInfo.....	78
4.1.22	SCAN_Open.....	79

4.1.23	SCAN_Read .....	80
4.1.24	SCAN_ReadCancel .....	80
4.1.25	SCAN_SetCODABAR .....	81
4.1.26	SCAN_SetCODE11 .....	82
4.1.27	SCAN_SetCODE128 .....	82
4.1.28	SCAN_SetCODE25 .....	83
4.1.29	SCAN_SetCODE39 .....	84
4.1.30	SCAN_SetCODE93 .....	85
4.1.31	SCAN_SetEAN13 .....	86
4.1.32	SCAN_SetEAN8 .....	87
4.1.33	SCAN_SetGS1 .....	87
4.1.34	SCAN_SetKOREAPOST .....	88
4.1.35	SCAN_SetMSI .....	89
4.1.36	SCAN_SetOption .....	90
4.1.37	SCAN_SetSymbology .....	91
4.1.38	SCAN_SetSymbologyAll .....	92
4.1.39	SCAN_SetSymbologyDefault .....	92
4.1.40	SCAN_SetTELEPEN .....	93
4.1.41	SCAN_SetUPCA .....	93
4.1.42	SCAN_SetUPCE .....	94
4.2	IMAGER (2D) .....	96
4.2.1	IMAGER_CAMCapture .....	108
4.2.2	IMAGER_CAMGetOption .....	108
4.2.3	IMAGER_CAMInit .....	109
4.2.4	IMAGER_CAMPreviewStart .....	110
4.2.5	IMAGER_CAMPreviewStop .....	110
4.2.6	IMAGER_CAMSetOption .....	111
4.2.7	IMAGER_CAMUnInit .....	112
4.2.8	IMAGER_Close .....	112
4.2.9	IMAGER_GetAZTEC .....	113
4.2.10	IMAGER_GetCenteringWindow .....	114
4.2.11	IMAGER_GetCHINAPOST .....	115
4.2.12	IMAGER_GetCODABAR .....	115
4.2.13	IMAGER_GetCODABLOCK .....	116
4.2.14	IMAGER_GetCODE11 .....	117
4.2.15	IMAGER_GetCODE128 .....	118
4.2.16	IMAGER_GetCODE16K .....	119
4.2.17	IMAGER_GetCODE39 .....	119
4.2.18	IMAGER_GetCODE49 .....	120

4.2.19	IMAGER_GetCODE93 .....	121
4.2.20	IMAGER_GetCOMPOSITE .....	122
4.2.21	IMAGER_GetDATAMATRIX .....	123
4.2.22	IMAGER_GetDecOption .....	123
4.2.23	IMAGER_GetDeviceType .....	124
4.2.24	IMAGER_GetEAN13 .....	125
4.2.25	IMAGER_GetEAN8 .....	126
4.2.26	IMAGER_GetIATA25 .....	126
4.2.27	IMAGER_GetINT25 .....	127
4.2.28	IMAGER_GetKOREAPOST .....	128
4.2.29	IMAGER_GetMATRIX25 .....	129
4.2.30	IMAGER_GetMAXICODE .....	129
4.2.31	IMAGER_GetMICROPDF .....	130
4.2.32	IMAGER_GetMSI .....	131
4.2.33	IMAGER_GetOCR .....	132
4.2.34	IMAGER_GetOption .....	133
4.2.35	IMAGER_GetPDF417 .....	134
4.2.36	IMAGER_GetPLANET .....	135
4.2.37	IMAGER_GetPLESSEY .....	135
4.2.38	IMAGER_GetPOSICODE .....	136
4.2.39	IMAGER_GetPOSTNET .....	137
4.2.40	IMAGER_GetQR .....	138
4.2.41	IMAGER_GetRSS .....	138
4.2.42	IMAGER_GetScanData .....	139
4.2.43	IMAGER_GetSTRT25 .....	140
4.2.44	IMAGER_GetSymbology .....	141
4.2.45	IMAGER_GetTELEPEN .....	143
4.2.46	IMAGER_GetUPCA .....	144
4.2.47	IMAGER_GetUPCE .....	144
4.2.48	IMAGER_GetVersionInfo .....	145
4.2.49	IMAGER_IQGetBarcodeData .....	146
4.2.50	IMAGER_IQGetOption .....	147
4.2.51	IMAGER_IQImagingStart .....	147
4.2.52	IMAGER_IQImagingStop .....	148
4.2.53	IMAGER_IQInit .....	149
4.2.54	IMAGER_IQSetOption .....	149
4.2.55	IMAGER_IQUnInit .....	150
4.2.56	IMAGER_Open .....	151
4.2.57	IMAGER_Read .....	151

4.2.58	IMAGER_ReadCancel .....	152
4.2.59	IMAGER_SetAZTEC .....	152
4.2.60	IMAGER_SetCenteringWindow .....	153
4.2.61	IMAGER_SetCHINAPOST .....	154
4.2.62	IMAGER_SetCODABAR.....	155
4.2.63	IMAGER_SetCODABLOCK.....	155
4.2.64	IMAGER_SetCODE11 .....	156
4.2.65	IMAGER_SetCODE128 .....	157
4.2.66	IMAGER_SetCODE16K .....	158
4.2.67	IMAGER_SetCODE39 .....	159
4.2.68	IMAGER_SetCODE49 .....	159
4.2.69	IMAGER_SetCODE93 .....	160
4.2.70	IMAGER_SetCOMPOSITE .....	161
4.2.71	IMAGER_SetDATAMATRIX .....	162
4.2.72	IMAGER_SetDecOption .....	163
4.2.73	IMAGER_SetEAN13.....	163
4.2.74	IMAGER_SetEAN8 .....	164
4.2.75	IMAGER_SetIATA25 .....	165
4.2.76	IMAGER_SetINT25 .....	166
4.2.77	IMAGER_SetKOREAPOST .....	167
4.2.78	IMAGER_SetMATRIX25 .....	167
4.2.79	IMAGER_SetMAXICODE .....	168
4.2.80	IMAGER_SetMICROPDF .....	169
4.2.81	IMAGER_SetMSI .....	170
4.2.82	IMAGER_SetOCR .....	170
4.2.83	IMAGER_SetOption .....	171
4.2.84	IMAGER_SetPDF417 .....	172
4.2.85	IMAGER_SetPLANET .....	173
4.2.86	IMAGER_SetPLESSEY .....	174
4.2.87	IMAGER_SetPOSICODE .....	175
4.2.88	IMAGER_SetPOSTNET .....	175
4.2.89	IMAGER_SetQR .....	176
4.2.90	IMAGER_SetRSS .....	177
4.2.91	IMAGER_SetSTRT25.....	178
4.2.92	IMAGER_SetSymbology .....	178
4.2.93	IMAGER_SetSymbologyAll .....	181
4.2.94	IMAGER_SetSymbologyDefault.....	181
4.2.95	IMAGER_SetTELEPEN .....	182
4.2.96	IMAGER_SetUPCA .....	183

4.2.97	IMAGER_SetUPCE.....	183
4.3	CAMERA (CE) .....	185
4.3.1	CAM_Open.....	189
4.3.2	CAM_Close.....	189
4.3.3	CAM_Capture.....	190
4.3.4	CAM_PreviewStart .....	191
4.3.5	CAM_PreviewStop.....	191
4.3.6	CAM_GetLastSaveFilePath.....	192
4.3.7	CAM_AutoFocus.....	192
4.3.8	CAM_EnableAutoAF .....	193
4.3.9	CAM_Zoom .....	194
4.3.10	CAM_SetCameraOption.....	194
4.3.11	CAM_GetCameraOption .....	195
4.3.12	CAM_Brightness .....	196
4.3.13	CAM_FlashOn.....	196
4.3.14	CAM_FlashOff .....	197
4.3.15	CAM_RawData .....	197
4.3.16	CAM_InsertExifInformation .....	198
4.3.17	CAM_UseGPSExifData .....	199
4.3.18	CAM_RegisterMsgWnd.....	199
4.3.19	CAM_GetScannerType.....	200
4.3.20	CAM_GetVersion.....	200
4.4	CAMERA (WM).....	202
4.4.1	CAM_Open.....	206
4.4.2	CAM_Close.....	206
4.4.3	CAM_SetPreviewWindow .....	207
4.4.4	CAM_PreviewStart .....	208
4.4.5	CAM_PreviewStop.....	208
4.4.6	CAM_GetRegCapturePath .....	209
4.4.7	CAM_GetRegCapturePathEx .....	210
4.4.8	CAM_SetRegCapturePath .....	210
4.4.9	CAM_Capture .....	211
4.4.10	CAM_CaptureEx .....	212
4.4.11	CAM_EnableShutterSound .....	213
4.4.12	CAM_VideoStart.....	213
4.4.13	CAM_VideoStop .....	214
4.4.14	CAM_VideoStopEx.....	215
4.4.15	CAM_GetFlashState.....	215
4.4.16	CAM_AlwaysFlash .....	216



4.4.17	CAM_CaptureFlash .....	217
4.4.18	CAM_AutoFocus .....	217
4.4.19	CAM_SetAfMode .....	218
4.4.20	CAM_Zoom .....	218
4.4.21	CAM_SetResolution .....	219
4.4.22	CAM_GetResolution .....	220
4.4.23	CAM_SetQuality .....	220
4.4.24	CAM_GetQuality .....	221
4.4.25	CAM_SetBrightness .....	221
4.4.26	CAM_GetBrightness .....	222
4.4.27	CAM_SetWhiteBalance .....	222
4.4.28	CAM_GetWhiteBalance .....	223
4.4.29	CAM_SetContrast .....	224
4.4.30	CAM_GetContrast .....	224
4.4.31	CAM_SetSharpness .....	225
4.4.32	CAM_GetSharpness .....	225
4.4.33	CAM_SetHistoEqual .....	226
4.4.34	CAM_GetHistoEqual .....	226
4.4.35	CAM_RegisterPreview .....	227
4.4.36	CAM_RegisterMsgWnd .....	228
4.4.37	CAM_InsertExifInformation .....	228
4.4.38	CAM_GetRegExifEnable .....	229
4.4.39	CAM_UseGPSExifData .....	229
4.4.40	CAM_GetRegExifGpsEnable .....	230
4.4.41	CAM_GetRegInsertDateTimeEnable .....	231
4.4.42	CAM_SetInsertDateTimeEnable .....	231
4.4.43	CAM_GetModelType .....	232
4.4.44	CAM_GetScannerType .....	232
4.4.45	CAM_GetVersion .....	233
4.5	RFID (LF/HF) .....	234
4.5.1	RFID_Open .....	237
4.5.2	RFID_Close .....	237
4.5.3	RFID_PowerSupply .....	238
4.5.4	RFID_GetType .....	239
4.5.5	RFID_SelectTag .....	240
4.5.6	RFID_HighSelectTag .....	240
4.5.7	RFID_LoginTag .....	241
4.5.8	RFID_ReadBlock .....	242
4.5.9	RFID_WriteBlock .....	243

4.5.10	RFID_ReadMultiBlock .....	244
4.5.11	RFID_WriteMultiBlock .....	244
4.5.12	RFID_SetAntenna .....	245
4.5.13	RFID_GetVersion .....	246
4.5.14	RFID_EnableMultiTag .....	247
4.5.15	RFID_SendReadMultiTag .....	247
4.5.16	RFID_SendTransferCommand .....	248
4.5.17	RFID_SendContinuousRead .....	249
4.5.18	RFID_SendCommand .....	250
4.5.19	RFID_SendCommandGetData .....	251
4.5.20	RFID_GetData .....	251
4.5.21	RFID_CheckResult .....	252
4.5.22	RFID_SoundPlay .....	253
4.5.23	RFID_SetTagType .....	254
4.5.24	RFID_GetTagType .....	255
4.5.25	RFID_GetTagTypeToString .....	255
4.5.26	RFID_TagItInventory .....	256
4.5.27	RFID_TagItReadBlock .....	256
4.5.28	RFID_TagItWriteBlock .....	257
4.6	UHF GUN READER .....	259
4.6.1	UHF_GetError .....	264
4.6.2	UHF_IsReady .....	265
4.6.3	UHF_Init .....	265
4.6.4	UHF_Close .....	266
4.6.5	UHF_Inventory .....	267
4.6.6	UHF_InventoryStop .....	268
4.6.7	UHF_Read .....	268
4.6.8	UHF_Write .....	269
4.6.9	UHF_Lock .....	271
4.6.10	UHF_Kill .....	272
4.6.11	UHF_GetData .....	273
4.6.12	UHF_SetRegionFrequency .....	274
4.6.13	UHF_GetRegionFrequency .....	275
4.6.14	UHF_ReadBattery .....	275
4.6.15	UHF_ReadBatteryStatus .....	276
4.6.16	UHF_Version .....	277
4.7	WLAN .....	279
4.7.1	WLAN_ActivateConfig .....	282
4.7.2	WLAN_Close .....	282

4.7.3	WLAN_ConnectAP .....	283
4.7.4	WLAN_ConnectAPEX .....	284
4.7.5	WLAN_DeleteConfig .....	285
4.7.6	WLAN_ExportConfig .....	286
4.7.7	WLAN_ImportConfig .....	286
4.7.8	WLAN_Init .....	287
4.7.9	WLAN_GetAllConfigName .....	287
4.7.10	WLAN_GetCurrentAPIInfo .....	288
4.7.11	WLAN_GetBssidList .....	289
4.7.12	WLAN_GetPowerStatus .....	289
4.7.13	WLAN_PowerOn .....	290
4.7.14	WLAN_PowerOff .....	291
4.8	BLUETOOTH .....	292
4.8.1	BLUETOOTH_Close .....	302
4.8.2	BLUETOOTH_Connect .....	302
4.8.3	BLUETOOTH_CreateConnection .....	303
4.8.4	BLUETOOTH_CreateConnectionEx .....	303
4.8.5	BLUETOOTH_DeleteConnection .....	304
4.8.6	BLUETOOTH_Disconnect .....	305
4.8.7	BLUETOOTH_FindConnectionClose .....	305
4.8.8	BLUETOOTH_FindDeviceClose .....	306
4.8.9	BLUETOOTH_FindFirstConnection .....	307
4.8.10	BLUETOOTH_FindFirstConnectionEx .....	307
4.8.11	BLUETOOTH_FindFirstDevice .....	308
4.8.12	BLUETOOTH_FindFirstDeviceEx .....	309
4.8.13	BLUETOOTH_FindFirstService .....	310
4.8.14	BLUETOOTH_FindFirstServiceEx .....	311
4.8.15	BLUETOOTH_FindLocalDevice .....	312
4.8.16	BLUETOOTH_FindNextConnection .....	312
4.8.17	BLUETOOTH_FindNextConnectionEx .....	313
4.8.18	BLUETOOTH_FindNextDevice .....	314
4.8.19	BLUETOOTH_FindNextService .....	315
4.8.20	BLUETOOTH_FindNextServiceEx .....	315
4.8.21	BLUETOOTH_FindServiceClose .....	316
4.8.22	BLUETOOTH_GetBluetoothState .....	317
4.8.23	BLUETOOTH_GetSCOConnectionState .....	317
4.8.24	BLUETOOTH_GetSecurityMode .....	318
4.8.25	BLUETOOTH_Open .....	319
4.8.26	BLUETOOTH_PerformAction .....	319

4.8.27	BLUETOOTH_SetAuthenticationCallback .....	320
4.8.28	BLUETOOTH_SetBluetoothState .....	321
4.8.29	BLUETOOTH_SetConnectionCallback .....	322
4.8.30	BLUETOOTH_SetIncomingPIN .....	323
4.8.31	BLUETOOTH_SetOutgoingPIN .....	324
4.8.32	BLUETOOTH_SetSecurityMode .....	325
4.9	GPS .....	327
4.9.1	GPS_AGPSDown .....	330
4.9.2	GPS_AGPSRun .....	330
4.9.3	GPS_Close .....	331
4.9.4	GPS_EnableStaticMode .....	332
4.9.5	GPS_ModuleRestart .....	332
4.9.6	GPS_Open .....	333
4.10	SYSTEM .....	335
4.10.1	SYSTEM_BacklightOn .....	339
4.10.2	SYSTEM_Cleanboot .....	339
4.10.3	SYSTEM_GetBacklightlevel .....	340
4.10.4	SYSTEM_GetBacklightTimeOut .....	341
4.10.5	SYSTEM_GetBatteryLifePercent .....	342
4.10.6	SYSTEM_GetBatteryState .....	343
4.10.7	SYSTEM_GetBluetooth .....	343
4.10.8	SYSTEM_GetCpuClock .....	344
4.10.9	SYSTEM_GetDeviceInfo .....	345
4.10.10	SYSTEM_GetGUID .....	346
4.10.11	SYSTEM_GetGSensorValue .....	347
4.10.12	SYSTEM_GetOSVersionInfo .....	348
4.10.13	SYSTEM_GetPhoneVolumeLevel .....	349
4.10.14	SYSTEM_GetPowerTimeOut .....	350
4.10.15	SYSTEM_GetSerialNumber .....	350
4.10.16	SYSTEM_GetVersionInfo .....	351
4.10.17	SYSTEM_GetVolumeLevel .....	352
4.10.18	SYSTEM_GetWlan .....	353
4.10.19	SYSTEM_KeypadLock .....	354
4.10.20	SYSTEM_Reboot .....	354
4.10.21	SYSTEM_SetBacklightlevel .....	355
4.10.22	SYSTEM_SetBacklightTimeOut .....	355
4.10.23	SYSTEM_SetBluetooth .....	356
4.10.24	SYSTEM_SetCpuClock .....	357
4.10.25	SYSTEM_SetGSensorValue .....	358

4.10.26	SYSTEM_SetPhoneVolumeLevel .....	358
4.10.27	SYSTEM_SetPowerTimeOut .....	359
4.10.28	SYSTEM_SetSleepMode .....	360
4.10.29	SYSTEM_SetVolumeLevel.....	360
4.10.30	SYSTEM_SetWlan .....	361
4.10.31	SYSTEM_Vibrate .....	361
4.10.32	SYSTEM_VolumeMute .....	362
5	Demo Manual.....	364
5.1	WLAN .....	364
5.2	BLUETOOTH.....	367
5.3	POS.....	368
5.4	SYSTEM.....	370

# 1 Introduction

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

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

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

## 2 Samples

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

Module	Demo Application	Sample Source	Version	Date
SCANNER	ScanTest.exe Scanner.dll	SCANNER_TEST	1.1.1 1.2.3	2013-10-14 2013-09-25
IMAGER	ImagerTest.exe Imager.dll	IMAGER_TEST	1.1.1 1.0.4	2013-10-14 2013-10-25
CAMERA_CE	CamTest.exe CAM.dll	CAMERA_TEST	1.2.0 1.2.0	2013-10-10 2013-09-09
CAMERA_WM	CamTest.exe CAM.dll	CAMERA_TEST	1.2.0 1.1.1 (1001)	2013-08-07 2013-04-26
RFID	RfidTest.exe RFID.dll	RFID_TEST	1.1.0 1.0.4	2013-10-17 2013-03-18
UGR GUN READER	UHF_GunTest.exe RFID_UHF.dll	RFID_TEST	1.0.1	2014-04-17
WLAN	WlanTest.exe WLAN.dll	WLAN_TEST	1.0.1 1.1.5	2013-08-30 2013-10-10
BLUETOOTH	BluetoothTest.exe BLUETOOTH.dll	BLUETOOTH_TEST	1.0.1 1.0.0	
GPS	GpsTest.exe GPS.dll	GPS_TEST	1.0.1 1.0.1	
SYSTEM	SystemTest.exe M3System.dll	SYSTEM_TEST	1.1.4 1.1.6	2013-10-04 2013-10-10

### 3 Tutorial

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

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



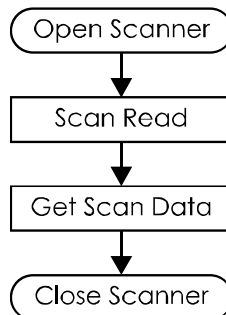
RFID	Open Rfid Close Rfid Start or Stop scanning Tag
------	---

## 3.1 SCANNER (1D)

Supported PDA:

Brand	Restriction
M3 OX10	No restriction

Basic scanner flow chart:



### Tip !!

- I. Scanner communicates through serial so that other programs cannot use scanner while scanner is being run. For example, it occurs an error when other program access scanner while ScanEmul is running.
- II. Combined version of Imager.dll can be available for all 1D Scanner regardless of model type, Scanner Engine and OS.
- III. Its Virtual Keys are VK\_F22 for WinCE and VK\_F14 for Windows Mobile.
- IV. Scan data can be obtained through Windows Message and it can be checked in Scanner.h.  
#define WM\_SCAN\_DATA WM\_APP+350

### Open Scanner

```
// Scanner Open
if(SCAN_Open() == FALSE)
{
    ::MessageBox(NULL, L"Error : SCAN_Open()", NULL, MB_TOPMOST);
}
m_DeviceType = SCAN_GetDeviceType();
// Set Scanner Key
if((m_DeviceType == DEVICE_M3SMART_CE) || (m_DeviceType == DEVICE_M3GREEN) || (m_DeviceType == DEVICE_M3T)
|| (m_DeviceType == DEVICE_M3POS))
{
    // Only for WinCE
    MoveWindow(0, 0, 240, 320);
    m_bWinCE = TRUE;
    RegisterHotKey(this->m_hWnd, HOTKEY_ONE, MOD_KEYUP, VK_F22);
    RegisterHotKey(this->m_hWnd, HOTKEY_TWO, MOD_KEYUP, 0);
}
```

```

        RegisterHotKey(this->m_hWnd, HOTKEY_THREE, MOD_KEYUP, 0);
    }
else
{
    m_bWinCE = FALSE;
    RegisterHotKey(this->m_hWnd, HOTKEY_ONE, MOD_KEYUP, VK_F14);
    RegisterHotKey(this->m_hWnd, HOTKEY_TWO, MOD_KEYUP, 0);
    RegisterHotKey(this->m_hWnd, HOTKEY_THREE, MOD_KEYUP, 0);
}

```

#### Scan Read

```

void CScanTestDlg::OnBnClickedBtnScan()
{
    SCAN_Read();
}

```

#### Get ScanData

```

BEGIN_MESSAGE_MAP(CScanTestDlg, CDialog)
    //}}AFX_MSG_MAP
    ON_MESSAGE(WM_SCAN_DATA, OnScanRead)
END_MESSAGE_MAP()

:

LRESULT CScanTestDlg::OnScanRead(WPARAM wParam, LPARAM lParam)
{
    if(m_bNewForm == TRUE)
        return FALSE;

    TCHAR szBarType[1024] = {0, };
    TCHAR szBarData[1024] = {0, };
    SCAN_GetScanData(szBarType, szBarData);
    m_strStaticType = szBarType;
    m_strEditData = szBarData;
    UpdateData(FALSE);
    return LRESULT();
}

```

#### Close Scanner

```

void CScanTestDlg::OnDestroy()
{
    CDialog::OnDestroy();
    UnregisterHotKey(this->m_hWnd, HOTKEY_ONE);
    UnregisterHotKey(this->m_hWnd, HOTKEY_TWO);
    UnregisterHotKey(this->m_hWnd, HOTKEY_THREE);
    SCAN_Close();
}

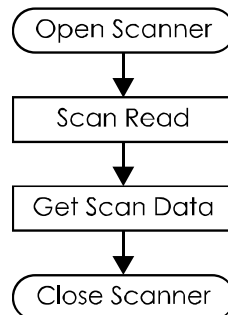
```

## 3.2 IMAGER (2D)

Supported PDA:

Brand	Restriction
M3 OX10	No restriction

Basic imager flow chart:



### Tip !!

- I. Imager Driver and Camera Driver use the same Quick Capture Interface of CPU. This interface cannot be used at the same time. Because of unloading imager driver while running the camera program so that Imager cannot be used.
- II. Its Virtual Keys are VK\_F22 for WinCE and VK\_F14 for Windows Mobile.
- III. Scan Data can be obtained through Windows Message and it can be checked in Scanner.h.  
#define WM\_SCAN\_DATA WM\_APP+350

### Open Scanner

```
// Scanner Open
if(IMAGER_Open () == FALSE)
{
    ::MessageBox(NULL, L"Error : IMAGER_Open ()", NULL, MB_TOPMOST);
}
m_DeviceType = IMAGER_GetDeviceType ();
// Enable All Symbologies
IMAGER_SetSymbologyAll();
// Set Scanner Key
if((m_DeviceType == DEVICE_M3SMART_CE) || (m_DeviceType == DEVICE_M3GREEN) || (m_DeviceType == DEVICE_M3T)
|| (m_DeviceType == DEVICE_M3POS))
{
    // Only for WinCE
    MoveWindow(0, 0, 240, 320);
    m_bWinCE = TRUE;
    RegisterHotKey(this->m_hWnd, HOTKEY_ONE, MOD_KEYUP, VK_F22);
    RegisterHotKey(this->m_hWnd, HOTKEY_TWO, MOD_KEYUP, 0);
    RegisterHotKey(this->m_hWnd, HOTKEY_THREE, MOD_KEYUP, 0);
}
```

```

}
else
{
    m_bWinCE = FALSE;
    RegisterHotKey(this->m_hWnd, HOTKEY_ONE, MOD_KEYUP, VK_F14);
    RegisterHotKey(this->m_hWnd, HOTKEY_TWO, MOD_KEYUP, 0);
    RegisterHotKey(this->m_hWnd, HOTKEY_THREE, MOD_KEYUP, 0);
}

```

#### Scan Read

```

void CImagerTestDlg::OnBnClickedBtnScan()
{
    IMAGER_Read();
}

```

#### Get ScanData

```

BEGIN_MESSAGE_MAP(CScanTestDlg, CDialog)
    //}}AFX_MSG_MAP
    ON_MESSAGE(WM_SCAN_DATA, OnScanRead)
END_MESSAGE_MAP()

:

LRESULT CImagerTestDlg::OnScanRead(WPARAM wParam, LPARAM lParam)
{
    if(m_bNewForm == TRUE)
        return FALSE;

    TCHAR szBarType[1024] = {0, };
    TCHAR szBarData[1024] = {0, };
    IMAGER_GetScanData(szBarType, szBarData);
    m_strStaticType = szBarType;
    m_strEditData = szBarData;
    UpdateData(FALSE);
    return LRESULT();
}

```

#### Close Scanner

```

void CImagerTestDlg::OnDestroy()
{
    CDialog::OnDestroy();
    UnregisterHotKey(this->m_hWnd, HOTKEY_ONE);
    UnregisterHotKey(this->m_hWnd, HOTKEY_TWO);
    UnregisterHotKey(this->m_hWnd, HOTKEY_THREE);
    if(IMAGER_Close() == FALSE)
        ::MessageBox(NULL, L"ERROR : IMAGER_Close()", NULL, MB_TOPMOST);
}

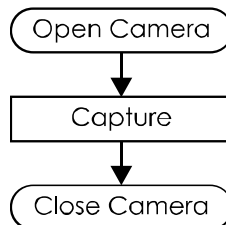
```

### 3.3 CAMERA (CE)

Supported PDA:

Brand	Restriction
M3 OX10	No restriction

Basic camera flow chart:



#### Tip !!

- I. Its Virtual Key is VK\_23.
- II. File name can be editable when capturing pictures. File is stored with file name that users input, otherwise it will be stored by date as default.
- III. CapStatusEvent function is for current status of capturing pictures and name of picture can be obtained through CAM\_GetLastSaveFilePath function when capturing is done completely.
- IV. EXIF information can be inserted when enabling option for EXIF information. GPS information can be included in EXIF information and for that, GPS supported device is required. After enabling option for GPS information, GPS information can be confirmed when receiving the grid information from the satellite.
- V. Self Auto-Focus function can be used in devices that AF function is available such as M3 T. It recognizes changes in the preview screen automatically and perform focusing automatically.

Open Camera
<pre>MODEL_TYPE model = CAM_Open(m_hWnd, m_ctrPreview.m_hWnd) ; if(m_Model == MODEL_UNKNOWN) {     ::MessageBox(NULL, L"Open error", L"Open", NULL);     return; }</pre>
Capture Still
<pre>TCHAR m_tzSavePath[256] = {0x00}; if(!m_bCapturing) {     CAM_Capture(m_tzSavePath); }</pre>
Preview

```
BOOL PreviewStart(BOOL bOn)
```

```
{
    if(bOn == TRUE)
    {
        CAM_PreviewStart();
    }
    else
    {
        CAM_PreviewStop();
    }
}
```

#### Insert Exif Information

```
#define WM_SHOW_CAP_STATUS      (WM_USER + 5)
BEGIN_MESSAGE_MAP(CCamTestDlg, CDialog)
    ON_MESSAGE(WM_SHOW_CAP_STATUS, OnShowCaptureStatus)
END_MESSAGE_MAP()
BOOL Init()
{
    HANDLE m_hCapStatusEvent = CreateEvent(NULL, FALSE, 0, _T("CapStatusEvent"));
    if (m_hCapStatusEvent == NULL)
        return FALSE;
}
DWORD ThreadFuncCapStatus(LPVOID pParam)
{
    DWORD CapStatus;
    while(!m_bThreadCapStatusExit) {
        WaitForSingleObject(m_hCapStatusEvent, INFINITE);
        if (m_bThreadCapStatusExit) {
            return 0;
        }
        CapStatus = GetEventData(m_hCapStatusEvent);
        PostMessage(WM_SHOW_CAP_STATUS, CapStatus, 0);
    }
    return 0;
}
LRESULT CCamTestDlg::OnShowCaptureStatus(WPARAM wParam, LPARAM lParam)
{
    switch (wParam) {
        case CAP_STATUS_END:
            m_bCapturing= FALSE;
            TCHAR tzFileName[MAX_PATH];
            memset(tzFileName, 0x00, sizeof(TCHAR) * MAX_PATH);
            CAM_GetLastSaveFilePath(tzFileName);
            if(_tcsstr(tzFileName, _T(".jpg"))!=NULL)
```

```

        if(m_onEXIF)
            InsertExifInform();

        break;

    default:
        break;
}
return 0L;
}

BOOL InsertExifInform()
{
    ExifInfo info={0,};
    TCHAR OutFileBuf[260]={0,};
    TCHAR tzFileName[260]={0,};
    if(!CAM_GetLastSaveFilePath(tzFileName))
        return FALSE;
    _tcscpy(OutFileBuf, tzFileName);
    TCHAR* ImageName;
    ImageName=_tcstok(OutFileBuf, _T("\\"));
    while (ImageName!=NULL)
    {
        if(!_tcsstr(ImageName, _T(".jpg"))!=NULL || _tcsstr(ImageName, _T(".JPG"))!=NULL)
            break;
        ImageName=_tcstok(NULL, _T("\\"));
    }
    _tcscpy(info.TitleName, ImageName);
    _tcscpy(info.Make, _T("M3 Mobile Co.Ltd"));
    switch(m_Model)
    {
        case MODEL_SMART:
            _tcscpy(info.Model, _T("M3 Smart"));
            break;
        case MODEL_POS:
            _tcscpy(info.Model, _T("M3 POS"));
            break;
        case MODEL_GREEN:
            _tcscpy(info.Model, _T("M3 Green"));
            break;
        case MODEL_T:
            _tcscpy(info.Model, _T("M3 T"));
            break;
    }
    if(CAM_InsertExifInformation(tzFileName, info))
    {
        return TRUE;
    }
}

```



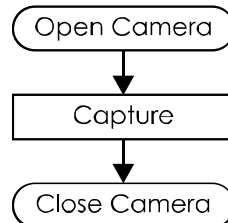
<pre>else {     return FALSE; } }</pre>
Close Camra
<pre>CAM_Close();</pre>

### 3.4 CAMERA (WM)

Supported PDA:

Brand	Restriction
M3 OX10	No restriction

Basic camera flow chart:



#### Tip !!

- I. Combined version of Camera.dll for Windows Mobile OS.
- II. Its Virtual Key is VK\_13.
- III. File name can be editable when capturing pictures. File is stored with file name that users input, otherwise it will be stored by date as default.
- IV. EXIF information can be inserted when enabling option for EXIF information. GPS information can be included in EXIF information and for that, GPS supported device is required. After enabling option for GPS information, GPS information can be confirmed when receiving the grid information from the satellite.
- V. Self Auto-Focus function can be used in devices that AF function is available such as M3 SKY and M3 ORANGE. It recognizes changes in the preview screen automatically and perform focusing automatically.

Open Camera
<pre>m_CameraMode = STILL_MODE;    // or VIDEO_MODE m_VideoType = VIDEO_WMV;      // or VIDEO_ASF CAM_Open(m_hWnd, m_CameraMode, m_VideoType);</pre>
Capture Still
<pre>TCHAR tzFile[MAX_PATH] = {0x00}; TCHAR tzOutFile[MAX_PATH] = {0x00}; memset(tzFile, 0x00, sizeof(TCHAR) * MAX_PATH); memset(tzOutFile, 0x00, sizeof(TCHAR) * MAX_PATH); wsprintf(tzFile, L"\\Flash Disk\\Cam\\Picture1.jpg"); m_bAutoInit = TRUE; CAM_Capture(tzFile, tzOutFile, m_bAutoInit);</pre>
Preview
<pre>BOOL PreviewStart(BOOL bStart)</pre>

```

{
    if(bStart)
    {
        CAM_PreviewStart();
    }
    else
    {
        CAM_PreviewStop();
    }
}

```

#### Insert Exif Information

```

BOOL InsertExifInform(TCHAR * tzFileName)
{
    ExifInfo info={0,};
    TCHAR OutFileBuf[260]={0,};
    _tcscpy(OutFileBuf, tzFileName);
    TCHAR* ImageName;
    ImageName=_tcstok(OutFileBuf, _T("\\"));
    while (ImageName!=NULL)
    {
        if(_tcsstr(ImageName, _T(".jpg"))!=NULL || _tcsstr(ImageName, _T(".JPG"))!=NULL)
            break;
        ImageName=_tcstok(NULL, _T("\\"));
    }
    _tcscpy(info.TitleName, ImageName);
    _tcscpy(info.Make, _T("M3 Mobile Co.Ltd"));
    switch(m_ModelType)
    {
        case MODEL_MM3:
            _tcscpy(info.Model, _T("MM3"));
            break;
        case MODEL_SMART:
            _tcscpy(info.Model, _T("M3 Smart"));
            break;
        case MODEL_SKY:
            _tcscpy(info.Model, _T("M3 Sky"));
            break;
        case MODEL_ORANGE:
            _tcscpy(info.Model, _T("M3 Orange"));
            break;
        case MODEL_UNKONWN:
            _tcscpy(info.Model, _T("M3"));
            break;
    }
    if(CAM_InsertExifInformation(tzFileName, info))

```

```
{  
    RETAILMSG(RT_MSG, (_T("CamTest - Insert EXIF Success!!\n")));  
    return TRUE;  
}  
else  
{  
    return FALSE;  
}  
}
```

Close Camra

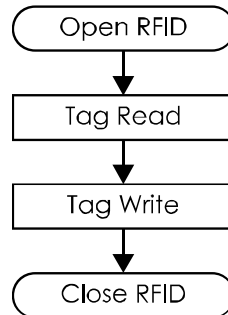
CAM\_Close();

### 3.5 RFID (LF/HF)

Supported PDA:

Brand	Restriction
M3 O10	HF RFID modules is installed. UHF is not supported

Basic RFID flow chart:



#### Tip !!

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

#### Open RFID

```
RFID_TYPE m_RfType = RFID_Open();
TCHAR tzRadioType[256] = {0x00};
if(m_RfType == RFID_LF)
{
    wsprintf(tzRadioType, L"Low Frequency");
}
else if(m_RfType == RFID_HF)
{
    wsprintf(tzRadioType, L"High Frequency");
}
else
{
    return;
}
```

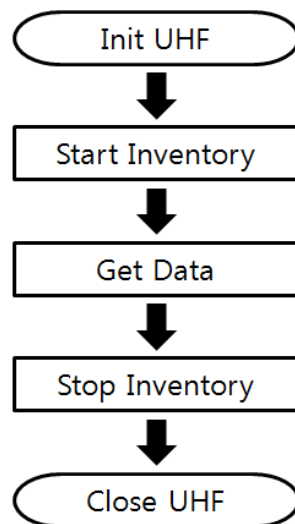
Antenna On/Off
<pre>if(bAntennaOn == TRUE) {     RFID_SetAntenna(TRUE); } else {     RFID_SetAntenna(FALSE); }</pre>
Tag Select
<pre>TCHAR tzSerial[100] = {0x00}; RFID_SelectTag(tzSerial);</pre>
Data Read
<pre>TCHAR tzSerial[32] ={0x00}; TCHAR tzData[nMaxData] = {0x00}; memset(tzSerial, 0x00, sizeof(TCHAR) * 32); memset(tzData, 0x00, sizeof(TCHAR) * nMaxData); if(RFID_SelectTag(tzSerial)) {     RFID_ReadBlock(L"01", tzData); }</pre>
Data Write
<pre>TCHAR tzSerial[32] ={0x00}; TCHAR tzOutData[nMaxData] = {0x00}; memset(tzSerial, 0x00, sizeof(TCHAR) * 32); memset(tzOutData, 0x00, sizeof(TCHAR) * 1024); if(RFID_SelectTag(tzSerial)) {     RFID_WriteBlock(L"01", L"00112233", tzOutData); }</pre>

### 3.6 UHF Gun Reader

Supported PDA:

Brand	Restriction
M3 O10	HF RFID modules is installed. UHF is 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:
  - PDA detached from gun reader
  - RFID / SCAN switch is at SCAN position
  - Gun reader's battery is detached or empty
  - 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
<pre>BEGIN_MESSAGE_MAP(CUHF_GunTestDlg, CDialog)     ON_MESSAGE(WM_MSG_POWER, ReceivedPower) END_MESSAGE_MAP()</pre>

```

BOOL Init()
{
    RFID_STATUS status;
    CString strstatus;

    status = UHF_Init(m_hWnd);
    if(status != RFID_STATUS_OK)
    {
        strstatus.Format(_T("RFID Init Error-[%x]"),status);
        // MessageBox(strstatus);

        return FALSE;
    }
    return TRUE;
}

LRESULT ReceivedPower(WPARAM wParam, LPARAM lParam)
{
    BOOL* bPowerOn = (BOOL*)wParam;

    if(*bPowerOn)
    {
        if(m_bOpen == FALSE)
        {
            Init();
            m_bOpen = TRUE;
        }
    }
    else
    {
        if(m_bOpen == TRUE)
        {
            Close();
            m_bOpen = FALSE;
        }
    }

    return 0;
}

```

#### Start Inventory

```

void Inventory()
{
    UHF_Inventory(m_hWnd);
}

```



## Get Data

```
BEGIN_MESSAGE_MAP(CUHF_GunTestDlg, CDialog)
    ON_MESSAGE(WM_MSG_INVENTORY, ReceivedInventory)
END_MESSAGE_MAP()

LRESULT ReceivedInventory(WPARAM wParam, LPARAM lParam)
{
    int nTaglength = 0;
    unsigned char czTagData[128] = {0};
    CString strTagData;

    nTaglength = UHF_GetData(czTagData);

    if(m_chkCRC.GetCheck())
    {
        for(int i=0;i<nTaglength;i++)
        {
            strTagData.AppendFormat(_T("%02X"), czTagData[i]);
        }
    }
    else
    {
        for(int i=0;i<nTaglength-2;i++)
        {
            strTagData.AppendFormat(_T("%02X"), czTagData[i]);
        }
    }

    return LRESULT();
}
```

## Stop Inventory

```
void InventoryStop()
{
    UHF_InventoryStop();
}
```

## Close UHF

```
BOOL Close()
{
    RFID_STATUS status;
    CString strstatus;

    status = UHF_Close();
    if(status != RFID_STATUS_OK)
```

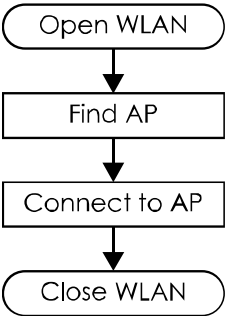
```
{  
    strstatus.Format(_T("%x"),status);  
    // MessageBox(_T("RFID Close Error"), strstatus);  
  
    return FALSE;  
}  
  
// MessageBox(L"Close");  
::SetWindowText(::GetDlgItem(m_hWnd, IDC_STATIC_STATUS), L"Close UHF");  
return TRUE;  
}
```

### 3.7 WLAN

Supported PDA:

Brand	Restriction
M3 OX10	WLAN must be SUMMIT

Basic WLAN flow chart:



**Tip !!**

WLANTest.exe functions as same as SCU (Summit Client Utility) and they are synchronized.

Combined version of WLAN.dll can be used in all M3 Summit devices regardless of model type or OS.

Init Wlan
<pre>if(WLAN_Init() == FALSE) {     AfxMessageBox( L"Fail to WLAN_Init()"); }</pre>
Searching AP
<pre>WLAN_SSID_LIST* SSIDList=new WLAN_SSID_LIST(); SSIDList-&gt;m_SSID=new WLAN_SSID[40]; do {     WLAN_GetBssidList(SSIDList); } while(SSIDList-&gt;m_NumberOfItems==0);</pre>
Connect Config
<pre>int result=0; // Security result=WLAN_ConnectAPEX (_T("SSID"), _T("12345"), Encryption Type, 2); if(result!=0)     AfxMessageBox(_T("Fail to WLAN_ConnectAPEX ()")); // Open result=WLAN_ConnectAPEX (_T("SSID"), _T("\0"), 0, 2);</pre>

AfxMessageBox(_T("Fail to WLAN_ConnectAPEX()"));
<b>Delete Config</b>
<pre>int result=WLAN_DeleteConfig(_T("SSID"); if(!result)     AfxMessageBox(_T("Activate config can not removed!!"));</pre>
<b>Change Config</b>
<pre>if result=WLAN_ActivateConfig(_T("SSID"); if(!result)     AfxMessageBox(_T("Fail to WLAN_ActivateConfig()"));</pre>
<b>Export Configuration</b>
<pre>TCHAR szFilter[]=_T("SDC files(*.sdc)   *.sdc   All files(*.*)   *.*    "); CFileDialog SaveDlg(FALSE, _T("sdc"), _T("SummitSettings.sdc"), OFN_HIDEREADONLY, szFilter); if (IDOK==SaveDlg.DoModal())     m_filepath=SaveDlg.GetPathName();     if(WLAN_ExportConfig(m_filepath.GetBuffer()))         m_result=_T("Export Success!!");     else         m_result=_T("Export Fail!!"); UpdateData(FALSE);</pre>
<b>WLAN Power On</b>
<pre>BOOL result= WLAN_PowerOn(); if(!result)     AfxMessageBox(_T("Fail to WLAN_PowerOn()"));</pre>
<b>WLAN Power Off</b>
<pre>BOOL result= WLAN_PowerOff(); if(!result)     AfxMessageBox(_T("Fail to WLAN_PowerOff()"));</pre>
<b>Import Configuration</b>
<pre>TCHAR szFilter[]=_T("SDC files(*.sdc)   *.sdc   All files(*.*)   *.*    "); CFileDialog LoadDlg(TRUE, _T("sdc"), _T("SummitSettings.sdc"), OFN_HIDEREADONLY, szFilter); if (IDOK==LoadDlg.DoModal())     m_filepath=LoadDlg.GetPathName();     if(WLAN_ImportConfig(m_filepath.GetBuffer()))         m_result=_T("Import Success!!");     else         m_result=_T("Import Fail!!"); UpdateData(FALSE);</pre>
<b>Close WLAN</b>
<pre>BOOL result=WLAN_Close();</pre>

```
if(!result)
```

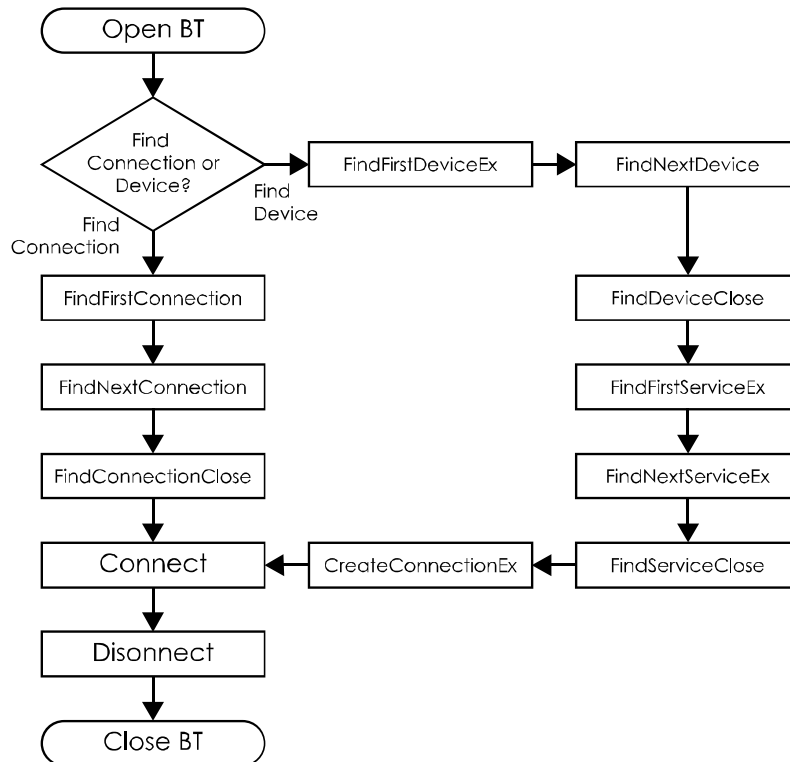
```
    AfxMessageBox(_T("Fail to WLAN_Close()"));
```

## 3.8 BLUETOOTH

Supported PDA:

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

Basic Bluetooth flow chart:



### Tip !!

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

## Initialization

```
// Bluetooth Open
void CBluetoothTestDlg::OnBnClickedBtnOpen()
{
    TCHAR* result = NULL;
    if(BLUETOOTH_Open())
    {
        m_List.DeleteAllItems();
        m_State.SetWindowText(L"Open Success");
        g_bBluetoothOpen = TRUE;
        m_Open.EnableWindow(FALSE);
        m_DeviceFind.EnableWindow(TRUE);
        m_ServiceFind.EnableWindow(FALSE);
        m_CreateConn.EnableWindow(FALSE);
        m_ConnFind.EnableWindow(TRUE);
        m_Connect.EnableWindow(FALSE);
        m_Disconnect.EnableWindow(FALSE);
        m_DeleteConn.EnableWindow(FALSE);
        m_Close.EnableWindow(TRUE);
        m_AllDelDevice.EnableWindow(TRUE);
        m_LocalInfo.EnableWindow(TRUE);
    }
    else
        m_State.SetWindowText(L"Open Fail");
}
```

## Device Find

```
// Bluetooth Device Find
void CBluetoothTestDlg::OnBnClickedBtnDeviceFind()
{
    if(g_bBluetoothOpen == TRUE)
    {
        m_listtype = DeviceFind;
        m_List.DeleteAllItems();
        m_Deviceitem.RemoveAll();
        m_Connectitem.RemoveAll();
        memset(&m_connectionInfo, 0, sizeof(m_connectionInfo));
        m_connectionInfo.Size = sizeof(BTP_Connection_Info_Ex_t);
        m_connectionInfo.ProfileType = BTP_PROFILE_SPP;
        CString strAddr;
        CString strName;
        BTP_Device_Find deviceFind;
        BTP_Device_Info_t deviceInfo;
        BTP_Device_Query_Ex_t deviceQuery;
        memset(&deviceQuery, 0, sizeof(deviceQuery));
    }
}
```

```

deviceQuery.Size = sizeof(BTP_Device_Query_Ex_t);
deviceQuery.DeviceAttributes = BTP_DEVICE_ALL;
deviceQuery.DeviceType = (m_connectionInfo.ProfileType == BTP_PROFILE_SPP ? bdAll : bdHID);
deviceQuery.InquiryTimeout = 10;
BeginWaitCursor();
hResult = BLUETOOTH_FindFirstDeviceEx(&deviceFind, &deviceInfo, &deviceQuery);
if (BTP_ERROR_SUCCESS == hResult)
{
    EndWaitCursor();
    do
    {
        BTP_Device_Info_t deviceInfo_temp;
        memcpy(&deviceInfo_temp, &deviceInfo, sizeof(BTP_Device_Info_t));
        m_Deviceitem.AddTail(deviceInfo_temp);
        hResult = BLUETOOTH_FindNextDevice(deviceFind, &deviceInfo);
    } while (BTP_ERROR_SUCCESS == hResult);
}
BLUETOOTH_FindDeviceClose(deviceFind);
int Importdevicecount = m_Deviceitem.GetCount();
int i = 0;
for(i = Importdevicecount-1; i >= 0; i--)
{
    deviceInfo = m_Deviceitem.GetAt(m_Deviceitem.FindIndex(i));
    strName.Format(L"%s", CString(deviceInfo.Name));
    m_List.InsertItem(0, strName, 0);
    strAddr.Format(L"%02X:%02X:%02X:%02X:%02X:%02X",
        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);
    m_List.SetItem(0, 1, LVIF_TEXT, strAddr, NULL, NULL, NULL, NULL);
}
m_State.SetWindowText(L"Device Find");
g_bAllDelDevice = FALSE;
m_Open.EnableWindow(FALSE);
m_DeviceFind.EnableWindow(FALSE);
m_ServiceFind.EnableWindow(TRUE);
m_CreateConn.EnableWindow(FALSE);
m_ConnFind.EnableWindow(TRUE);
m_Connect.EnableWindow(FALSE);
m_Disconnect.EnableWindow(FALSE);
m_DeleteConn.EnableWindow(FALSE);
m_Close.EnableWindow(TRUE);
m_AllDelDevice.EnableWindow(TRUE);

```



```

        m_LocalInfo.EnableWindow(TRUE);
    }
}

```

## Service Find

```

// Bluetooth Service Find
void CBluetoothTestDlg::OnBnClickedBtnServiceFind()
{
    if(g_bBluetoothOpen == TRUE && g_bAllDelDevice == FALSE)
    {
        m_listtype = ServiceFind;
        m_List.DeleteAllItems();

        BTP_Service_Find      serviceFind;
        BTP_Service_Info_Ex_t  serviceInfo;
        BTP_Service_Query_t    serviceQuery;
        SDP_UUID_Entry_t      service[1];
        CString                strname;

        serviceInfo.Size = sizeof(BTP_Service_Info_Ex_t);
        memset(&serviceQuery, 0, sizeof(serviceQuery));
        serviceQuery.BD_ADDR = m_connectionInfo.BD_ADDR;
        serviceQuery.NumberServiceUUID = 1;
        serviceQuery.Service = service;
        service[0].SDP_Data_Element_Type = deUUID_16;
        switch(m_connectionInfo.ProfileType)
        {
        case BTP_PROFILE_SPP:
            if (128 == serviceUUID)
            {
                service[0].SDP_Data_Element_Type = deUUID_128;
                ASSIGN_UUID_128((service[0].UUID_Value.UUID_128),
                    0x85, 0x60, 0xCA, 0x18,    // 0x8560CA18
                    0x62, 0x3F,                // 0x623f
                    0x4A, 0x77,                // 0x4a77
                    0x9F, 0x5E, 0x3C, 0x52, 0x66, 0x68, 0x05, 0x1A); // 0x9f, 0x5e, 0x3c, 0x52, 0x66, 0x68, 0x05, 0x1a
            }
            else
            {
                ASSIGN_UUID_16((service[0].UUID_Value.UUID_16), 0x11, 0x01)
            }
            break;
        case BTP_PROFILE_HID_HOST:
        case BTP_PROFILE_HID_DEVICE:
            ASSIGN_UUID_16((service[0].UUID_Value.UUID_16), 0x11, 0x24)
            break;
        }
        BeginWaitCursor();
    }
}

```

```

hResult = BLUETOOTH_FindFirstServiceEx(&serviceFind, &serviceInfo, &serviceQuery);
if (BTP_ERROR_SUCCESS == hResult)
{
    EndWaitCursor();
    do
    {
        m_connectionInfo.ProfileType = serviceInfo.ProfileType;
        m_connectionInfo.MajorVersion = serviceInfo.MajorVersion;
        m_connectionInfo.MinorVersion = serviceInfo.MinorVersion;
        switch(m_connectionInfo.ProfileType)
        {
            case BTP_PROFILE_UNKNOWN:
                m_connectionInfo.ProfileInformation.RemoteSerialPortProfileInfo.RFCOMMServerPort =
                serviceInfo.ProfileInformation.RemoteSerialPortProfileInfo.RFCOMMServerPort;

                m_connectionInfo.ProfileInformation.RemoteSerialPortProfileInfo.UseActiveSync =
                serviceInfo.ProfileInformation.RemoteSerialPortProfileInfo.UseActiveSync;

                break;

            case BTP_PROFILE_SPP:
                m_connectionInfo.ProfileInformation.RemoteSerialPortProfileInfo.RFCOMMServerPort =
                serviceInfo.ProfileInformation.RemoteSerialPortProfileInfo.RFCOMMServerPort;

                m_connectionInfo.ProfileInformation.RemoteSerialPortProfileInfo.UseActiveSync =
                serviceInfo.ProfileInformation.RemoteSerialPortProfileInfo.UseActiveSync;

                break;

            case BTP_PROFILE_HID_HOST:
            case BTP_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.Format(L"%s", (CString)serviceInfo.ServiceName);
        strname = serviceInfo.ServiceName;
        m_List.InsertItem(0, strname, 0);

        hResult = BLUETOOTH_FindNextServiceEx(serviceFind, &serviceInfo);
    } while (BTP_ERROR_SUCCESS == hResult);
    if (BTP_ERROR_NO_MORE != hResult)
        m_State.SetWindowText(L"Failed to Find Next Service");

    BLUETOOTH_FindServiceClose(serviceFind);
}

m_State.SetWindowText(L"Service Find");

```

```

        m_Open.EnableWindow(FALSE);
        m_DeviceFind.EnableWindow(TRUE);
        m_ServiceFind.EnableWindow(FALSE);
        m_CreateConn.EnableWindow(TRUE);
        m_ConnFind.EnableWindow(TRUE);
        m_Connect.EnableWindow(FALSE);
        m_Disconnect.EnableWindow(FALSE);
        m_DeleteConn.EnableWindow(FALSE);
        m_Close.EnableWindow(TRUE);
        m_AllDelDevice.EnableWindow(FALSE);
        m_LocalInfo.EnableWindow(TRUE);
    }
    else if(g_bAllDelDevice == TRUE)
    {
        m_State.SetWindowText(L"Please,'Device Find' is re-click.");
    }
}

```

### Connection Management

```

// Bluetooth Connection Management - (1) Create Connection
void CBluetoothTestDlg::OnBnClickedBtnCreateConnection()
{
    if(g_bBluetoothOpen == TRUE)
    {
        m_List.DeleteAllItems();
        m_connectionInfo.ConnectionAttributes = BTP_CONNECTION_REMEMBERED | BTP_CONNECTION_ACTIVE;
        m_connectionInfo.LocalCOMPort = 9;
        m_connectionInfo.ProfileType = BTP_PROFILE_SPP;
        hResult = BLUETOOTH_CreateConnectionEx(&m_connectionInfo);
        if (BTP_ERROR_SUCCESS == hResult)
            m_State.SetWindowText(L"Create connection");
        else
        {
            m_State.SetWindowText(L"Create Connection Error");
            m_Open.EnableWindow(FALSE);
            m_DeviceFind.EnableWindow(TRUE);
            m_ServiceFind.EnableWindow(TRUE);
            m_CreateConn.EnableWindow(FALSE);
            m_ConnFind.EnableWindow(TRUE);
            m_Connect.EnableWindow(FALSE);
            m_Disconnect.EnableWindow(FALSE);
            m_DeleteConn.EnableWindow(FALSE);
            m_Close.EnableWindow(TRUE);
            m_AllDelDevice.EnableWindow(FALSE);
            m_LocalInfo.EnableWindow(TRUE);
        }
    }
}

```

```

// Bluetooth Connection Management - (2) Delete Connection
void CBluetoothTestDlg::OnBnClickedBtnDeleteConnection()
{
    if(g_bBluetoothOpen == TRUE)
    {
        m_List.DeleteAllItems();
        hResult = BLUETOOTH_DeleteConnection(m_connectionInfo.ConnectionID);
        if (BTP_ERROR_SUCCESS == hResult)
            m_State.SetWindowText(L"Delete Connection");
        else
            m_State.SetWindowText(L"Delete Connection Error");
        m_Open.EnableWindow(FALSE);
        m_DeviceFind.EnableWindow(TRUE);
        m_ServiceFind.EnableWindow(FALSE);
        m_CreateConn.EnableWindow(FALSE);
        m_ConnFind.EnableWindow(TRUE);
        m_Connect.EnableWindow(FALSE);
        m_Disconnect.EnableWindow(FALSE);
        m_DeleteConn.EnableWindow(FALSE);
        m_Close.EnableWindow(TRUE);
        m_AllDelDevice.EnableWindow(TRUE);
        m_LocalInfo.EnableWindow(TRUE);
    }
}

// Bluetooth Connection Management - (3) Connection Find
void CBluetoothTestDlg::OnBnClickedBtnConnectionFind()
{
    if(g_bBluetoothOpen == TRUE)
    {
        m_List.DeleteAllItems();
        m_Deviceitem.RemoveAll();
        m_Connectitem.RemoveAll();
        m_listtype = ConnectionFind;
        CString strID;
        CString strAddr;
        BTP_Connection_Find    connectFind;
        BTP_Connection_Info_t  connectionInfo;
        BTP_Connection_Query_t connectQuery;
        BTP_Connection_Info_t  connectInfo_temp;
        memset(&connectQuery, 0, sizeof(connectQuery));
        connectQuery.ConnectionAttributes = BTP_CONNECTION_REMEMBERED;
        BeginWaitCursor();
        hResult = BLUETOOTH_FindFirstConnection(&connectFind, &connectionInfo, &connectQuery);
        if(BTP_ERROR_NO_MORE == hResult)
        {
            EndWaitCursor();
        }
    }
}

```

```

        m_State.SetWindowText(L"PDA doesn't have a 'Connection'.");
        return;
    }
    else if (BTP_ERROR_SUCCESS == hResult)
    {
        EndWaitCursor();
        do
        {
            memcpy(&connectInfo_temp,&connectionInfo,sizeof(BTP_Connection_Info_t));
            m_Connectitem.AddTail(connectInfo_temp);
            hResult = BLUETOOTH_FindNextConnection(connectFind, &connectionInfo);
        } while (BTP_ERROR_SUCCESS == hResult);
    }
    BLUETOOTH_FindConnectionClose(connectFind);
    int ImportDeviceCount = m_Connectitem.GetCount();
    int i = 0;
    for(i = ImportDeviceCount-1;i>=0;i--)
    {
        connectionInfo = m_Connectitem.GetAt(m_Connectitem.FindIndex(i));
        strID.Format(L"%d", connectionInfo.ConnectionID);
        m_List.InsertItem(0, strID, 0);
        strAddr.Format(L"%02X:%02X:%02X:%02X:%02X:%02X",
            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);
        m_List.SetItem(0, 1, LVIF_TEXT, strAddr, NULL, NULL, NULL, NULL);
    }
    m_State.SetWindowText(L"Connection Find");
    m_Open.EnableWindow(FALSE);
    m_DeviceFind.EnableWindow(FALSE);
    m_ServiceFind.EnableWindow(FALSE);
    m_CreateConn.EnableWindow(FALSE);
    m_ConnFind.EnableWindow(FALSE);
    m_Connect.EnableWindow(TRUE);
    m_Disconnect.EnableWindow(FALSE);
    m_DeleteConn.EnableWindow(TRUE);
    m_Close.EnableWindow(TRUE);
    m_AllDelDevice.EnableWindow(FALSE);
    m_LocalInfo.EnableWindow(TRUE);
}
}

```

## Connect

```
// Bluetooth Connect
void CBluetoothTestDlg::OnBnClickedBtnConnect()
{
    if(g_bBluetoothOpen == TRUE)
    {
        m_List.DeleteAllItems();
        hResult = BLUETOOTH_Connect(m_connectionInfo.ConnectionID);
        CreateFile((LPCWSTR)"COM9:", GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL,
        NULL);
        if(BTP_ERROR_SUCCESS == hResult)
        {
            m_State.SetWindowText(L"Connect");
        }
        else if(BTP_ERROR_INVALID_PARAMETER == hResult)
        {
            m_State.SetWindowText(L"Error Invalid parameter");
        }
        else
        {
            m_State.SetWindowText(L"Connect Error");
        }
        m_Open.EnableWindow(FALSE);
        m_DeviceFind.EnableWindow(TRUE);
        m_ServiceFind.EnableWindow(FALSE);
        m_CreateConn.EnableWindow(FALSE);
        m_ConnFind.EnableWindow(FALSE);
        m_Connect.EnableWindow(FALSE);
        m_Disconnect.EnableWindow(TRUE);
        m_DeleteConn.EnableWindow(FALSE);
        m_Close.EnableWindow(TRUE);
        m_AllDelDevice.EnableWindow(FALSE);
        m_LocalInfo.EnableWindow(TRUE);
    }
}
```

## Disconnect

```
// Bluetooth Disconnect
void CBluetoothTestDlg::OnBnClickedBtnDisconnect()
{
    if(g_bBluetoothOpen == TRUE)
    {
        m_List.DeleteAllItems();
        hResult = BLUETOOTH_Disconnect(m_connectionInfo.ConnectionID);
        if(BTP_ERROR_SUCCESS == hResult)
```

```

{
    m_State.SetWindowText(L"Disconnect");
}
else if(BTP_ERROR_INVALID_PARAMETER == hResult)
{
    m_State.SetWindowText(L"Error Invalid parameter");
}
else
{
    m_State.SetWindowText(L"Disconnect Error");
}
m_Open.EnableWindow(FALSE);
m_DeviceFind.EnableWindow(TRUE);
m_ServiceFind.EnableWindow(FALSE);
m_CreateConn.EnableWindow(FALSE);
m_ConnFind.EnableWindow(TRUE);
m_Connect.EnableWindow(FALSE);
m_Disconnect.EnableWindow(FALSE);
m_DeleteConn.EnableWindow(TRUE);
m_Close.EnableWindow(TRUE);
m_AllDelDevice.EnableWindow(FALSE);
m_LocalInfo.EnableWindow(TRUE);
}
}

```

#### Close

```

//Bluetooth Close
void CBluetoothTestDlg::OnBnClickedBtnClose()
{
    if(g_bBluetoothOpen == TRUE)
    {
        m_List.DeleteAllItems();
        m_ConnectItem.RemoveAll();
        BLUETOOTH_Close();
        m_State.SetWindowText(L"Close Success");
        m_Open.EnableWindow(TRUE);
        m_DeviceFind.EnableWindow(FALSE);
        m_ServiceFind.EnableWindow(FALSE);
        m_CreateConn.EnableWindow(FALSE);
        m_ConnFind.EnableWindow(FALSE);
        m_Connect.EnableWindow(FALSE);
        m_Disconnect.EnableWindow(FALSE);
        m_DeleteConn.EnableWindow(FALSE);
        m_Close.EnableWindow(FALSE);
        m_AllDelDevice.EnableWindow(FALSE);
        m_LocalInfo.EnableWindow(FALSE);
    }
}

```

```
}
```

```
g_bBluetoothOpen = FALSE;
```

```
}
```

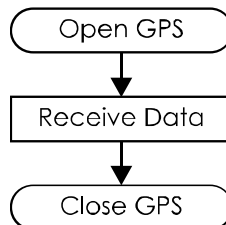


## 3.9 GPS

Supported PDA:

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

Basic GPS flow chart:



### Tip !!

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

### Open & Close

```
// GPS Open & Close
void CGpsTestDlg::OnBnClickedButtonConnect()
{
    //Connect
    if(m_bOpenGps == FALSE)
    {
        TCHAR tzCom[16] = {0x00};
        m_cmbComPort.GetLBText(m_cmbComPort.GetCurSel(), tzCom);
        if(GPS_Open(m_hWnd, tzCom, fnParseGps))
        if(GPS_GetModuleType() == MODULE_SIRF)
        {
            ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_CONNECT), FALSE);
            ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_CONNECT), TRUE);
            ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_RUN), FALSE);
            ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_SEARCH), FALSE);
            m_cmbAlpData.EnableWindow(FALSE);
        }
    }
}
```

```

    }
    else if(GPS_GetModuleType() == MODULE_UBLOX)
    {
        ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_CONNECT), FALSE);
        ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_CONNECT), TRUE);
        ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_RUN), TRUE);
        ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_SEARCH), TRUE);
        m_cmbAlpData.EnableWindow(TRUE);
    }
    m_cmbStartType.EnableWindow(TRUE);
    m_chkFileWrite.EnableWindow(TRUE);
    m_chkStatic.EnableWindow(TRUE);
    m_nFixTick = GetTickCount();
    m_bOpenGps = TRUE;
    ::SetWindowText(::GetDlgItem(m_hWnd, IDC_BUTTON_CONNECT), L"DISCONNECT");
}
else
{
    AfxMessageBox(L"Open Fail");
}
}
else
{
    if(GPS_Close())
    {
        ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_CONNECT), FALSE);
        ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_CONNECT), TRUE);
        ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_RUN), FALSE);
        ::EnableWindow(::GetDlgItem(m_hWnd, IDC_BUTTON_SEARCH), FALSE);
        m_cmbStartType.EnableWindow(FALSE);
        m_cmbAlpData.EnableWindow(FALSE);
        m_chkFileWrite.EnableWindow(FALSE);
        m_chkStatic.EnableWindow(FALSE);
        m_bOpenGps = FALSE;
        ::SetWindowText(::GetDlgItem(m_hWnd, IDC_BUTTON_CONNECT), L"CONNECT");
    }
    else
    {
        AfxMessageBox(L"Close Fail");
    }
}
}
}

```

#### Receive Message from GPS Module

```

// Receive Message from GPS Module - (1) Parse GPS Data
long CGpsTestDlg::OnRecvGPSData(WPARAM wParam, LPARAM lParam)

```

```

{
    GPSParseInfo *pInf = (GPSParseInfo*)wParam;
    // GPS Information
    ParseGPSMsg(pInf);
    return 0;
}

void CGpsTestDlg::ParseGPSMsg(GPSParseInfo *pInf)
{
    CString mTemp;
    long nUTChour, nUTCmin, nUTCsec;
    int deg = 0, min = 0, sec = 0;
    // Status
    if(pInf->nPosFix > 0)
    {
        if(m_nFixTick != 0)
        {
            m_nFixTick = GetTickCount() - m_nFixTick;
            m_strGpsStatus.Format(L"GPS On - %d Sec.", m_nFixTick/1000);
            m_nFixTick = 0;
        }
    }
    else
    {
        m_strGpsStatus.Format(L"Waiting...");
        if(m_nFixTick == 0)
        {
            m_nFixTick = GetTickCount();
        }
    }
    // Time
    nUTChour = (long)(pInf->dUTCTime / 10000);
    nUTCmin = (long)(pInf->dUTCTime / 100) - nUTChour * 100;
    nUTCsec = (long)(pInf->dUTCTime / 1) - nUTChour * 10000 - nUTCmin * 100;
    nUTChour += 9;
    if(nUTChour > 24) nUTChour -= 24;
    m_strTime.Format(_T("%02d:%02d:%02d"), nUTChour, nUTCmin, nUTCsec);
    // Latitued
    deg = (int)(pInf->dLatitude / 100);
    min = (int)(pInf->dLatitude / 1) - deg * 100;
    sec = (int)((pInf->dLatitude - deg * 100 - min) * 60);
    if(pInf->bNorthLatitude) m_strLatitude.Format(_T("N %2d, %2d, %2d"), deg, min, sec);
    else m_strLatitude.Format(_T("S %2d, %2d, %2d"), deg, min, sec);
    // Longitude
    deg = (int)(pInf->dLongitude / 100);
    min = (int)(pInf->dLongitude / 1) - deg * 100;

```

```

sec = (int)((pInf->dLongitude - deg * 100 - min ) * 60);
if(pInf->bEastLongitude) m_strLongitude.Format(_T("E %2d, %2d, %2d"), deg, min, sec);
else m_strLongitude.Format(_T("W %2d, %2d, %2d"), deg, min, sec);
// Altitude
m_strAltitude.Format(_T("%.3f m"), pInf->dAltitude);
// Satellite
int nSumSNR = 0;
int nSatCount = 0;
int nAvrSNR = 0;
CString szSNR;
for(int i = 0; i < pInf->nSatNum; i++)
{
    szSNR.Format(L "[%02d:%02d]  %s", pInf->mSat[i].nID, pInf->mSat[i].nSNR, szSNR);
}
if(nSatCount != 0) nAvrSNR = nSumSNR / nSatCount;
m_strSatellite.Format(_T("%d / %d"), pInf->nSatInUse, pInf->nSatNum);
// m_strSNR.Format(L"%d", nAvrSNR);
m_strSNR = szSNR;
// Velocity
m_strVelocity.Format(_T("%.2f km"), pInf->dVelocity);
// Heading
m_strHeading.Format(_T("%.2f"), pInf->dHeading);
// DumpFile
if(m_chkFileWrite.GetCheck() == BST_CHECKED)
{
    CString szFilePath;
    szFilePath.Format(L"%s\\LogNMEA.txt", LOG_DIRECTORY);
    WriteToLog((LPCTSTR)szFilePath, pInf->mNMEAMsg);
}
UpdateData(FALSE);
}
//Receive Message from GPS Module - (2) GPS Module Restart
void CGpsTestDlg::OnCbnSelchangeComboReset()
{
    switch(m_cmbStartType.GetCurSel())
    {
    case GPS_HOT_START:
        GPS_ModuleRestart(GPS_HOT_START);
        break;
    case GPS_WARM_START:
        GPS_ModuleRestart(GPS_WARM_START);
        break;
    case GPS_COLD_START:
        GPS_ModuleRestart(GPS_COLD_START);
        break;
    }
}

```

```

default:
    GPS_ModuleRestart(GPS_HOT_START);
    break;
}
}

//Receive Message from GPS Module - (3) GPS Static Mode
BOOL CGpsTestDlg::StaticMode()
{
    if(m_chkStatic.GetCheck() == BST_CHECKED)
    {
        GPS_EnableStaticMode(TRUE);
    }
    else if(m_chkStatic.GetCheck() == BST_UNCHECKED)
    {
        GPS_EnableStaticMode(FALSE);
    }
    return FALSE;
}

```

## AGPS

```

// AGPS - (1) Almanac Data Search
void CGpsTestDlg::OnBnClickedButtonSearch()
{
    TCHAR *szFilter = L"Almanac Plus (*.alp) | *.alp | |";
    CFileDialog dlg(TRUE, NULL, NULL, OFN_HIDEREADONLY, szFilter);
    if(IDOK == dlg.DoModal())
    {
        m_strPathName = dlg.GetPathName();
        m_cmbAlpData.InsertString(7, dlg.GetFileName());
        m_cmbAlpData.SetCurSel(7);
    }
}

// AGPS - (2) AGPS Data Down & Apply
void CGpsTestDlg::OnBnClickedButtonRun()
{
    if(m_cmbAlpData.GetCurSel() == 7)
    {
        TCHAR* tzInputFile = NULL;
        tzInputFile = (TCHAR *) (LPCTSTR)m_strPathName;
        GPS_AGPSRun(tzInputFile);
    }
    else
    {
        AGPS_DAY day = (AGPS_DAY)m_cmbAlpData.GetCurSel();
        TCHAR tzInputFile[MAX_PATH] = {0x00};
        memset(tzInputFile, 0x00, sizeof(TCHAR) * MAX_PATH);
    }
}

```

```

TCHAR szThisProgram[MAX_PATH] = {0, };
CString szThisFolder;
GetModuleFileName(NULL, szThisProgram, sizeof(TCHAR)*MAX_PATH);
szThisFolder.Format(_T("%s"), szThisProgram);
int nIndex = szThisFolder.ReverseFind('\\');
szThisFolder.Format(_T("%s\\"), szThisFolder.Left(nIndex));
switch(day)
{
case AGPS_1DAY:
    wsprintf(tzInputFile, L"%s\\AGPS_1Day.alp", szThisFolder);
    break;
case AGPS_2DAY:
    wsprintf(tzInputFile, L"%s\\AGPS_2Day.alp", szThisFolder);
    break;
case AGPS_3DAY:
    wsprintf(tzInputFile, L"%s\\AGPS_3Day.alp", szThisFolder);
    break;
case AGPS_5DAY:
    wsprintf(tzInputFile, L"%s\\AGPS_5Day.alp", szThisFolder);
    break;
case AGPS_7DAY:
    wsprintf(tzInputFile, L"%s\\AGPS_7Day.alp", szThisFolder);
    break;
case AGPS_10DAY:
    wsprintf(tzInputFile, L"%s\\AGPS_10Day.alp", szThisFolder);
    break;
case AGPS_14DAY:
    wsprintf(tzInputFile, L"%s\\AGPS_14Day.alp", szThisFolder);
    break;
default:
    wsprintf(tzInputFile, L"%s\\AGPS_AnyDay.alp", szThisFolder);
    break;
}
GPS_AGPSTDown(day,tzInputFile);
GPS_AGPSTRun(tzInputFile);
}
}

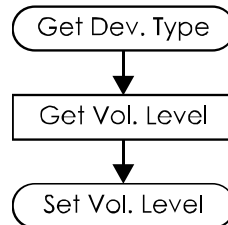
```

## 3.10 SYSTEM SDK

Supported PDA:

Brand	Restriction
M3 OX10	No restriction

Basic system SDK flow chart:



### Get Device Type

```
DeviceInfo = (DEVICE_INFO)SYSTEM_GetDeviceInfo();

if((DeviceInfo == DEVICE_M3SMARTCE) || (DeviceInfo == DEVICE_M3T) || (DeviceInfo == DEVICE_Pos))
{
    // Only for WinCE
    MoveWindow(0, 0, 240, 320);
}
```

### Get Volume Level

```
if(m_DeviceInfo != DEVICE_M3T || m_DeviceInfo != DEVICE_Pos)
{
    m_ctlMainVolumeLevel.SetRange(0, 5);
    m_ctlMainVolumeLevel.SetTicFreq(1);
    dwVolumeLevel_Cur = SYSTEM_GetVolumeLevel();

    switch (dwVolumeLevel_Cur)
    {
    case 0:
        dwVolumeLevel_Cur = 5;
        break;
    case 1:
        dwVolumeLevel_Cur = 4;
        break;
    case 2:
        dwVolumeLevel_Cur = 3;
        break;
    case 3:
        dwVolumeLevel_Cur = 2;
        break;
```

```
case 4:
    dwVolumeLevel_Cur = 1;
    break;

case 5:
    dwVolumeLevel_Cur = 0;
    break;

default:
    dwVolumeLevel_Cur = 3;
    break;

}

m_ctlMainVolumeLevel.SetPos(dwVolumeLevel_Cur);
}
```

#### Set Volume Level

```
if(pScrollBar == (CScrollBar*)&m_ctlMainVolumeLevel)
{
    dwPos = m_ctlMainVolumeLevel.GetPos();

    SYSTEM_SetVolumeLevel(dwPos);
}
```



## 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
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 Reader	RFID_UHF.h RFID_UHF.lib	M3 OX10
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
GPS	GPS.h GPS.lib	All brands with GPS *Please see <a href="#">GPS 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

Define
#define WM_SCAN_DATA     WM_APP + 350
Enum
<pre>typedef enum {     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 } SCAN_DEVICE_TYPE;  typedef enum {     OPTICON = 0,     SYMBOL,     INTERMEC,     SYMBOL_HW,     CINO,     UNKNOWN,     NOTYET } SCAN_ENGINE_TYPE;  typedef enum {     SOUND_DEFAULT = 0,     SOUND_BEEP,     SOUND_NONE } SCAN_SOUND;  typedef enum {     CODE39_STARNDARD = 0,     CODE39_CODE32,     CODE39_PZN,</pre>

```
CODE39_TRIOPTIC,  
CODE39_FULLASCII  
} SCAN_CODE39_FORMAT;
```

## Structure

```
typedef struct _DECODER{  
    BYTE bUPCA;  
    BYTE bUPCE;  
    BYTE bEAN13;  
    BYTE bEAN8;  
    BYTE bCODE39;  
    BYTE bCODE128;  
    BYTE bCODE93;  
    BYTE bCODE11;  
    BYTE bCODE25;  
    BYTE bCODABAR;  
    BYTE bKOREAPOST;  
    BYTE bMSI;  
    BYTE bGS1;  
    BYTE bTELEPEN;  
}DECODER, *PDECODER;  
  
typedef struct _DECODER_PARAMS{  
    BYTE bWindeScan;  
    BYTE bHighFilter;  
    BYTE bContinueMode;  
    BYTE bXmitAimID;  
    BYTE bVibrate;  
    int  nSound;  
    int  nTimeOut;  
    int  nSecurityLevel;  
}DECODER_PARAMS, *PDECODER_PARAMS;  
  
typedef struct _UPCA_PARAMS{  
    BYTE  bEnable;  
    BYTE  bXNum;  
    BYTE  bXCD;  
    BYTE  bUPCA_AS_EAN13;  
    BYTE  bAddOn;  
}UPCA_PARAMS, *PUPCA_PARAMS;  
  
typedef struct _UPCE_PARAMS{  
    BYTE  bEnable;  
    BYTE  bXNum;  
    BYTE  bXCD;  
    int  nConvert;  
}UPCE_PARAMS, *PUPCE_PARAMS;  
  
typedef struct _EAN13_PARAMS{  
    BYTE  bEnable;
```

```

    BYTE    bBOOKLAND;

    BYTE    bXCD;

    BYTE    bAddOn;
}EAN13_PARAMS, *PEAN13_PARAMS;

typedef struct _EAN8_PARAMS{
    BYTE    bEnable;

    BYTE    bXCD;

    BYTE    bEAN8_AS_EAN13;
}EAN8_PARAMS, *PEAN8_PARAMS;

typedef struct _CODE39_PARAMS{
    BYTE    bEnable;

    int     nFormat;

    BYTE    bCDV;

    BYTE    bXCD;

    int     nMinLen;

    int     nMaxLen;
}CODE39_PARAMS, *PCODE39_PARAMS;

typedef struct _CODE128_PARAMS{
    BYTE    bEnable;

    BYTE    bUCCEAN128;

    TCHAR   szFNC1_ASCII[256];

    int     nMinLen;

    int     nMaxLen;
}CODE128_PARAMS, *PCODE128_PARAMS;

typedef struct _CODE93_PARAMS{
    BYTE    bEnable;

    int     nMinLen;

    int     nMaxLen;
}CODE93_PARAMS, *PCODE93_PARAMS;

typedef struct _KOREAPOST_PARAMS{
    BYTE    bEnable;
}KOREAPOST_PARAMS, *PKOREAPOST_PARAMS;

typedef struct _CODE11_PARAMS{
    BYTE    bEnable;

    BYTE    bXCD;

    int     nCDV;

    int     nMinLen;

    int     nMaxLen;
}CODE11_PARAMS, *PCODE11_PARAMS;

typedef struct _CODE25_PARAMS{
    BYTE    bI2OF5;

    BYTE    bS2OF5;

    BYTE    bITF14;

    BYTE    bMATRIX2OF5;

    BYTE    bCHINAPOST;

    BYTE    bINDUSTRY;

```

```
    BYTE    bIATA;
    BYTE    bCDV;
    BYTE    bXCD;
    int     nMinLen;
    int     nMaxLen;
}CODE25_PARAMS, *PCODE25_PARAMS;

typedef struct _CODABAR_PARAMS{
    BYTE    bEnable;
    BYTE    bXSS;
    int     nMinLen;
    int     nMaxLen;
}CODABAR_PARAMS, *PCODABAR_PARAMS;

typedef struct _MSI_PARAMS{
    BYTE    bEnable;
    BYTE    bXCD;
    int     nCDV;
    int     nMinLen;
    int     nMaxLen;
}MSI_PARAMS, *PMSI_PARAMS;

typedef struct _GS1_PARAMS{
    BYTE    bGS1;
    BYTE    bGS1LIM;
    BYTE    bGS1EXP;
}GS1_PARAMS, *PGS1_PARAMS;

typedef struct _TELEPEN_PARAMS{
    BYTE    bEnable;
    BYTE    bNumeric;
}TELEPEN_PARAMS, *PTELEPEN_PARAMS;
```

## Functions for 1D Scanner

Name	Description
<a href="#">SCAN_Close</a>	Closes an open scanner
<a href="#">SCAN_GetCODABAR</a>	Gets the option of CODABAR Barcode
<a href="#">SCAN_GetCODE11</a>	Gets the option of CODE11 Barcode
<a href="#">SCAN_GetCODE128</a>	Gets the option of CODE128 Barcode
<a href="#">SCAN_GetCODE25</a>	Gets the option of CODE25 Barcode
<a href="#">SCAN_GetCODE39</a>	Gets the option of CODE39 Barcode
<a href="#">SCAN_GetCODE93</a>	Gets the option of CODE93 Barcode
<a href="#">SCAN_GetDeviceType</a>	Gets the type of device
<a href="#">SCAN_GetEAN13</a>	Gets the option of EAN-13 Barcode
<a href="#">SCAN_GetEAN8</a>	Gets the option of EAN-8 Barcode
<a href="#">SCAN_GetEngineType</a>	Gets the type of scanner engine
<a href="#">SCAN_GetGS1</a>	Gets the option of GS1 Barcode
<a href="#">SCAN_GetKOREAPOST</a>	Gets the option of KOREAPOST Barcode
<a href="#">SCAN_GetMSI</a>	Gets the option of MSI Barcode
<a href="#">SCAN_GetOption</a>	Gets the option of Scanner
<a href="#">SCAN_GetScanData</a>	Gets the ScanData which is read by scanner
<a href="#">SCAN_GetSymbology</a>	Gets Enable/Disable of Symbologies
<a href="#">SCAN_GetTELEPEN</a>	Gets the option of TELEPEN Barcode
<a href="#">SCAN_GetUPCA</a>	Gets the option of UPC-A Barcode
<a href="#">SCAN_GetUPCE</a>	Gets the option of UPC-E Barcode
<a href="#">SCAN_GetVersionInfo</a>	Gets the information of scanner engine and dll version
<a href="#">SCAN_Open</a>	Opens a scanner
<a href="#">SCAN_Read</a>	Starts the beaming of scanner
<a href="#">SCAN_ReadCancel</a>	Stops the beaming of scanner
<a href="#">SCAN_SetCODABAR</a>	Sets the option of CODABAR Barcode
<a href="#">SCAN_SetCODE11</a>	Sets the option of CODE11 Barcode
<a href="#">SCAN_SetCODE128</a>	Sets the option of CODE128 Barcode
<a href="#">SCAN_SetCODE25</a>	Sets the option of CODE25 Barcode
<a href="#">SCAN_SetCODE39</a>	Sets the option of CODE39 Barcode
<a href="#">SCAN_SetCODE93</a>	Sets the option of CODE93 Barcode

<a href="#"><u>SCAN_SetEAN13</u></a>	Sets the option of EAN-13 Barcode
<a href="#"><u>SCAN_SetEAN8</u></a>	Sets the option of EAN-8 Barcode
<a href="#"><u>SCAN_SetGS1</u></a>	Sets the option of GS1 Barcode
<a href="#"><u>SCAN_SetKOREAPOST</u></a>	Sets the option of KOREAPOST Barcode
<a href="#"><u>SCAN_SetMSI</u></a>	Sets the option of MSI Barcode
<a href="#"><u>SCAN_SetOption</u></a>	Sets the option of Scanner
<a href="#"><u>SCAN_SetSymbology</u></a>	Sets the Enable/Disable of Symbologies
<a href="#"><u>SCAN_SetSymbologyAll</u></a>	Enables all of Symbologies
<a href="#"><u>SCAN_SetSymbologyDefault</u></a>	Initializes all option of scanner
<a href="#"><u>SCAN_SetTELEPEN</u></a>	Sets the option of TELEPEN Barcode
<a href="#"><u>SCAN_SetUPCA</u></a>	Sets the option of UPC-A Barcode
<a href="#"><u>SCAN_SetUPCE</u></a>	Sets the option of UPC-E Barcode

### 4.1.1 SCAN\_Close

#### Description

Closes an open scanner.

#### Syntax

```
SCANNER_API BOOL SCAN_Close();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See also

SCAN\_Open

#### For .Net

Namespace : `ScannerNet.Scanner`

Function : `bool Close()`

#### Example

```
if(SCAN_Close() == FALSE)
{
    ::MessageBox(NULL, L"Error : SCAN_Close()", NULL, MB_TOPMOST);
}
```

### 4.1.2 SCAN\_GetCODABAR

#### Description

Gets the option of CODABAR Barcode.

#### Syntax

```
SCANNER_API BOOL SCAN_GetCODABAR(PCODABAR_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

SCAN\_SetCODABAR

## For .Net

Namespace : ScannerNet.Scanner

Function : bool GetCODABAR(out CODABAR\_PARAMS pCodabar)

## Example

```
PCODABAR_PARAMS  pCodabar = new CODABAR_PARAMS();
if(SCAN_GetCODABAR(pCodabar) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_GetCODABAR()", NULL, MB_TOPMOST);
m_bEnable = pCodabar->bEnable;
m_bXSS = pCodabar->bXSS;
m_nMinLen = pCodabar->nMinLen;
m_nMaxLen = pCodabar->nMaxLen;
delete pCodabar;
```

## 4.1.3 SCAN\_GetCODE11

### Description

Gets the option of CODE11 Barcode.

### Syntax

```
SCANNER_API BOOL SCAN_GetCODE11(PCODE11_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

SCAN\_SetCODE11

## For .Net

Namespace : ScannerNet.Scanner

Function : bool GetCODE11(out CODE11\_PARAMS pCode11)

## Example

```
PCODE11_PARAMS  pCode11 = new CODE11_PARAMS();
```

```

if(SCAN_GetCODE11(pCode11) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_GetCODE11()", NULL, MB_TOPMOST);
m_bEnable = pCode11->bEnable;
m_bXCD = pCode11->bXCD;
m_nCDV = pCode11->nCDV;
m_nMinLen = pCode11->nMinLen;
m_nMaxLen = pCode11->nMaxLen;
delete pCode11;

```

#### 4.1.4 SCAN\_GetCODE128

##### Description

Gets the option of CODE128 Barcode.

##### Syntax

```
SCANNER_API BOOL SCAN_GetCODE128(PCODE128_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

SCAN\_SetCODE128

##### For .Net

Namespace : ScannerNet.Scanner

Function : bool GetCODE128(out CODE128\_PARAMS pCode128)

##### Example

```

PCODE128_PARAMS pCode128 = new CODE128_PARAMS();
if(SCAN_GetCODE128(pCode128) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_GetCODE128()", NULL, MB_TOPMOST);
m_bEnable = pCode128->bEnable;
m_bUccean128 = pCode128->bUCCEAN128;
m_nMinLen = pCode128->nMinLen;
m_nMaxLen = pCode128->nMaxLen;

```

```

if((pScanTestDlg->m_EngineType == OPTICON) || (pScanTestDlg->m_EngineType == SYMBOL) ||
(pScanTestDlg->m_EngineType == INTERMEC))

    m_strFNCASCII.Format(L"%s", pCode128->szFNC1_ASCII);

delete pCode128;

```

### 4.1.5 SCAN\_GetCODE25

#### Description

Gets the option of CODE25 Barcode.

#### Syntax

```
SCANNER_API BOOL SCAN_GetCODE25(PCODE25_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

SCAN\_SetCODE25

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool GetCODE25(out CODE25\_PARAMS pCode25)

#### Example

```

PCODE25_PARAMS pCode25 = new CODE25_PARAMS();
if(SCAN_GetCODE25(pCode25) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_GetCODE25()", NULL, MB_TOPMOST);

m_bl2of5 = pCode25->bl2OF5;
m_bS2of5 = pCode25->bS2OF5;
m_bITF14 = pCode25->bITF14;
m_bMatrix2of5 = pCode25->bMATRIX2OF5;
m_bChinaPost = pCode25->bCHINAPOST;
m_bIndustry = pCode25->bINDUSTRY;
m_blata = pCode25->blATA;
m_bCDV = pCode25->bCDV;
m_bXCD = pCode25->bXCD;

```

```
m_nMinLen = pCode25->nMinLen;  
m_nMaxLen = pCode25->nMaxLen;  
delete pCode25;
```

#### 4.1.6 SCAN\_GetCODE39

##### Description

Gets the option of CODE39 Barcode.

##### Syntax

```
SCANNER_API BOOL SCAN_GetCODE39(PCODE39_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

SCAN\_SetCODE39

##### For .Net

Namespace : ScannerNet.Scanner

Function : bool GetCODE39(out CODE39\_PARAMS pCode39)

##### Example

```
PCODE39_PARAMS pCode39 = new CODE39_PARAMS();  
if(SCAN_GetCODE39(pCode39) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_GetCODE39()", NULL, MB_TOPMOST);  
m_bEnable = pCode39->bEnable;  
m_nFormat = pCode39->nFormat;  
m_bCDV = pCode39->bCDV;  
m_bXCD = pCode39->bXCD;  
m_nMinLen = pCode39->nMinLen;  
m_nMaxLen = pCode39->nMaxLen;  
delete pCode39;
```

### 4.1.7 SCAN\_GetCODE93

#### Description

Gets the option of CODE93 Barcode.

#### Syntax

```
SCANNER_API BOOL SCAN_GetCODE93(PCODE93_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

SCAN\_SetCODE39

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool GetCODE93(out CODE93\_PARAMS pCode93)

#### Example

```
PCODE93_PARAMS pCode93 = new CODE93_PARAMS();
if(SCAN_GetCODE93(pCode93) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_GetCODE93()", NULL, MB_TOPMOST);
m_bEnable = pCode93->bEnable;
m_nMinLen = pCode93->nMinLen;
m_nMaxLen = pCode93->nMaxLen;
delete pCode93;
```

### 4.1.8 SCAN\_GetDeviceType

#### Description

Gets the type of device.

#### Syntax

```
SCANNER_API SCAN_DEVICE_TYPE SCAN_GetDeviceType();
```

#### Parameters

None

#### Return Value

The return value is SCAN\_DEVICE\_TYPE.

**Remarks**

None

**See Also**

SCAN\_GetEngineType

**For .Net**

Namespace : ScannerNet.Scanner

Function : SCAN\_DEVICE\_TYPE GetDeviceType();

**Example**

None

### 4.1.9 SCAN\_GetEAN13

**Description**

Gets the option of EAN-13 Barcode.

**Syntax**

```
SCANNER_API BOOL SCAN_GetEAN13(PEAN13_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**

SCAN\_SetEAN13

**For .Net**

Namespace : ScannerNet.Scanner

Function : bool GetEAN13(out EAN13\_PARAMS pEan13)

**Example**

```
PEAN13_PARAMS pEan13 = new EAN13_PARAMS();  
if(SCAN_GetEAN13(pEan13) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_GetEAN13()", NULL, MB_TOPMOST);  
m_bEnable = pEan13->bEnable;  
m_bBookland = pEan13->bBOOKLAND;
```

```
m_bXCD = pEan13->bXCD;  
m_bAddOn = pEan13->bAddOn;  
delete pEan13;
```

#### 4.1.10 SCAN\_GetEAN8

##### Description

Gets the option of EAN-8 Barcode.

##### Syntax

```
SCANNER_API BOOL SCAN_GetEAN8(PEAN8_PARAMS pEan8);
```

##### Parameters

*pEan8*

Pointer to an 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

SCAN\_SetEAN8

##### For .Net

Namespace : ScannerNet.Scanner

Function : bool GetEAN8(out EAN8\_PARAMS pEan8)

##### Example

```
PEAN8_PARAMS pEan8 = new EAN8_PARAMS();  
if(SCAN_GetEAN8(pEan8) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_GetEAN8()", NULL, MB_TOPMOST);  
m_bEnable = pEan8->bEnable;  
m_bXCD = pEan8->bXCD;  
m_bEan8AsEan13 = pEan8->bEAN8_AS_EAN13;  
delete pEan8;
```

#### 4.1.11 SCAN\_GetEngineType

##### Description

Gets the type of scanner engine.

##### Syntax

```
SCANNER_API SCAN_ENGINE_TYPE SCAN_GetEngineType();
```

**Parameters**

None

**Return Value**

The return value is SCAN\_ENGINE\_TYPE.

**Remarks**

None

**See Also**

SCAN\_GetDeviceType

**For .Net**

Namespace : ScannerNet.Scanner

Function : SCAN\_ENGINE\_TYPE GetEngineType();

**Example**

None

### 4.1.12 SCAN\_GetGS1

**Description**

Gets the option of GS1 Barcode.

**Syntax**

```
SCANNER_API BOOL SCAN_GetGS1PGS1_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**

SCAN\_SetGS1

**For .Net**

Namespace : ScannerNet.Scanner

Function : bool GetGS1(out GS1\_PARAMS pGs1)

**Example**

```
PGS1_PARAMS pGs1 = new GS1_PARAMS();
```



```

if(SCAN_GetGS1(pGs1) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_GetGS1()", NULL, MB_TOPMOST);
m_bEnable = pGs1->bGS1;
m_bGs1Lim = pGs1->bGS1LIM;
m_bGs1Exp = pGs1->bGS1EXP;
delete pGs1;

```

### 4.1.13 SCAN\_GetKOREAPOST

#### Description

Gets the option of KOREAPOST Barcode.

#### Syntax

```
SCANNER_API BOOL SCAN_GetKOREAPOST(PKOREAPOST_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

SCAN\_SetKOREAPOST

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool GetKOREAPOST(out KOREAPOST\_PARAMS pKoreaPost)

#### Example

```

PKOREAPOST_PARAMS pKoreaPost = new KOREAPOST_PARAMS();
if(SCAN_GetKOREAPOST(pKoreaPost) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_GetKOREAPOST()", NULL, MB_TOPMOST);
m_bEnable = pKoreaPost->bEnable;
delete pKoreaPost;

```

### 4.1.14 SCAN\_GetMSI

#### Description

Gets the option of MSI Barcode.

**Syntax**

```
SCANNER_API BOOL SCAN_GetMSI(PMSI_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**

SCAN\_SetMSI

**For .Net**

Namespace : ScannerNet.Scanner

Function : bool GetMSI(out MSI\_PARAMS pMsi)

**Example**

```
PMSI_PARAMS pMsi = new MSI_PARAMS();  
if(SCAN_GetMSI(pMsi) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_GetMSI()", NULL, MB_TOPMOST);  
m_bEnable = pMsi->bEnable;  
m_bXCD = pMsi->bXCD;  
m_nCDV = pMsi->nCDV;  
m_nMinLen = pMsi->nMinLen;  
m_nMaxLen = pMsi->nMaxLen;  
delete pMsi;
```

### 4.1.15 SCAN\_GetOption

**Description**

Gets the option of Scanner.

**Syntax**

```
SCANNER_API BOOL SCAN_GetOption(PDECODER_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

SCAN\_SetOption

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool GetOption(out DECODER\_PARAMS pOption)

#### Example

```
PDECODER_PARAMS pOption = new DECODER_PARAMS();
SCAN_GetOption(pOption);
m_nSyncMode = pScanTestDlg->m_bSyncMode;
m_nSound = pOption->nSound;
m_bVibrate = pOption->bVibrate;
m_bAimID = pOption->bXmitAimID;
m_bContinueMode = pOption->bContinueMode;
m_bWideScan = pOption->bWindeScan;
m_bHighFilter = pOption->bHighFilter;
m_ctrlComboTimeOut.SetCurSel(pOption->nTimeOut - 1);
m_ctrlComboSecurityLevel.SetCurSel(pOption->nSecurityLevel - 1);
delete pOption;
```

### 4.1.16 SCAN\_GetScanData

#### Description

Gets the ScanData which is read by scanner.

#### Syntax

```
SCANNER_API BOOL SCAN_GetScanData(TCHAR *pszBarType, TCHAR *pszBarData);
```

#### Parameters

*pszBarType*

Pointer to barcode type.

*pszBarData*

Pointer to barcode data

#### Return Value

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : ScannerNet.Scanner

Function : bool GetScanData(StringBuilder szBarType, StringBuilder szBarData)

**Example**

```
TCHAR  szBarType[1024] = {0, };
```

```
TCHAR  szBarData[1024] = {0, };
```

```
SCAN_GetScanData(szBarType, szBarData);
```

### 4.1.17 SCAN\_GetSymbology

**Description**

Gets Enable/Disable of Symbologies.

**Syntax**

```
SCANNER_API BOOL SCAN_GetSymbology(PDECODER 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**

SCAN\_SetSymbology

**For .Net**

Namespace : ScannerNet.Scanner

Function : bool GetSymbology(out DECODER pSymbology)

**Example**

```
PDECODER pDecoder = new DECODER();
```

```
SCAN_GetSymbology(pDecoder);
```

```
m_bUpca = pDecoder->bUPCA;
```

```
m_bUpce = pDecoder->bUPCE;
```

```

m_bEan13 = pDecoder->bEAN13;
m_bEan8 = pDecoder->bEAN8;
m_bCode39 = pDecoder->bCODE39;
m_bCode128 = pDecoder->bCODE128;
m_bCode93 = pDecoder->bCODE93;
m_bCode11 = pDecoder->bCODE11;
m_bCode25 = pDecoder->bCODE25;
m_bCodabar = pDecoder->bCODABAR;
m_bKoreaPost = pDecoder->bKOREAPOST;
m_bMsi = pDecoder->bMSI;
m_bGs1 = pDecoder->bGS1;
m_bTelepen = pDecoder->bTELEPEN;
delete pDecoder;

```

#### 4.1.18 SCAN\_GetTELEPEN

##### Description

Gets the option of TELEPEN Barcode.

##### Syntax

```
SCANNER_API BOOL SCAN_GetTELEPEN(PTELEPEN_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

SCAN\_SetTELEPEN

##### For .Net

Namespace : ScannerNet.Scanner

Function : bool GetTELEPEN(out TELEPEN\_PARAMS pTelepen)

##### Example

```

PTELEPEN_PARAMS pTelepen = new TELEPEN_PARAMS();
if(SCAN_GetTELEPEN(pTelepen) == FALSE)

```

```
        ::MessageBox(NULL, L"Error : SCAN_GetTELEPEN()", NULL, MB_TOPMOST);  
m_bEnable = pTelepen->bEnable;  
m_bNumeric = pTelepen->bNumeric;  
delete pTelepen;
```

#### 4.1.19 SCAN\_GetUPCA

##### Description

Gets the option of UPC-A Barcode.

##### Syntax

```
SCANNER_API BOOL SCAN_GetUPCA(PUPCA_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

SCAN\_SetUPCA

##### For .Net

Namespace : ScannerNet.Scanner

Function : bool GetUPCA(out UPCA\_PARAMS pUpca)

##### Example

```
PUPCA_PARAMS pUpca = new UPCA_PARAMS();  
if(SCAN_GetUPCA(pUpca) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_GetUPCA()", NULL, MB_TOPMOST);  
m_bEnable = pUpca->bEnable;  
m_bXNum = pUpca->bXNum;  
m_bXCD = pUpca->bXCD;  
m_bUpcaAsEan13 = pUpca->bUPCA_AS_EAN13;  
m_bAddOn = pUpca->bAddOn;  
delete pUpca;
```

## 4.1.20 SCAN\_GetUPCE

### Description

Gets the option of UPC-E Barcode.

### Syntax

```
SCANNER_API BOOL SCAN_GetUPCE(PUPCE_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

SCAN\_SetUPCE

### For .Net

Namespace : ScannerNet.Scanner

Function : bool GetUPCE(out UPCE\_PARAMS pUpce)

### Example

```
PUPCE_PARAMS pUpce = new UPCE_PARAMS();  
if(SCAN_GetUPCE(pUpce) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_GetUPCE()", NULL, MB_TOPMOST);  
m_bEnable = pUpce->bEnable;  
m_bXNum = pUpce->bXNum;  
m_bXCD = pUpce->bXCD;  
m_nConvert = pUpce->nConvert;  
delete pUpce;
```

## 4.1.21 SCAN\_GetVersionInfo

### Description

Gets the information of scanner engine and dll version.

### Syntax

```
SCANNER_API BOOL SCAN_GetVersionInfo(TCHAR* pszVersion);
```

### 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 .Net**

Namespace : ScannerNet.Scanner

Function : string GetVersionInfo()

**Example**

```
TCHAR szVersionInfo[1024] = {0, };  
SCAN_GetVersionInfo(szVersionInfo);
```

## 4.1.22 SCAN\_Open

**Description**

Opens a scanner.

**Syntax**

```
SCANNER_API BOOL SCAN_Open();
```

**Parameters**

None

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

SCAN\_Close

**For .Net**

Namespace : ScannerNet.Scanner

Function : bool Open()

**Example**

```
if(SCAN_Open() == FALSE)  
{  
    ::MessageBox(NULL, L"Error : SCAN_Open()", NULL, MB_TOPMOST);  
}
```



}

### 4.1.23 SCAN\_Read

#### Description

Starts the beaming of scanner.

#### Syntax

```
SCANNER_API BOOL SCAN_Read();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

SCAN\_ReadCancel

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool Read()

#### Example

None

### 4.1.24 SCAN\_ReadCancel

#### Description

Stops the beaming of scanner.

#### Syntax

```
SCANNER_API BOOL SCAN_ReadCancel();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

SCAN\_Read

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool ReadCancel()

#### Example

None

### 4.1.25 SCAN\_SetCODABAR

#### Description

Sets the option of CODABAR Barcode.

#### Syntax

SCANNER\_API BOOL SCAN\_SetCODABAR(PCODABAR\_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

SCAN\_GetCODABAR

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetCODABAR(ref CODABAR\_PARAMS pCodabar)

#### Example

```
PCODABAR_PARAMS  pCodabar = new CODABAR_PARAMS();
pCodabar->bEnable = m_bEnable;
pCodabar->bXSS = m_bXSS;
pCodabar->nMinLen = m_nMinLen;
pCodabar->nMaxLen = m_nMaxLen;
if(SCAN_SetCODABAR(pCodabar) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetCODABAR()", NULL, MB_TOPMOST);
delete pCodabar;
```

## 4.1.26 SCAN\_SetCODE11

### Description

Sets the option of CODE11 Barcode.

### Syntax

```
SCANNER_API BOOL SCAN_SetCODE11(PCODE11_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

SCAN\_GetCODE11

### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetCODE11(ref CODE11\_PARAMS pCode11)

### Example

```
PCODE11_PARAMS pCode11 = new CODE11_PARAMS();
pCode11->bEnable = m_bEnable;
pCode11->bXCD = m_bXCD;
pCode11->nCDV = m_nCDV;
pCode11->nMinLen = m_nMinLen;
pCode11->nMaxLen = m_nMaxLen;
if(SCAN_SetCODE11(pCode11) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetCODE11()", NULL, MB_TOPMOST);
delete pCode11;
```

## 4.1.27 SCAN\_SetCODE128

### Description

Sets the option of CODE128 Barcode.

### Syntax

```
SCANNER_API BOOL SCAN_SetCODE128(PCODE128_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**

SCAN\_GetCODE128

#### **For .Net**

Namespace : ScannerNet.Scanner

Function : bool SetCODE128(ref CODE128\_PARAMS pCode128)

#### **Example**

```
PCODE128_PARAMS pCode128 = new CODE128_PARAMS();
pCode128->bEnable = m_bEnable;
pCode128->bUCCEAN128 = m_bUccean128;
pCode128->nMinLen = m_nMinLen;
pCode128->nMaxLen = m_nMaxLen;
wsprintf(pCode128->szFNC1_ASCII, L"%s", m_strFNCASCII);
if(SCAN_SetCODE128(pCode128) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetCODE128()", NULL, MB_TOPMOST);
delete pCode128;
```

### **4.1.28 SCAN\_SetCODE25**

#### **Description**

Sets the option of CODE25 Barcode.

#### **Syntax**

```
SCANNER_API BOOL SCAN_SetCODE25(PCODE25_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

SCAN\_GetCODE25

## For .Net

Namespace : ScannerNet.Scanner

Function : bool SetCODE25(ref CODE25\_PARAMS pCode25)

## Example

```
PCODE25_PARAMS pCode25 = new CODE25_PARAMS();
pCode25->bl2OF5 = m_bl2of5;
pCode25->bS2OF5 = m_bS2of5;
pCode25->blTF14 = m_blTF14;
pCode25->bMATRIX2OF5 = m_bMatrix2of5;
pCode25->bCHINAPOST = m_bChinaPost;
pCode25->bINDUSTRY = m_bIndustry;
pCode25->blATA = m_blata;
pCode25->bCDV = m_bCDV;
pCode25->bXCD = m_bXCD;
pCode25->nMinLen = m_nMinLen;
pCode25->nMaxLen = m_nMaxLen;
if(SCAN_SetCODE25(pCode25) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetCODE25()", NULL, MB_TOPMOST);
delete pCode25;
```

## 4.1.29 SCAN\_SetCODE39

### Description

Sets the option of CODE39 Barcode.

### Syntax

```
SCANNER_API BOOL SCAN_SetCODE39(PCODE39_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

SCAN\_GetCODE39

## For .Net

Namespace : ScannerNet.Scanner

Function : bool SetCODE39(ref CODE39\_PARAMS pCode39)

## Example

```
PCODE39_PARAMS pCode39 = new CODE39_PARAMS();
if(SCAN_GetCODE39(pCode39) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_GetCODE39()", NULL, MB_TOPMOST);
m_bEnable = pCode39->bEnable;
m_nFormat = pCode39->nFormat;
m_bCDV = pCode39->bCDV;
m_bXCD = pCode39->bXCD;
m_nMinLen = pCode39->nMinLen;
m_nMaxLen = pCode39->nMaxLen;
delete pCode39;
```

## 4.1.30 SCAN\_SetCODE93

### Description

Sets the option of CODE93 Barcode.

### Syntax

```
SCANNER_API BOOL SCAN_SetCODE93(PCODE93_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

SCAN\_GetCODE39

## For .Net

Namespace : ScannerNet.Scanner

Function : bool SetCODE93(ref CODE93\_PARAMS pCode93)

### Example

```
PCODE93_PARAMS pCode93 = new CODE93_PARAMS();  
pCode93->bEnable = m_bEnable;  
pCode93->nMinLen = m_nMinLen;  
pCode93->nMaxLen = m_nMaxLen;  
if(SCAN_SetCODE93(pCode93) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetCODE93()", NULL, MB_TOPMOST);  
delete pCode93;
```

## 4.1.31 SCAN\_SetEAN13

### Description

Sets the option of EAN-13 Barcode.

### Syntax

```
SCANNER_API BOOL SCAN_SetEAN13(PEAN13_PARAMS pEan13);
```

### Parameters

*pEan13*

Pointer to an EAN13\_PARAMS structure holding the EAN-13 common parameters.

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

SCAN\_GetEAN13

### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetEAN13(ref EAN13\_PARAMS pEan13)

### Example

```
PEAN13_PARAMS pEan13 = new EAN13_PARAMS();  
pEan13->bEnable = m_bEnable;  
pEan13->bBOOKLAND = m_bBookland;  
pEan13->bXCD = m_bXCD;  
pEan13->bAddOn = m_bAddOn;  
if(SCAN_SetEAN13(pEan13) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetEAN13()", NULL, MB_TOPMOST);
```

delete pEan13;

### 4.1.32 SCAN\_SetEAN8

#### Description

Sets the option of EAN-8 Barcode.

#### Syntax

```
SCANNER_API BOOL SCAN_SetEAN8(PEAN8_PARAMS pEan8);
```

#### Parameters

*pEan8*

Pointer to an EAN8\_PARAMS structure holding the EAN9 common parameters.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

SCAN\_GetEAN8

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetEAN8(ref EAN8\_PARAMS pEan8)

#### Example

```
PEAN8_PARAMS pEan8 = new EAN8_PARAMS();  
pEan8->bEnable = m_bEnable;  
pEan8->bXCD = m_bXCD;  
pEan8->bEAN8_AS_EAN13 = m_bEan8AsEan13;  
if(SCAN_SetEAN8(pEan8) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetEAN8()", NULL, MB_TOPMOST);  
delete pEan8;
```

### 4.1.33 SCAN\_SetGS1

#### Description

Sets the option of GS1 Barcode.

#### Syntax

```
SCANNER_API BOOL SCAN_SetGS1(PGS1_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**

SCAN\_GetGS1

#### **For .Net**

Namespace : ScannerNet.Scanner

Function : bool SetGS1(ref GS1\_PARAMS pGs1)

#### **Example**

```
PGS1_PARAMS    pGs1 = new GS1_PARAMS();
pGs1->bGS1 = m_bEnable;
pGs1->bGS1LIM = m_bGs1Lim;
pGs1->bGS1EXP = m_bGs1Exp;
if(SCAN_SetGS1(pGs1) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetGS1()", NULL, MB_TOPMOST);
delete pGs1;
```

### **4.1.34 SCAN\_SetKOREAPOST**

#### **Description**

Sets the option of KOREAPOST Barcode.

#### **Syntax**

SCANNER\_API BOOL SCAN\_SetKOREAPOST(PKOREAPOST\_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**

SCAN\_GetKOREAPOST

### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetKOREAPOST(ref KOREAPOST\_PARAMS pKoreaPost)

### Example

```
PKOREAPOST_PARAMS pKoreaPost = new KOREAPOST_PARAMS();  
pKoreaPost->bEnable = m_bEnable;  
if(SCAN_SetKOREAPOST(pKoreaPost) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetKOREAPOST()", NULL, MB_TOPMOST);  
delete pKoreaPost;
```

## 4.1.35 SCAN\_SetMSI

### Description

Sets the option of MSI Barcode.

### Syntax

SCANNER\_API BOOL SCAN\_SetMSI(PMSI\_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

SCAN\_GetMSI

### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetMSI(ref MSI\_PARAMS pMsi)

### Example

```
PMSI_PARAMS pMsi = new MSI_PARAMS();  
pMsi->bEnable = m_bEnable;  
pMsi->bXCD = m_bXCD;  
pMsi->nCDV = m_nCDV;  
pMsi->nMinLen = m_nMinLen;  
pMsi->nMaxLen = m_nMaxLen;
```

```

if(SCAN_SetMSI(pMsi) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetMSI()", NULL, MB_TOPMOST);
delete pMsi;

```

### 4.1.36 SCAN\_SetOption

#### Description

Sets the option of Scanner.

#### Syntax

```
SCANNER_API BOOL SCAN_SetOption(PDECODER_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

SCAN\_GetOption

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetOption(ref DECODER\_PARAMS pOption)

#### Example

```

PDECODER_PARAMS pOption = new DECODER_PARAMS();
pOption->nSound = m_nSound;
pOption->bVibrate = m_bVibrate;
pOption->bXmitAimID = m_bAimID;
pOption->bContinueMode = m_bContinueMode;
pOption->bWindeScan = m_bWideScan;
pOption->bHighFilter = m_bHighFilter;
pOption->nTimeOut = m_ctrlComboTimeOut.GetCurSel() + 1;
pOption->nSecurityLevel = m_ctrlComboSecurityLevel.GetCurSel() + 1;
SCAN_SetOption(pOption);
delete pOption;

```

### 4.1.37 SCAN\_SetSymbology

#### Description

Sets Enable/Disable of Symbologies.

#### Syntax

```
SCANNER_API BOOL SCAN_SetSymbology(PDECODER 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

SCAN\_GetSymbology

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetSymbology(ref DECODER pSymbology)

#### Example

```
PDECODER pDecoder = new DECODER();  
pDecoder->bUPCA = m_bUpca;  
pDecoder->bUPCE = m_bUpce;  
pDecoder->bEAN13 = m_bEan13;  
pDecoder->bEAN8 = m_bEan8;  
pDecoder->bCODE39 = m_bCode39;  
pDecoder->bCODE128 = m_bCode128;  
pDecoder->bCODE93 = m_bCode93;  
pDecoder->bCODE11 = m_bCode11;  
pDecoder->bCODE25 = m_bCode25;  
pDecoder->bCODABAR = m_bCodabar;  
pDecoder->bKOREAPOST = m_bKoreaPost;  
pDecoder->bMSI = m_bMsi;  
pDecoder->bGS1 = m_bGs1;  
pDecoder->bTELEPEN = m_bTelepen;  
SCAN_SetSymbology(pDecoder);
```

delete pDecoder;

### 4.1.38 SCAN\_SetSymbologyAll

#### Description

Enables all of Symbologies.

#### Syntax

```
SCANNER_API BOOL SCAN_SetSymbologyAll();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

SCAN\_SetSymbologyDefault

#### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetSymbologyAll();

#### Example

None

### 4.1.39 SCAN\_SetSymbologyDefault

#### Description

Initializes all option of scanner.

#### Syntax

```
SCANNER_API BOOL SCAN_SetSymbologyDefault();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

SCAN\_SetSymbologyAll

### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetSymbologyDefault()

### Example

None

## 4.1.40 SCAN\_SetTELEPEN

### Description

Sets the option of TELEPEN Barcode.

### Syntax

```
SCANNER_API BOOL SCAN_SetTELEPEN(PTELEPEN_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

SCAN\_GetTELEPEN

### For .Net

Namespace : ScannerNet.Scanner

Function : bool SetTELEPEN(ref TELEPEN\_PARAMS pTelepen)

### Example

```
PTELEPEN_PARAMS pTelepen = new TELEPEN_PARAMS();  
pTelepen->bEnable = m_bEnable;  
pTelepen->bNumeric = m_bNumeric;  
if(SCAN_SetTELEPEN(pTelepen) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetTELEPEN()", NULL, MB_TOPMOST);  
delete pTelepen;
```

## 4.1.41 SCAN\_SetUPCA

### Description

Sets the option of UPC-A Barcode.

**Syntax**

```
SCANNER_API BOOL SCAN_SetUPCA(PUPCA_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**

SCAN\_GetUPCA

**For .Net**

Namespace : ScannerNet.Scanner

Function : bool SetUPCA(ref UPCA\_PARAMS pUpca)

**Example**

```
PUPCA_PARAMS pUpca = new UPCA_PARAMS();  
pUpca->bEnable = m_bEnable;  
pUpca->bXNum = m_bXNum;  
pUpca->bXCD = m_bXCD;  
pUpca->bUPCA_AS_EAN13 = m_bUpcaAsEan13;  
pUpca->bAddOn = m_bAddOn;  
if(SCAN_SetUPCA(pUpca) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetUPCA()", NULL, MB_TOPMOST);  
delete pUpca;
```

## 4.1.42 SCAN\_SetUPCE

**Description**

Sets the option of UPC-E Barcode.

**Syntax**

```
SCANNER_API BOOL SCAN_SetUPCE(PUPCE_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**

SCAN\_GetUPCE

**For .Net**

Namespace : ScannerNet.Scanner

Function : bool SetUPCE(ref UPCE\_PARAMS pUpce)

**Example**

```
PUPCE_PARAMS pUpce = new UPCE_PARAMS();  
pUpce->bEnable = m_bEnable;  
pUpce->bXNum = m_bXNum;  
pUpce->bXCD = m_bXCD;  
pUpce->nConvert = m_nConvert;  
if(SCAN_SetUPCE(pUpce) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetUPCE()", NULL, MB_TOPMOST);  
delete pUpce;
```



## 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

Define
<pre>#define WM_SCAN_DATA WM_APP + 350 #define WM_IQ_DATA WM_APP + 380</pre>
Enum
<pre>typedef enum {     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 } SCAN_DEVICE_TYPE;  typedef enum {     SOUND_DEFAULT = 0,     SOUND_BEEP,     SOUND_NONE } SCAN_SOUND;  typedef enum {     CODE39_STANDARD = 0,     CODE39_CODE32,     CODE39_TRIOPTIC }CODE39_FORMAT;  typedef enum {     MODE_OCR_DISABLED = 0,     MODE_OCR_A,     MODE_OCR_B,     MODE_OCR_MONEY,</pre>

```

    MODE_OCR_MICR_UNSUPPORTED,
}OCR_MODE;

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

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

typedef enum {
    CAM_JPG = 0,
    CAM_BMP,
} CAM_FORMAT;

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

typedef enum {
    IQ_AZTEC = 0,
    IQ_CODE39
} IQ_TYPE;

```

## Structure

```

typedef struct _DECODER{
    BYTE bAZTEC;
    BYTE bCODABAR;
    BYTE bCODE11;
    BYTE bCODE128;
    BYTE bCODE39;
    BYTE bCODE49;
    BYTE bCODE93;
    BYTE bCOMPOSITE;
    BYTE bDATAMATRIX;
    BYTE bEAN8;
    BYTE bEAN13;
    BYTE bINT25;
    BYTE bMAXICODE;
    BYTE bMICROPDF;
    BYTE bOCR;
    BYTE bPDF417;
    BYTE bPOSTNET;
    BYTE bQR;
}

```

```

BYTE bRSS;
BYTE bUPCA;
BYTE bUPCE;
BYTE bISBT;
BYTE bBPO;
BYTE bCANPOST;
BYTE bAUSPOST;
BYTE bIATA25;
BYTE bCODABLOCK;
BYTE bJAPOST;
BYTE bPLANET;
BYTE bDUTCHPOST;
BYTE bMSI;
BYTE bTLCODE39;
BYTE bSTRT25;
BYTE bMATRIX25;
BYTE bPLESSEY;
BYTE bCHINAPOST;
BYTE bKOREAPOST;
BYTE bTELEPEN;
BYTE bCODE16K;
BYTE bPOSICODE;
BYTE bCOUPONCODE;
BYTE bUSPS4CB;
BYTE bIDTAG;
BYTE bLABEL;
BYTE bGS1_128;
BYTE bHANXIN;
BYTE bGRIDMATRIX;
}DECODER, *PDECODER;
typedef struct _DECODER_PARAMS{
    BYTE bCentering;
    BYTE bContinueMode;
    BYTE bXmitAimID;
    BYTE bHexMode;
    BYTE bVibrate;
    int  nSound;
    int  nTimeOut;
    int  nLightMode;
}DECODER_PARAMS, *PDECODER_PARAMS;
typedef struct _DECOPTION_PARAMS{
    int  nDecodeMode;
    int  nLinearRange;
    int  nPrintWeight;
    int  nMaxDecode;
    int  nMaxSearch;

```

```

    BOOL bVideoReverse;
}DECOPTION_PARAMS, *PDECOPTION_PARAMS;

typedef struct _CAM_PARAMS{
    int      nSaveMode;
    int      nSaveFormat;
    int      nJpegQuality;
    int      nResolution;
    TCHAR    szSaveFolder[256];
    TCHAR    szFileName[256];
} CAM_PARAMS, *PCAM_PARAMS;

typedef struct _IQ_PARAMS{
    int      nSaveMode;
    int      nIQType;
    TCHAR    szSaveFolder[256];
    TCHAR    szFileName[256];
} IQ_PARAMS, *PIQ_PARAMS;

typedef struct _AZTEC_PARAMS{
    BYTE     bEnable;
    int      nMinLen;
    int      nMaxLen;
}AZTEC_PARAMS, *PAZTEC_PARAMS;

typedef struct _CODABAR_PARAMS{
    BYTE     bEnable;
    BYTE     bCDV;
    BYTE     bXCD;
    BYTE     bXSS;
    int      nMinLen;
    int      nMaxLen;
}CODABAR_PARAMS, *PCODABAR_PARAMS;

typedef struct _CODE11_PARAMS{
    BYTE     bEnable;
    BYTE     bCDV;
    int      nMinLen;
    int      nMaxLen;
}CODE11_PARAMS, *PCODE11_PARAMS;

typedef struct _CODE128_PARAMS{
    BYTE     bEnable;
    int      nMinLen;
    int      nMaxLen;
}CODE128_PARAMS, *PCODE128_PARAMS;

typedef struct _CODE39_PARAMS{
    BYTE     bEnable;
    Int      nFormat;
    BYTE     bCDV;
    BYTE     bXCD;
    BYTE     bFullASCII;

```

```

    int    nMinLen;
    int    nMaxLen;
}CODE39_PARAMS, *PCODE39_PARAMS;

typedef struct _CODE49_PARAMS{
    BYTE    bEnable;
    int     nMinLen;
    int     nMaxLen;
}CODE49_PARAMS, *PCODE49_PARAMS;

typedef struct _CODE93_PARAMS{
    BYTE    bEnable;
    int     nMinLen;
    int     nMaxLen;
}CODE93_PARAMS, *PCODE93_PARAMS;

typedef struct _COMPOSITE_PARAMS{
    BYTE    bEnable;
    BYTE    bComposite_Upc;
    int     nMinLen;
    int     nMaxLen;
}COMPOSITE_PARAMS, *PCOMPOSITE_PARAMS;

typedef struct _DATAMATRIX_PARAMS{
    BYTE    bEnable;
    int     nMinLen;
    int     nMaxLen;
}DATAMATRIX_PARAMS, *PDATAMATRIX_PARAMS;

typedef struct _EAN8_PARAMS{
    BYTE    bEnable;
    BYTE    bXCD;
    BYTE    bAddOn;
}EAN8_PARAMS, *PEAN8_PARAMS;

typedef struct _EAN13_PARAMS{
    BYTE    bEnable;
    BYTE    bXCD;
    BYTE    bAddOn;
}EAN13_PARAMS, *PEAN13_PARAMS;

typedef struct _INT25_PARAMS{
    BYTE    bEnable;
    BYTE    bCDV;
    BYTE    bXCD;
    int     nMinLen;
    int     nMaxLen;
}INT25_PARAMS, *PINT25_PARAMS;

typedef struct _MAXICODE_PARAMS{
    BYTE    bEnable;
    int     nMinLen;
    int     nMaxLen;
}MAXICODE_PARAMS, *PMAXICODE_PARAMS;

```

```

typedef struct _MICROPDF_PARAMS{
    BYTE    bEnable;
    int     nMinLen;
    int     nMaxLen;
}MICROPDF_PARAMS, *PMICROPDF_PARAMS;

typedef struct _OCR_PARAMS{
    BYTE     bEnable;
    OCR_MODE nMode;
    TCHAR    szTemplate[256];
    TCHAR    szGroupG[256];
    TCHAR    szGroupH[256];
    TCHAR    szCheckChar[64];
}OCR_PARAMS, *POCR_PARAMS;

typedef struct _PDF417_PARAMS{
    BYTE    bEnable;
    int     nMinLen;
    int     nMaxLen;
}PDF417_PARAMS, *PPDF417_PARAMS;

typedef struct _POSTNET_PARAMS{
    BYTE    bEnable;
    BYTE    bXCD;
}POSTNET_PARAMS, *PPOSTNET_PARAMS;

typedef struct _QR_PARAMS{
    BYTE    bEnable;
    int     nMinLen;
    int     nMaxLen;
}QR_PARAMS, *PQR_PARAMS;

typedef struct _RSS_PARAMS{
    BYTE    bEnable;
    BYTE    bRssLim;
    BYTE    bRssExp;
    int     nMinLen;
    int     nMaxLen;
}RSS_PARAMS, *PRSS_PARAMS;

typedef struct _UPCA_PARAMS{
    BYTE    bEnable;
    BYTE    bXCD;
    BYTE    bXNum;
    BYTE    bAddOn;
}UPCA_PARAMS, *PUPCA_PARAMS;

typedef struct _UPCE_PARAMS{
    BYTE    bEnable;
    BYTE    bXCD;
    BYTE    bXNum;
    BYTE    bAddOn;
}UPCE_PARAMS, *PUPCE_PARAMS;

```

```
typedef struct _IATA25_PARAMS{
    BYTE    bEnable;
    int      nMinLen;
    int      nMaxLen;
}IATA25_PARAMS, *PIATA25_PARAMS;
typedef struct _CODABLOCK_PARAMS{
    BYTE    bEnable;
    int      nMinLen;
    int      nMaxLen;
}CODABLOCK_PARAMS, *PCODABLOCK_PARAMS;
typedef struct _PLANET_PARAMS{
    BYTE    bEnable;
    BYTE    bXCD;
}PLANET_PARAMS, *PPLANET_PARAMS;
typedef struct _MSI_PARAMS{
    BYTE    bEnable;
    BYTE    bXCD;
    int      nMinLen;
    int      nMaxLen;
}MSI_PARAMS, *PMSI_PARAMS;
typedef struct _STRT25_PARAMS{
    BYTE    bEnable;
    int      nMinLen;
    int      nMaxLen;
}STRT25_PARAMS, *PSTRT25_PARAMS;
typedef struct _MATRIX25_PARAMS{
    BYTE    bEnable;
    int      nMinLen;
    int      nMaxLen;
}MATRIX25_PARAMS, *PMATRIX25_PARAMS;
typedef struct _PLESSEY_PARAMS{
    BYTE    bEnable;
    int      nMinLen;
    int      nMaxLen;
}PLESSEY_PARAMS, *PPLESSEY_PARAMS;
typedef struct _CHINAPOST_PARAMS{
    BYTE    bEnable;
    int      nMinLen;
    int      nMaxLen;
}CHINAPOST_PARAMS, *PCHINAPOST_PARAMS;
typedef struct _KOREAPOST_PARAMS{
    BYTE    bEnable;
    int      nMinLen;
    int      nMaxLen;
}KOREAPOST_PARAMS, *PKOREAPOST_PARAMS;
typedef struct _TELEPEN_PARAMS{
```

```
    BYTE    bEnable;  
    BYTE    bNumeric;  
    int     nMinLen;  
    int     nMaxLen;  
}TELEPEN_PARAMS, *PTELEPEN_PARAMS;  
  
typedef struct _CODE16K_PARAMS{  
    BYTE    bEnable;  
    int     nMinLen;  
    int     nMaxLen;  
}CODE16K_PARAMS, *PCODE16K_PARAMS;  
  
typedef struct _POSICODE_PARAMS{  
    BYTE    bEnable;  
    BYTE    bPosi_Lim1;  
    BYTE    bPosi_Lim2;  
    int     nMinLen;  
    int     nMaxLen;  
}POSICODE_PARAMS, *PPOSICODE_PARAMS;
```



## Functions for 2D Imager

Name	Description
<a href="#"><u>IMAGER_CAMCapture</u></a>	Captures the preview of imaging
<a href="#"><u>IMAGER_CAMGetOption</u></a>	Gets the option of imaging
<a href="#"><u>IMAGER_CAMInit</u></a>	Initializes imaging device
<a href="#"><u>IMAGER_CAMPreviewStart</u></a>	Starts the preview of imaging
<a href="#"><u>IMAGER_CAMPreviewStop</u></a>	Stops the preview of imaging
<a href="#"><u>IMAGER_CAMSetOption</u></a>	Sets the option of imaging
<a href="#"><u>IMAGER_CAMUnInit</u></a>	Uninitializes imaging device
<a href="#"><u>IMAGER_Close</u></a>	Closes an open imager
<a href="#"><u>IMAGER_GetAZTEC</u></a>	Gets the option of AZTEC Barcode
<a href="#"><u>IMAGER_GetCenteringWindow</u></a>	Gets Enable/Disable of centering window function
<a href="#"><u>IMAGER_GetCHINAPOST</u></a>	Gets the option of CHINAPOST Barcode
<a href="#"><u>IMAGER_GetCODABAR</u></a>	Gets the option of CODABAR Barcode
<a href="#"><u>IMAGER_GetCODABLOCK</u></a>	Gets the option of CODABLOCK Barcode
<a href="#"><u>IMAGER_GetCODE11</u></a>	Gets the option of CODE11 Barcode
<a href="#"><u>IMAGER_GetCODE128</u></a>	Gets the option of CODE128 Barcode
<a href="#"><u>IMAGER_GetCODE16K</u></a>	Gets the option of CODE16K Barcode
<a href="#"><u>IMAGER_GetCODE39</u></a>	Gets the option of CODE39 Barcode
<a href="#"><u>IMAGER_GetCODE49</u></a>	Gets the option of CODE49 Barcode
<a href="#"><u>IMAGER_GetCODE93</u></a>	Gets the option of CODE93 Barcode
<a href="#"><u>IMAGER_GetCOMPOSITE</u></a>	Gets the option of COMPOSITE Barcode
<a href="#"><u>IMAGER_GetDATAMATRIX</u></a>	Gets the option of DATAMATRIX Barcode
<a href="#"><u>IMAGER_GetDecOption</u></a>	Gets the option of Decoder
<a href="#"><u>IMAGER_GetDeviceType</u></a>	Gets the type of device
<a href="#"><u>IMAGER_GetEAN13</u></a>	Gets the option of EAN-13 Barcode
<a href="#"><u>IMAGER_GetEAN8</u></a>	Gets the option of EAN-8 Barcode
<a href="#"><u>IMAGER_GetIATA25</u></a>	Gets the option of IATA25 Barcode
<a href="#"><u>IMAGER_GetINT25</u></a>	Gets the option of INT25 Barcode
<a href="#"><u>IMAGER_GetKOREAPOST</u></a>	Gets the option of KOREAPOST Barcode
<a href="#"><u>IMAGER_GetMATRIX25</u></a>	Gets the option of MATRIX25 Barcode
<a href="#"><u>IMAGER_GetMAXICODE</u></a>	Gets the option of MAXICODE Barcode

<a href="#"><u>IMAGER_GetMICROPDF</u></a>	Gets the option of MICROPDF Barcode
<a href="#"><u>IMAGER_GetMSI</u></a>	Gets the option of MSI Barcode
<a href="#"><u>IMAGER_GetOCR</u></a>	Gets the option of OCR Barcode
<a href="#"><u>IMAGER_GetOption</u></a>	Gets the option of imager
<a href="#"><u>IMAGER_GetPDF417</u></a>	Gets the option of PDF417 Barcode
<a href="#"><u>IMAGER_GetPLANET</u></a>	Gets the option of PLANET Barcode
<a href="#"><u>IMAGER_GetPLESSEY</u></a>	Gets the option of PLESSEY Barcode
<a href="#"><u>IMAGER_GetPOSICODE</u></a>	Gets the option of POSICODE Barcode
<a href="#"><u>IMAGER_GetPOSTNET</u></a>	Gets the option of POSTNET Barcode
<a href="#"><u>IMAGER_GetQR</u></a>	Gets the option of QR Barcode
<a href="#"><u>IMAGER_GetRSS</u></a>	Gets the option of RSS Barcode
<a href="#"><u>IMAGER_GetScanData</u></a>	Gets the ScanData which is read by imager
<a href="#"><u>IMAGER_GetSTRT25</u></a>	Gets the option of STRT25 Barcode
<a href="#"><u>IMAGER_GetSymbology</u></a>	Gets Enable/Disable of Symbologies
<a href="#"><u>IMAGER_GetTELEPEN</u></a>	Gets the option of TELEPEN Barcode
<a href="#"><u>IMAGER_GetUPCA</u></a>	Gets the option of UPC-A Barcode
<a href="#"><u>IMAGER_GetUPCE</u></a>	Gets the option of UPC-E Barcode
<a href="#"><u>IMAGER_GetVersionInfo</u></a>	Gets the information of imager driver and dll version
<a href="#"><u>IMAGER_IQGetBarcodeData</u></a>	Gets the ScanData which is read by imager
<a href="#"><u>IMAGER_IQGetOption</u></a>	Gets the option of IQ Imaging
<a href="#"><u>IMAGER_IQImagingStart</u></a>	Starts IQ Imaging
<a href="#"><u>IMAGER_IQImagingStop</u></a>	Stops IQ Imaging
<a href="#"><u>IMAGER_IQInit</u></a>	Initializes IQ imaging
<a href="#"><u>IMAGER_IQSetOption</u></a>	Sets the option of IQ Imaging
<a href="#"><u>IMAGER_IQUnInit</u></a>	Uninitializes IQ imaging
<a href="#"><u>IMAGER_Open</u></a>	Opens a imager
<a href="#"><u>IMAGER_Read</u></a>	Starts the beaming of imager
<a href="#"><u>IMAGER_ReadCancel</u></a>	Stops the beaming of imager
<a href="#"><u>IMAGER_SetAZTEC</u></a>	Sets the option of AZTEC Barcode
<a href="#"><u>IMAGER_SetCenteringWindow</u></a>	Enables centering window function
<a href="#"><u>IMAGER_SetCHINAPOST</u></a>	Sets the option of CHINAPOST Barcode
<a href="#"><u>IMAGER_SetCODABAR</u></a>	Sets the option of CODABAR Barcode

<a href="#"><u>IMAGER_SetCODABLOCK</u></a>	Sets the option of CODABLOCK Barcode
<a href="#"><u>IMAGER_SetCODE11</u></a>	Sets the option of CODE11 Barcode
<a href="#"><u>IMAGER_SetCODE128</u></a>	Sets the option of CODE128 Barcode
<a href="#"><u>IMAGER_SetCODE16K</u></a>	Sets the option of CODE16K Barcode
<a href="#"><u>IMAGER_SetCODE39</u></a>	Sets the option of CODE39 Barcode
<a href="#"><u>IMAGER_SetCODE49</u></a>	Sets the option of CODE49 Barcode
<a href="#"><u>IMAGER_SetCODE93</u></a>	Sets the option of CODE93 Barcode
<a href="#"><u>IMAGER_SetCOMPOSITE</u></a>	Sets the option of COMPOSITE Barcode
<a href="#"><u>IMAGER_SetDATAMATRIX</u></a>	Sets the option of DATAMATRIX Barcode
<a href="#"><u>IMAGER_SetDecOption</u></a>	Sets the option of Decoder
<a href="#"><u>IMAGER_SetEAN13</u></a>	Sets the option of EAN-13 Barcode
<a href="#"><u>IMAGER_SetEAN8</u></a>	Sets the option of EAN-8 Barcode
<a href="#"><u>IMAGER_SetIATA25</u></a>	Sets the option of IATA25 Barcode
<a href="#"><u>IMAGER_SetINT25</u></a>	Sets the option of INT25 Barcode
<a href="#"><u>IMAGER_SetKOREAPOST</u></a>	Sets the option of KOREAPOST Barcode
<a href="#"><u>IMAGER_SetMATRIX25</u></a>	Sets the option of MATRIX25 Barcode
<a href="#"><u>IMAGER_SetMAXICODE</u></a>	Sets the option of MAXICODE Barcode
<a href="#"><u>IMAGER_SetMICROPDF</u></a>	Sets the option of MICROPDF Barcode
<a href="#"><u>IMAGER_SetMSI</u></a>	Sets the option of MSI Barcode
<a href="#"><u>IMAGER_SetOCR</u></a>	Sets the option of OCR Barcode
<a href="#"><u>IMAGER_SetOption</u></a>	Sets the option of imager
<a href="#"><u>IMAGER_SetPDF417</u></a>	Sets the option of PDF417 Barcode
<a href="#"><u>IMAGER_SetPLANET</u></a>	Sets the option of PLANET Barcode
<a href="#"><u>IMAGER_SetPLESSEY</u></a>	Sets the option of PLESSEY Barcode
<a href="#"><u>IMAGER_SetPOSICODE</u></a>	Sets the option of POSICODE Barcode
<a href="#"><u>IMAGER_SetPOSTNET</u></a>	Sets the option of POSTNET Barcode
<a href="#"><u>IMAGER_SetQR</u></a>	Sets the option of QR Barcode
<a href="#"><u>IMAGER_SetRSS</u></a>	Sets the option of RSS Barcode
<a href="#"><u>IMAGER_SetSTRT25</u></a>	Sets the option of STRT25 Barcode
<a href="#"><u>IMAGER_SetSymbology</u></a>	Sets the Enable/Disable of Symbologies
<a href="#"><u>IMAGER_SetSymbologyAll</u></a>	Enables all of Symbologies
<a href="#"><u>IMAGER_SetSymbologyDefault</u></a>	Initializes all option of scanner

<a href="#"><u>IMAGER_SetTELEPEN</u></a>	Sets the option of TELEPEN Barcode
<a href="#"><u>IMAGER_SetUPCA</u></a>	Sets the option of UPC-A Barcode
<a href="#"><u>IMAGER_SetUPCE</u></a>	Sets the option of UPC-E Barcode

## 4.2.1 IMAGER\_CAMCapture

### Description

Captures the preview of imaging.

### Syntax

```
IMAGER_API BOOL IMAGER_CAMCapture();
```

### Parameters

None

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

IMAGER\_CAMPreviewStart, IMAGER\_CAMPreviewStop

### For .Net

Namespace : ImagerNet.Imager

Function : bool CAMCapture()

### Example

None

## 4.2.2 IMAGER\_CAMGetOption

### Description

Gets the option of imaging.

### Syntax

```
IMAGER_API BOOL IMAGER_CAMGetOption(PCAM_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

IMAGER\_CAMSetOption

### For .Net

Namespace : ImagerNet.Imager

Function : bool CAMGetOption(out CAM\_PARAMS pCamOption)

#### Example

```
PCAM_PARAMS pCamOption = new CAM_PARAMS();
if(!IMAGER_CAMGetOption(pCamOption) == FALSE)
{
    ::MessageBox(NULL, L"Error : IMAGER_CAMGetOption()", NULL, MB_TOPMOST);
    return FALSE;
}
m_nSaveFormat = pCamOption->nSaveFormat;
m_nJpegQuality = pCamOption->nJpegQuality;
m_strSaveFolder = pCamOption->szSaveFolder;
m_strFileName = pCamOption->szFileName;
m_ctrlComboSaveMode.SetCurSel(pCamOption->nSaveMode);
m_ctrlComboResolution.SetCurSel(pCamOption->nResolution);
delete pCamOption;
```

### 4.2.3 IMAGER\_CAMInit

#### Description

Initializes imaging device.

#### Syntax

```
IMAGER_API BOOL IMAGER_CAMInit(HWND hPictureWnd);
```

#### Parameters

*hPictureWnd*

Window that will show preview.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

IMAGER\_CAMUnInit

#### For .Net

Namespace : ImagerNet.Imager

Function : bool CAMInit(IntPtr hPictureWnd)

## Example

None

### 4.2.4 IMAGER\_CAMPreviewStart

#### Description

Starts the preview of imaging.

#### Syntax

```
IMAGER_CAMPreviewStart()
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

IMAGER\_CAMPreviewStop

#### For .Net

Namespace : ImagerNet.Imager

Function : bool CAMPreviewStop()

#### Example

None

### 4.2.5 IMAGER\_CAMPreviewStop

#### Description

Stops the preview of imaging.

#### Syntax

```
IMAGER_API BOOL IMAGER_CAMPreviewStop();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

IMAGER\_CAMPreviewStart

#### For .Net

Namespace : ImagerNet.Imager

Function : bool CAMPreviewStart()

#### Example

None

### 4.2.6 IMAGER\_CAMSetOption

#### Description

Sets the option of imaging.

#### Syntax

IMAGER\_API BOOL IMAGER\_CAMSetOption(PCAM\_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

IMAGER\_CAMGetOption

#### For .Net

Namespace : ImagerNet.Imager

Function : bool CAMSetOption(ref CAM\_PARAMS pCamOption)

#### Example

```
PCAM_PARAMS pCamOption = new CAM_PARAMS();
pCamOption->nSaveFormat = m_nSaveFormat;
pCamOption->nJpegQuality = m_nJpegQuality;
pCamOption->nSaveMode = m_ctrlComboSaveMode.GetCurSel();
pCamOption->nResolution = m_ctrlComboResolution.GetCurSel();
wsprintf(pCamOption->szSaveFolder, L"%s", m_strSaveFolder);
wsprintf(pCamOption->szFileName, L"%s", m_strFileName);
if(IMAGER_CAMSetOption(pCamOption) == FALSE)
{
```



```
        ::MessageBox(NULL, L"Error : IMAGER_CAMSetOption()", NULL, MB_TOPMOST);  
        return FALSE;  
    }  
    delete pCamOption;
```

### 4.2.7 IMAGER\_CAMUnInit

#### Description

Uninitializes imaging device.

#### Syntax

```
IMAGER_API BOOL IMAGER_CAMUnInit();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

IMAGER\_CAMInit

#### For .Net

Namespace : ImagerNet.Imager

Function : bool CAMUnInit()

#### Example

None

### 4.2.8 IMAGER\_Close

#### Description

Closes an open imager.

#### Syntax

```
IMAGER_API BOOL IMAGER_Close();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

IMAGER\_Open

#### For .Net

Namespace : ImagerNet.Imager

Function : bool Close()

#### Example

```
if(IMAGER_Close() == FALSE)
{
    ::MessageBox(NULL, L"Error : IMAGER_Close()", NULL, MB_TOPMOST);
}
```

### 4.2.9 IMAGER\_GetAZTEC

#### Description

Gets the option of AZTEC Barcode.

#### Syntax

IMAGER\_API BOOL IMAGER\_GetAZTEC(PAZTEC\_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

IMAGER\_SetAZTEC

#### For .Net

Namespace : ImagerNet.Imager

Function : bool GetAZTEC(out AZTEC\_PARAMS pAztec)

#### Example

```
PAZTEC_PARAMS    pAztec = new AZTEC_PARAMS();
if(IMAGER_GetAZTEC(pAztec) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetAZTEC()", NULL, MB_TOPMOST);
m_ bEnable = pAztec->bEnable;
```

```
m_nMinLen = pAztec->nMinLen;  
m_nMaxLen = pAztec->nMaxLen;  
delete pAztec;
```

#### 4.2.10 IMAGER\_GetCenteringWindow

##### Description

Gets Enable/Disable of centering window function.

##### Syntax

```
IMAGER_API BOOL IMAGER_GetCenteringWindow(RECT* pRect);
```

##### Parameters

*pRect*

Defines the region of the image.

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

IMAGER\_SetCenteringWindow

##### For .Net

Namespace : ImagerNet.Imager

Function : bool GetCenteringWindow(out RECT\_PARAM pRect)

##### Example

```
RECT rect;  
if(!IMAGER_GetCenteringWindow(&rect) == FALSE)  
{  
    ::MessageBox(NULL, L"ERROR : IMAGER_GetCenteringWindow()", NULL, MB_TOPMOST);  
    return FALSE;  
}  
m_nEditTop = rect.top;  
m_nEidtBottom = rect.bottom;  
m_nEditLeft = rect.left;  
m_nEditRight = rect.right;
```

## 4.2.11 IMAGER\_GetCHINAPOST

### Description

Gets the option of CHINAPOST Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_GetCHINAPOST(PCHINAPOST_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

IMAGER\_SetCHINAPOST

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetCHINAPOST(out CHINAPOST\_PARAMS pChinaPost)

### Example

```
PCHINAPOST_PARAMS pChinapost = new CHINAPOST_PARAMS();  
if(IMAGER_GetCHINAPOST(pChinapost) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_GetCHINAPOST()", NULL, MB_TOPMOST);  
m_bEnable = pChinapost->bEnable;  
m_nMinLen = pChinapost->nMinLen;  
m_nMaxLen = pChinapost->nMaxLen;  
delete pChinapost;
```

## 4.2.12 IMAGER\_GetCODABAR

### Description

Gets the option of CODABAR Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_GetCODABAR(PCODABAR_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

IMAGER\_SetCODABAR

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetCODABAR(out CODABAR\_PARAMS pCodabar)

### Example

```
PCODABAR_PARAMS pCodabar = new CODABAR_PARAMS();
if(IMAGER_GetCODABAR(pCodabar) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCODABAR()", NULL, MB_TOPMOST);
m_bEnable = pCodabar->bEnable;
m_bCDV = pCodabar->bCDV;
m_bXCD = pCodabar->bXCD;
m_bXSS = pCodabar->bXSS;
m_nMinLen = pCodabar->nMinLen;
m_nMaxLen = pCodabar->nMaxLen;
delete pCodabar;
```

## 4.2.13 IMAGER\_GetCODABLOCK

### Description

Gets the option of CODABLOCK Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_GetCODABLOCK(PCODABLOCK_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

IMAGER\_SetCODABLOCK

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetCODABLOCK(out CODABLOCK\_PARAMS pCodablock);

### Example

```
PCODABLOCK_PARAMS pCodablock = new CODABLOCK_PARAMS();
if(!IMAGER_GetCODABLOCK(pCodablock) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCODABLOCK()", NULL, MB_TOPMOST);
m_bEnable = pCodablock->bEnable;
m_nMinLen = pCodablock->nMinLen;
m_nMaxLen = pCodablock->nMaxLen;
delete pCodablock;
```

## 4.2.14 IMAGER\_GetCODE11

### Description

Gets the option of CODE11 Barcode.

### Syntax

IMAGER\_API BOOL IMAGER\_GetCODE11(PCODE11\_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

IMAGER\_SetCODE11

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetCODE11(out CODE11\_PARAMS pCode11)

### Example

```
PCODE11_PARAMS pCode11 = new CODE11_PARAMS();
```

```

if(IMAGER_GetCODE11(pCode11) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCODE11()", NULL, MB_TOPMOST);
m_bEnable = pCode11->bEnable;
m_bCDV = pCode11->bCDV;
m_nMinLen = pCode11->nMinLen;
m_nMaxLen = pCode11->nMaxLen;
delete pCode11;

```

## 4.2.15 IMAGER\_GetCODE128

### Description

Gets the option of CODE128 Barcode

### Syntax

```
IMAGER_API BOOL IMAGER_GetCODE128(PCODE128_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

IMAGER\_SetCODE128

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetCODE128(out CODE128\_PARAMS pCode128)

### Example

```

PCODE128_PARAMS    pCode128 = new CODE128_PARAMS();
if(IMAGER_GetCODE128(pCode128) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCODE128()", NULL, MB_TOPMOST);
m_bEnable = pCode128->bEnable;
m_nMinLen = pCode128->nMinLen;
m_nMaxLen = pCode128->nMaxLen;
delete pCode128;

```

## 4.2.16 IMAGER\_GetCODE16K

### Description

Gets the option of CODE16K Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_GetCODE16K(PCODE16K_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

IMAGER\_SetCODE16K

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetCODE16K(out CODE16K\_PARAMS pCode16k)

### Example

```
PCODE16K_PARAMS    pCode16k = new CODE16K_PARAMS();
if(!IMAGER_GetCODE16K(pCode16k) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCODE16K()", NULL, MB_TOPMOST);
m_bEnable = pCode16k->bEnable;
m_nMinLen = pCode16k->nMinLen;
m_nMaxLen = pCode16k->nMaxLen;
delete pCode16k;
```

## 4.2.17 IMAGER\_GetCODE39

### Description

Gets the option of CODE39 Barcode

### Syntax

```
IMAGER_API BOOL IMAGER_GetCODE39(PCODE39_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**

IMAGER\_SetCODE39

**For .Net**

Namespace : ImagerNet.Imager

Function : bool GetCODE39(out CODE39\_PARAMS pCode39)

**Example**

```
PCODE39_PARAMS pCode39 = new CODE39_PARAMS();
if(IMAGER_GetCODE39(pCode39) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCODE39()", NULL, MB_TOPMOST);

m_bEnable = pCode39->bEnable;
m_nFormat = pCode39->nFormat;
m_bCDV = pCode39->bCDV;
m_bXCD = pCode39->bXCD;
m_bFullASCII = pCode39->bFullASCII;
m_nMinLen = pCode39->nMinLen;
m_nMaxLen = pCode39->nMaxLen;
delete pCode39;
```

## 4.2.18 IMAGER\_GetCODE49

**Description**

Gets the option of CODE49 Barcode.

**Syntax**

```
IMAGER_API BOOL IMAGER_GetCODE49(PCODE49_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

IMAGER\_SetCODE49

## For .Net

Namespace : ImagerNet.Imager

Function : bool GetCODE49(out CODE49\_PARAMS pCode49)

## Example

```
PCODE49_PARAMS    pCode49 = new CODE49_PARAMS();
if(IMAGER_GetCODE49(pCode49) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCODE49()", NULL, MB_TOPMOST);

m_bEnable = pCode49->bEnable;
m_nMinLen = pCode49->nMinLen;
m_nMaxLen = pCode49->nMaxLen;
delete pCode49;
```

## 4.2.19 IMAGER\_GetCODE93

### Description

Gets the option of CODE93 Barcode.

### Syntax

IMAGER\_API BOOL IMAGER\_GetCODE93(PCODE93\_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

IMAGER\_SetCODE93

## For .Net

Namespace : ImagerNet.Imager

Function : bool GetCODE93(out CODE93\_PARAMS pCode93)

## Example

```
PCODE93_PARAMS    pCode93 = new CODE93_PARAMS();
if(IMAGER_GetCODE93(pCode93) == FALSE)
```

```

        ::MessageBox(NULL, L"Error : IMAGER_GetCODE93()", NULL, MB_TOPMOST);
m_bEnable    = pCode93->bEnable;
m_nMinLen    = pCode93->nMinLen;
m_nMaxLen    = pCode93->nMaxLen;
delete pCode93;

```

## 4.2.20 IMAGER\_GetCOMPOSITE

### Description

Gets the option of COMPOSITE Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_GetCOMPOSITE(PCOMPOSITE_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

IMAGER\_SetCOMPOSITE

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetCOMPOSITE(out COMPOSITE\_PARAMS pComposite)

### Example

```

PCOMPOSITE_PARAMS pComposite = new COMPOSITE_PARAMS();
if(IMAGER_GetCOMPOSITE(pComposite) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCOMPOSITE()", NULL, MB_TOPMOST);
m_bEnable = pComposite->bEnable;
m_bComposite_Upc = pComposite->bComposite_Upc;
m_nMinLen = pComposite->nMinLen;
m_nMaxLen = pComposite->nMaxLen;
delete pComposite;

```

## 4.2.21 IMAGER\_GetDATAMATRIX

### Description

Gets the option of DATAMATRIX Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_GetDATAMATRIX(PDATAMATRIX_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

IMAGER\_SetDATAMATRIX

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetDATAMATRIX(out DATAMATRIX\_PARAMS pDataMatirx)

### Example

```
PDATAMATRIX_PARAMS    pDatamatrix = new DATAMATRIX_PARAMS();  
if(IMAGER_GetDATAMATRIX(pDatamatrix) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_GetDATAMATRIX()", NULL, MB_TOPMOST);  
m_bEnable = pDatamatrix->bEnable;  
m_nMinLen = pDatamatrix->nMinLen;  
m_nMaxLen = pDatamatrix->nMaxLen;  
delete pDatamatrix;
```

## 4.2.22 IMAGER\_GetDecOption

### Description

Gets the option of Decoder.

### Syntax

```
IMAGER_API BOOL IMAGER_GetDecOption(PDECOPTION_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**

IMAGER\_SetDecOption

**For .Net**

Namespace : ImagerNet.Imager

Function : bool GetDecOption(out DECOPTION\_PARAMS pDecOption)

**Example**

```
PDECOPTION_PARAMS pDecOption = new DECOPTION_PARAMS();
if(IMAGER_GetDecOption(pDecOption) == FALSE)
{
    ::MessageBox(NULL, L"ERROR : IMAGER_GetDecOption()", NULL, MB_TOPMOST);
    return FALSE;
}
m_ctrlComboDecodeMode.SetCurSel(pDecOption->nDecodeMode);
m_nLinearRange = pDecOption->nLinearRange;
m_nPrintWeight = pDecOption->nPrintWeight;
m_nMaxDecode = pDecOption->nMaxDecode;
m_nMaxSearch = pDecOption->nMaxSearch;
m_bVideoReverse = pDecOption->bVideoReverse;
delete pDecOption;
```

## 4.2.23 IMAGER\_GetDeviceType

**Description**

Gets the type of device.

**Syntax**

```
IMAGER_API SCAN_DEVICE_TYPE IMAGER_GetDeviceType();
```

**Parameters**

None

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

#### See Also

None

#### For .Net

Namespace : ImagerNet.Imager

Function : SCAN\_DEVICE\_TYPE GetDeviceType()

#### Example

None

### 4.2.24 IMAGER\_GetEAN13

#### Description

Gets the option of EAN-13 Barcode.

#### Syntax

```
IMAGER_API BOOL IMAGER_GetEAN13(PEAN13_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

IMAGER\_SetEAN13

#### For .Net

Namespace : ImagerNet.Imager

Function : bool GetEAN13(out EAN13\_PARAMS pEan13)

#### Example

```
PEAN13_PARAMS pEan13 = new EAN13_PARAMS();  
if(IMAGER_GetEAN13(pEan13) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_GetEAN13()", NULL, MB_TOPMOST);  
m_bEnable = pEan13->bEnable;  
m_bXCD = pEan13->bXCD;  
m_bAddOn = pEan13->bAddOn;  
delete pEan13;
```

## 4.2.25 IMAGER\_GetEAN8

### Description

Gets the option of EAN-8 Barcode

### Syntax

```
IMAGER_API BOOL IMAGER_GetEAN8(PEAN8_PARAMS pEan8);
```

### Parameters

*pEan8*

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

IMAGER\_SetEAN8

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetEAN8(out EAN8\_PARAMS pEan8)

### Example

```
PEAN8_PARAMS pEan8 = new EAN8_PARAMS();  
if(!IMAGER_GetEAN8(pEan8) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_GetEAN8()", NULL, MB_TOPMOST);  
m_bEnable = pEan8->bEnable;  
m_bXCD = pEan8->bXCD;  
m_bAddOn = pEan8->bAddOn;  
delete pEan8;
```

## 4.2.26 IMAGER\_GetIATA25

### Description

Gets the option of IATA25 Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_GetIATA25(PIATA25_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**

IMAGER\_SetIATA25

**For .Net**

Namespace : ImagerNet.Imager

Function : bool GetIATA25(out IATA25\_PARAMS plata25)

**Example**

```
PIATA25_PARAMS    plata25 = new IATA25_PARAMS();
if(IMAGER_GetIATA25(plata25) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetIATA25()", NULL, MB_TOPMOST);
m_ bEnable = plata25->bEnable;
m_nMinLen = plata25->nMinLen;
m_nMaxLen = plata25->nMaxLen;
delete plata25;
```

## 4.2.27 IMAGER\_GetINT25

**Description**

Sets the option of INT25 Barcode.

**Syntax**

IMAGER\_API BOOL IMAGER\_GetINT25(PINT25\_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**

IMAGER\_SetINT25

**For .Net**

Namespace : ImagerNet.Imager



Function : public bool GetINT25(out INT25\_PARAMS pInt25)

#### Example

```
PINT25_PARAMS pInt25 = new INT25_PARAMS();
if(IMAGER_GetINT25(pInt25) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetINT25()", NULL, MB_TOPMOST);
m_bEnable = pInt25->bEnable;
m_bCDV = pInt25->bCDV;
m_bXCD = pInt25->bXCD;
m_nMinLen = pInt25->nMinLen;
m_nMaxLen = pInt25->nMaxLen;
delete pInt25;
```

### 4.2.28 IMAGER\_GetKOREAPOST

#### Description

Gets the option of KOREAPOST Barcode.

#### Syntax

```
IMAGER_API BOOL IMAGER_GetKOREAPOST(PKOREAPOST_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

IMAGER\_SetKOREAPOST

#### For .Net

Namespace : ImagerNet.Imager

Function : bool GetKOREAPOST(out KOREAPOST\_PARAMS pKoreaPost)

#### Example

```
PKOREAPOST_PARAMS pKoreapost= new KOREAPOST_PARAMS();
if(IMAGER_GetKOREAPOST(pKoreapost) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetKOREAPOST()", NULL, MB_TOPMOST);
m_bEnable = pKoreapost->bEnable;
```

```
m_nMinLen = pKoreapost->nMinLen;  
m_nMaxLen = pKoreapost->nMaxLen;  
delete pKoreapost;
```

### 4.2.29 IMAGER\_GetMATRIX25

#### Description

Gets the option of MATRIX25 Barcode.

#### Syntax

```
IMAGER_API BOOL IMAGER_GetMATRIX25(PMATRIX25_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

IMAGER\_SetMATRIX25

#### For .Net

Namespace : ImagerNet.Imager

Function : bool GetMATRIX25(out MATRIX25\_PARAMS pMatrix25)

#### Example

```
PMATRIX25_PARAMS pMatrix25 = new MATRIX25_PARAMS();  
if(IMAGER_GetMATRIX25(pMatrix25) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_GetMATRIX25()", NULL, MB_TOPMOST);  
m_bEnable = pMatrix25->bEnable;  
m_nMinLen = pMatrix25->nMinLen;  
m_nMaxLen = pMatrix25->nMaxLen;  
delete pMatrix25;
```

### 4.2.30 IMAGER\_GetMAXICODE

#### Description

Gets the option of MAXICODE Barcode

#### Syntax

IMAGER\_API BOOL IMAGER\_GetMAXICODE(PMAXICODE\_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

IMAGER\_SetMAXICODE

#### For .Net

Namespace : ImagerNet.Imager

Function : bool GetMAXICODE(out MAXICODE\_PARAMS pMaxiCode)

#### Example

```
PMAXICODE_PARAMS    pMaxicode = new MAXICODE_PARAMS();
if(!IMAGER_GetMAXICODE(pMaxicode) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetMAXICODE()", NULL, MB_TOPMOST);
m_ bEnable = pMaxicode->bEnable;
m_nMinLen = pMaxicode->nMinLen;
m_nMaxLen = pMaxicode->nMaxLen;
delete pMaxicode;
```

### 4.2.31 IMAGER\_GetMICROPDF

#### Description

Gets the option of MICROPDF Barcode

#### Syntax

IMAGER\_API BOOL IMAGER\_GetMICROPDF(PMICROPDF\_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

IMAGER\_SetMICROPDF

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetMICROPDF(out MICROPDF\_PARAMS pMicroPdf)

### Example

```
PMICROPDF_PARAMS pMicropdf = new MICROPDF_PARAMS();  
if(IMAGER_GetMICROPDF(pMicropdf) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_GetMICROPDF()", NULL, MB_TOPMOST);  
m_bEnable = pMicropdf->bEnable;  
m_nMinLen = pMicropdf->nMinLen;  
m_nMaxLen = pMicropdf->nMaxLen;  
delete pMicropdf;
```

## 4.2.32 IMAGER\_GetMSI

### Description

Gets the option of MSI Barcode

### Syntax

IMAGER\_API BOOL IMAGER\_GetMSI(PMSI\_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

IMAGER\_SetMSI

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetMSI(out MSI\_PARAMS pMsi)

### Example

```
PMSI_PARAMS pMsi = new MSI_PARAMS();
```

```

if(IMAGER_GetMSI(pMsi) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetMSI()", NULL, MB_TOPMOST);
m_bEnable = pMsi->bEnable;
m_bXCD = pMsi->bXCD;
m_nMinLen = pMsi->nMinLen;
m_nMaxLen = pMsi->nMaxLen;
delete pMsi;

```

### 4.2.33 IMAGER\_GetOCR

#### Description

Gets the option of OCR Barcode.

#### Syntax

```
IMAGER_API BOOL IMAGER_GetOCR(POCR_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

IMAGER\_SetOCR

#### For .Net

Namespace : ImagerNet.Imager

Function : bool GetOCR(out OCR\_PARAMS pOcr)

#### Example

```

POCR_PARAMS pOcr = new OCR_PARAMS();
if(IMAGER_GetOCR(pOcr) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetOcr()", NULL, MB_TOPMOST)
m_ctrlComboMode.InsertString(0, L"OCR_DISABLED");
m_ctrlComboMode.InsertString(1, L"OCR_A");
m_ctrlComboMode.InsertString(2, L"OCR_B");
m_ctrlComboMode.InsertString(3, L"OCR_MONEY");
m_ctrlComboMode.InsertString(4, L"OCR_MICR_UNSUPPORTED");

```

```

m_ctrlComboMode.SetCurSel(pOcr->nMode);
m_bEnable = pOcr->bEnable;
m_strEditTemplate = (CString)pOcr->szTemplate;
m_strEditGroupG = (CString)pOcr->szGroupG;
m_strEditGroupH = (CString)pOcr->szGroupH;
m_strCheckChar = (CString)pOcr->szCheckChar;
delete pOcr;

```

### 4.2.34 IMAGER\_GetOption

#### Description

Gets the option of imager.

#### Syntax

```
IMAGER_API BOOL IMAGER_GetOption(PDECODER_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

IMAGER\_SetOption

#### For .Net

Namespace : ImagerNet.Imager

Function : bool GetOption(out DECODER\_PARAMS pOption)

#### Example

```

PDECODER_PARAMS pOption = new DECODER_PARAMS();
if(IMAGER_GetOption(pOption) == FALSE)
{
    ::MessageBox(NULL, L"ERROR : IMAGER_GetOption()", NULL, MB_TOPMOST);
    return FALSE;
}
m_nSound = pOption->nSound;
m_bCentering = pOption->bCentering;

```

```

m_bVibrate = pOption->bVibrate;
m_bAimID = pOption->bXmitAimID;
m_bContinueMode = pOption->bContinueMode;
m_bHexMode = pOption->bHexMode;
m_ctrlComboTimeOut.SetCurSel(pOption->nTimeOut - 1);
m_ctrlComboLightMode.SetCurSel(pOption->nLightMode);
delete pOption;

```

### 4.2.35 IMAGER\_GetPDF417

#### Description

Gets the option of PDF417 Barcode.

#### Syntax

```
IMAGER_API BOOL IMAGER_GetPDF417(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

IMAGER\_SetPDF417

#### For .Net

Namespace : ImagerNet.Imager

Function : bool GetPDF417(out PDF417\_PARAMS pPdf417)

#### Example

```

PDF417_PARAMS      pPdf417 = new PDF417_PARAMS();
if(IMAGER_GetPDF417(pPdf417) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetPDF417()", NULL, MB_TOPMOST);
m_bEnable = pPdf417->bEnable;
m_nMinLen = pPdf417->nMinLen;
m_nMaxLen = pPdf417->nMaxLen;
delete pPdf417;

```

### 4.2.36 IMAGER\_GetPLANET

#### Description

Gets the option of PLANET Barcode.

#### Syntax

```
IMAGER_API BOOL IMAGER_GetPLANET(PPLANET_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

IMAGER\_SetPLANET

#### For .Net

Namespace : ImagerNet.Imager

Function : public bool GetPLANET(out PLANET\_PARAMS pPlanet)

#### Example

```
PPLANET_PARAMS pPlanet = new PLANET_PARAMS();  
if(IMAGER_GetPLANET(pPlanet) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_GetPLANET()", NULL, MB_TOPMOST);  
m_bEnable = pPlanet->bEnable;  
m_bXCDD = pPlanet->bXCDD;  
delete pPlanet;
```

### 4.2.37 IMAGER\_GetPLESSEY

#### Description

Gets the option of PLESSEY Barcode.

#### Syntax

```
IMAGER_API BOOL IMAGER_GetPLESSEY(PPLESSEY_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

IMAGER\_SetPLESSEY

#### For .Net

Namespace : ImagerNet.Imager

Function : bool GetPLESSEY(out PLESSEY\_PARAMS pPlessey)

#### Example

```
PPLESSEY_PARAMS      pPlessey = new PLESSEY_PARAMS();
if(IMAGER_GetPLESSEY(pPlessey) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetPLESSEY()", NULL, MB_TOPMOST);
m_bEnable = pPlessey->bEnable;
m_nMinLen = pPlessey->nMinLen;
m_nMaxLen = pPlessey->nMaxLen;
delete pPlessey;
```

## 4.2.38 IMAGER\_GetPOSICODE

#### Description

Gets the option of POSICODE Barcode.

#### Syntax

IMAGER\_API BOOL IMAGER\_GetPOSICODE(PPOSICODE\_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

IMAGER\_SetPOSICODE

#### For .Net

Namespace : ImagerNet.Imager

Function : bool GetPOSICODE(out POSICODE\_PARAMS pPosiCode);

### Example

```
PPOSICODE_PARAMS pPosicode = new POSICODE_PARAMS();
if(IMAGER_GetPOSIcode(pPosicode) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetPOSIcode()", NULL, MB_TOPMOST);
m_bEnable = pPosicode->bEnable;
m_bPosi_Lim1 = pPosicode->bPosi_Lim1;
m_bPosi_Lim2 = pPosicode->bPosi_Lim2;
m_nMinLen = pPosicode->nMinLen;
m_nMaxLen = pPosicode->nMaxLen;
delete pPosicode;
```

## 4.2.39 IMAGER\_GetPOSTNET

### Description

Gets the option of POSTNET Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_GetPOSTNET(PPOSTNET_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

IMAGER\_SetPOSTNET

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetPOSTNET(out POSTNET\_PARAMS pPostNet)

### Example

```
PPOSTNET_PARAMS pPostnet = new POSTNET_PARAMS();
if(IMAGER_GetPOSTNET(pPostnet) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetPOSTNET()", NULL, MB_TOPMOST);
m_bEnable = pPostnet->bEnable;
m_bXCD = pPostnet->bXCD;
```

delete pPostnet;

#### 4.2.40 IMAGER\_GetQR

##### Description

Gets the option of QR Barcode

##### Syntax

```
IMAGER_API BOOL IMAGER_GetQR(PQR_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

IMAGER\_SetQR

##### For .Net

Namespace : ImagerNet.Imager

Function : bool GetQR(out QR\_PARAMS pQr)

##### Example

```
PQR_PARAMS      pQr = new QR_PARAMS();  
if(!IMAGER_GetQR(pQr) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_GetQR()", NULL, MB_TOPMOST);  
m_bEnable = pQr->bEnable;  
m_nMinLen = pQr->nMinLen;  
m_nMaxLen = pQr->nMaxLen;  
delete pQr;
```

#### 4.2.41 IMAGER\_GetRSS

##### Description

Gets the option of RSS Barcode.

##### Syntax

```
IMAGER_API BOOL IMAGER_GetRSS(PRSS_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**

IMAGER\_SetRSS

### **For .Net**

Namespace : ImagerNet.Imager

Function : bool GetRSS(out RSS\_PARAMS pRss)

### **Example**

```
PRSS_PARAMS pRss = new RSS_PARAMS();  
if(IMAGER_GetRSS(pRss) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_GetRSS()", NULL, MB_TOPMOST);  
m_bEnable = pRss->bEnable;  
m_bRssLim = pRss->bRssLim;  
m_bRssExp = pRss->bRssExp;  
m_nMinLen = pRss->nMinLen;  
m_nMaxLen = pRss->nMaxLen;  
delete pRss;
```

## **4.2.42 IMAGER\_GetScanData**

### **Description**

Gets the ScanData which is read by imager.

### **Syntax**

IMAGER\_API BOOL IMAGER\_GetScanData(TCHAR \*pszBarType, TCHAR \*pszBarData);

### **Parameters**

*pszBarType*

Pointer to barcode type.

*pszBarData*

Pointer to barcode data

### **Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : ImagerNet.Imager

Function : bool GetScanData(StringBuilder szBarType, StringBuilder szBarData)

**Example**

```
TCHAR  szBarType[1024] = {0, };
```

```
TCHAR  szBarData[1024] = {0, };
```

```
IMAGER_GetScanData(szBarType, szBarData);
```

## 4.2.43 IMAGER\_GetSTRT25

**Description**

Gets the option of STRT25 Barcode.

**Syntax**

```
IMAGER_API BOOL IMAGER_GetSTRT25(PSTRT25_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**

IMAGER\_SetSTRT25

**For .Net**

Namespace : ImagerNet.Imager

Function : bool GetSTRT25(out STRT25\_PARAMS pStrt25)

**Example**

```
PSTRT25_PARAMS  pStrt25 = new STRT25_PARAMS();
```

```
if(IMAGER_GetSTRT25(pStrt25) == FALSE)
```

```
    ::MessageBox(NULL, L"Error : IMAGER_GetSTRT25()", NULL, MB_TOPMOST);
```

```
m_bEnable = pStrt25->bEnable;
```

```
m_nMinLen = pStrt25->nMinLen;  
m_nMaxLen = pStrt25->nMaxLen;  
delete pStrt25;
```

#### 4.2.44 IMAGER\_GetSymbology

##### Description

Gets Enable/Disable of Symbologies.

##### Syntax

```
IMAGER_API BOOL IMAGER_GetSymbology(PDECODER 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

IMAGER\_SetSymbology

##### For .Net

Namespace : ImagerNet.Imager

Function : bool GetSymbology(out DECODER pSymbology)

##### Example

```
PDECODER pSym = new DECODER();  
BOOL bEnable[47] = {0, };  
int i = 0;  
if(IMAGER_GetSymbology(pSym) == FALSE)  
    ::MessageBox(NULL, L"ERROR : IMAGER_GetSymbology()", NULL, MB_TOPMOST);  
bEnable[0] = pSym->bAZTEC;  
bEnable[1] = pSym->bCODABAR;  
bEnable[2] = pSym->bCODE11;  
bEnable[3] = pSym->bCODE128;  
bEnable[4] = pSym->bCODE39;  
bEnable[5] = pSym->bCODE49;  
bEnable[6] = pSym->bCODE93;
```

bEnable[7] = pSym->bCOMPOSITE;  
bEnable[8] = pSym->bDATAMATRIX;  
bEnable[9] = pSym->bEAN8;  
bEnable[10] = pSym->bEAN13;  
bEnable[11] = pSym->bINT25;  
bEnable[12] = pSym->bMAXICODE;  
bEnable[13] = pSym->bMICROPDF;  
bEnable[14] = pSym->bOCR;  
bEnable[15] = pSym->bPDF417;  
bEnable[16] = pSym->bPOSTNET;  
bEnable[17] = pSym->bQR;  
bEnable[18] = pSym->bRSS;  
bEnable[19] = pSym->bUPCA;  
bEnable[20] = pSym->bUPCE;  
bEnable[21] = pSym->bISBT;  
bEnable[22] = pSym->bBPO;  
bEnable[23] = pSym->bCANPOST;  
bEnable[24] = pSym->bAUSPOST;  
bEnable[25] = pSym->bIATA25;  
bEnable[26] = pSym->bCODABLOCK;  
bEnable[27] = pSym->bJAPOST;  
bEnable[28] = pSym->bPLANET;  
bEnable[29] = pSym->bDUTCHPOST;  
bEnable[30] = pSym->bMSI;  
bEnable[31] = pSym->bTLCODE39;  
bEnable[32] = pSym->bSTRT25;  
bEnable[33] = pSym->bMATRIX25;  
bEnable[34] = pSym->bPLESSEY;  
bEnable[35] = pSym->bCHINAPOST;  
bEnable[36] = pSym->bKOREAPOST;  
bEnable[37] = pSym->bTELEPEN;  
bEnable[38] = pSym->bCODE16K;  
bEnable[39] = pSym->bPOSI CODE;  
bEnable[40] = pSym->bCOUPONCODE;

```

bEnable[41] = pSym->bUSPS4CB;
bEnable[42] = pSym->bIDTAG;
bEnable[43] = pSym->bLABEL;
bEnable[44] = pSym->bGS1_128;
bEnable[45] = pSym->bHANXIN;
bEnable[46] = pSym->bGRIDMATRIX;
for(i=0; i<47; i++)
{
    m_ctrlListSymbology.SetCheck(i, bEnable[i]);
}
delete pSym;

```

#### 4.2.45 IMAGER\_GetTELEPEN

##### Description

Gets the option of TELEPEN Barcode.

##### Syntax

```
IMAGER_API BOOL IMAGER_GetTELEPEN(PTELEPEN_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

IMAGER\_SetTELEPEN

##### For .Net

Namespace : ImagerNet.Imager

Function : bool GetTELEPEN(out TELEPEN\_PARAMS pTelepen)

##### Example

```

PTELEPEN_PARAMS pTelepen = new TELEPEN_PARAMS();
if(IMAGER_GetTELEPEN(pTelepen) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetTELEPEN()", NULL, MB_TOPMOST);
m_bEnable = pTelepen->bEnable;

```



```
m_bNumeric = pTelepen->bNumeric;  
m_nMinLen = pTelepen->nMinLen;  
m_nMaxLen = pTelepen->nMaxLen;  
delete pTelepen;
```

#### 4.2.46 IMAGER\_GetUPCA

##### Description

Gets the option of UPC-A Barcode.

##### Syntax

```
IMAGER_API BOOL IMAGER_GetUPCA(PUPCA_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

IMAGER\_SetUPCA

##### For .Net

Namespace : ImagerNet.Imager

Function : bool GetUPCA(out UPCA\_PARAMS pUpca)

##### Example

```
PUPCA_PARAMS pUpca = new UPCA_PARAMS();  
if(IMAGER_GetUPCA(pUpca) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_GetUPCA()", NULL, MB_TOPMOST);  
m_bEnable = pUpca->bEnable;  
m_bXCD = pUpca->bXCD;  
m_bXNum = pUpca->bXNum;  
m_bAddOn = pUpca->bAddOn;  
delete pUpca;
```

#### 4.2.47 IMAGER\_GetUPCE

##### Description

Gets the option of UPC-E Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_GetUPCE(PUPCE_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

IMAGER\_SetUPCE

### For .Net

Namespace : ImagerNet.Imager

Function : bool GetUPCE(out UPCE\_PARAMS pUpce)

### Example

```
PUPCE_PARAMS pUpce = new UPCE_PARAMS();
if(IMAGER_GetUPCE(pUpce) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetUPCE()", NULL, MB_TOPMOST);
m_bEnable = pUpce->bEnable;
m_bXCD = pUpce->bXCD;
m_bXNum = pUpce->bXNum;
m_bAddOn = pUpce->bAddOn;
delete pUpce;
```

## 4.2.48 IMAGER\_GetVersionInfo

### Description

Gets the information of imager driver and dll version.

### Syntax

```
IMAGER_API BOOL IMAGER_GetVersionInfo(TCHAR* pszVersion);
```

### 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 .Net

Namespace : ImagerNet.Imager

Function : string GetVersionInfo()

#### Example

```
TCHAR szVersionInfo[1024] = {0, };
```

```
IMAGER_GetVersionInfo(szVersionInfo);
```

### 4.2.49 IMAGER\_IQGetBarcodeData

#### Description

Gets the ScanData which is read by imager.

#### Syntax

```
IMAGER_API BOOL IMAGER_IQGetBarcodeData(TCHAR *pszBarData);
```

#### Parameters

*pszBarData*

Pointer to barcode data.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

None

#### For .Net

Namespace : ImagerNet.Imager

Function : bool IQGetBarcodeData(StringBuilder szBarData)

#### Example

```
TCHAR szBarData[1024] = {0, };
```

```
IMAGER_IQGetBarcodeData(szBarData);
```

## 4.2.50 IMAGER\_IQGetOption

### Description

Gets the option of IQ Imaging

### Syntax

```
IMAGER_API BOOL IMAGER_IQGetOption(PIQ_PARAMS plQOption);
```

### Parameters

*plQOption*

Pointer to an IQ\_PARAMS structure to be filled in with the IQ Imaging parameters.

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

None

### For .Net

Namespace : ImagerNet.Imager

Function : bool IQGetOption(out IQ\_PARAMS plQOption)

### Example

```
PIQ_PARAMS plQOption = new IQ_PARAMS();
if(!IMAGER_IQGetOption(plQOption) == FALSE)
{
    ::MessageBox(NULL, L"Error : IMAGER_IQGetOption()", NULL, MB_TOPMOST);
    return FALSE;
}
m_nIQType = plQOption->nIQType;
m_strIqSaveFolder = plQOption->szSaveFolder;
m_strIqFileName = plQOption->szFileName;
m_strSaveFormat = L".BMP";
m_ctrlComboSaveMode.SetCurSel(plQOption->nSaveMode);
delete plQOption;
```

## 4.2.51 IMAGER\_IQImagingStart

### Description

Starts IQ Imaging.

**Syntax**

IMAGER\_API BOOL IMAGER\_IQImagingStart();

**Parameters**

None

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

IMAGER\_IQImagingStop

**For .Net**

Namespace : ImagerNet.Imager

Function : bool IQImagingStart()

**Example**

None

## 4.2.52 IMAGER\_IQImagingStop

**Description**

Stops IQ Imaging.

**Syntax**

IMAGER\_API BOOL IMAGER\_IQImagingStop();

**Parameters**

None

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

IMAGER\_IQImagingStart

**For .Net**

Namespace : ImagerNet.Imager

Function : bool IQImagingStop()

**Example**

None

### 4.2.53 IMAGER\_IQInit

#### Description

Initializes IQ imaging

#### Syntax

```
IMAGER_API BOOL IMAGER_IQInit(HWND hPictureWnd);
```

#### Parameters

*hPictureWnd*

Window that will show preview.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

IMAGER\_IQUnInit

#### For .Net

Namespace : ImagerNet.Imager

Function : bool IQInit(IntPtr hPictureWnd);

#### Example

None

### 4.2.54 IMAGER\_IQSetOption

#### Description

Sets the option of IQ Imaging.

#### Syntax

```
IMAGER_API BOOL IMAGER_IQSetOption(PIQ_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

IMAGER\_IQGetOption

### For .Net

Namespace : ImagerNet.Imager

Function : bool IQSetOption(ref IQ\_PARAMS plQOption)

### Example

```
PIQ_PARAMS plQOption = new IQ_PARAMS();
plQOption->nSaveMode = m_ctrlComboSaveMode.GetCurSel();
plQOption->nIQType = m_nIQType;
wsprintf(plQOption->szSaveFolder, L"%s", m_strlqSaveFolder);
wsprintf(plQOption->szFileName, L"%s", m_strlqFileName);
if(IMAGER_IQSetOption(plQOption) == FALSE)
{
    ::MessageBox(NULL, L"Error : IMAGER_IQSetOption()", NULL, MB_TOPMOST);
    return FALSE;
}
delete plQOption;
```

## 4.2.55 IMAGER\_IQUnInit

### Description

Uninitializes IQ imaging.

### Syntax

IMAGER\_API BOOL IMAGER\_IQUnInit();

### Parameters

None

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

IMAGER\_IQInit

### For .Net

Namespace : ImagerNet.Imager

Function : public bool IQUnInit()

### Example

None

## 4.2.56 IMAGER\_Open

### Description

Opens an imager.

### Syntax

```
IMAGER_API BOOL IMAGER_Open();
```

### Parameters

None

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

IMAGER\_Close

### For .Net

Namespace : ImagerNet.Imager

Function : bool Open()

### Example

None

## 4.2.57 IMAGER\_Read

### Description

Starts the beaming of imager

### Syntax

```
IMAGER_API BOOL IMAGER_Read();
```

### Parameters

None

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

IMAGER\_ReadCancel

### For .Net

Namespace : ImagerNet.Imager



Function : bool Read()

### Example

None

## 4.2.58 IMAGER\_ReadCancel

### Description

Stops the beaming of imager

### Syntax

```
IMAGER_API BOOL IMAGER_ReadCancel();
```

### Parameters

None

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

IMAGER\_Read

### For .Net

Namespace : ImagerNet.Imager

Function : bool ReadCancel()

### Example

None

## 4.2.59 IMAGER\_SetAZTEC

### Description

Sets the option of AZTEC Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetAZTEC(PAZTEC_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

IMAGER\_GetAZTEC

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetAZTEC(ref AZTEC\_PARAMS pAztec)

### Example

```
PAZTEC_PARAMS pAztec = new AZTEC_PARAMS();  
pAztec->bEnable = m_bEnable;  
pAztec->nMinLen = m_nMinLen;  
pAztec->nMaxLen = m_nMaxLen;  
if(IMAGER_SetAZTEC(pAztec) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_ SetAZTEC ()", NULL, MB_TOPMOST);  
delete pAztec;
```

## 4.2.60 IMAGER\_SetCenteringWindow

### Description

Enables centering window function.

### Syntax

IMAGER\_API BOOL IMAGER\_SetCenteringWindow(RECT\* pRect);

### Parameters

*pRect*

Defines the region of the image.

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

IMAGER\_GetCenteringWindow

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetCenteringWindow(ref RECT\_PARAM pRect)

### Example

```
RECT rect;
```

```

rect.top = m_nEditTop;
rect.bottom = m_nEditBottom;
rect.left = m_nEditLeft;
rect.right = m_nEditRight;
if(!IMAGER_SetCenteringWindow(&rect) == FALSE)
{
    ::MessageBox(NULL, L"ERROR : IMAGER_SetCenteringWindow()", NULL, MB_TOPMOST);
    return FALSE;
}

```

## 4.2.61 IMAGER\_SetCHINAPOST

### Description

Sets the option of CHINAPOST Barcode

### Syntax

```
IMAGER_API BOOL IMAGER_SetCHINAPOST(PCHINAPOST_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

IMAGER\_GetCHINAPOST

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetCHINAPOST(ref CHINAPOST\_PARAMS pChinaPost);

### Example

```

PCHINAPOST_PARAMS pChinapost = new CHINAPOST_PARAMS();
pChinapost->bEnable = m_bEnable;
pChinapost->nMinLen = m_nMinLen;
pChinapost->nMaxLen = m_nMaxLen;
if(!IMAGER_SetCHINAPOST(pChinapost) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_SetCHINAPOST()", NULL, MB_TOPMOST);

```

delete pChinapost;

## 4.2.62 IMAGER\_SetCODABAR

### Description

Sets the option of CODABAR Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetCODABAR(PCODABAR_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

IMAGER\_GetCODABAR

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetCODABAR(ref CODABAR\_PARAMS pCodabar)

### Example

```
PCODABAR_PARAMS  pCodabar = new CODABAR_PARAMS();
pCodabar->bEnable = m_bEnable;
pCodabar->bCDV = m_bCDV;
pCodabar->bXCD = m_bXCD;
pCodabar->bXSS = m_bXSS;
pCodabar->nMinLen = m_nMinLen;
pCodabar->nMaxLen = m_nMaxLen;
if(IMAGER_SetCODABAR(pCodabar) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_SetCODABAR()", NULL, MB_TOPMOST);
delete pCodabar;
```

## 4.2.63 IMAGER\_SetCODABLOCK

### Description

Sets the option of CODABLOCK Barcode.

**Syntax**

```
IMAGER_API BOOL IMAGER_SetCODABLOCK(PCODABLOCK_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**

IMAGER\_GetCODABLOCK

**For .Net**

Namespace : ImagerNet.Imager

Function : bool SetCODABLOCK(ref CODABLOCK\_PARAMS pCodablock)

**Example**

```
PCODABLOCK_PARAMS pCodablock = new CODABLOCK_PARAMS();  
pChinapost->bEnable = m_bEnable;  
pChinapost->nMinLen = m_nMinLen;  
pChinapost->nMaxLen = m_nMaxLen;  
if(IMAGER_SetCODABLOCK(pCodablock) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_SetCODABLOCK()", NULL, MB_TOPMOST);  
delete pChinapost;
```

## 4.2.64 IMAGER\_SetCODE11

**Description**

Sets the option of CODE11 Barcode.

**Syntax**

```
IMAGER_API BOOL IMAGER_SetCODE11(PCODE11_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

IMAGER\_GetCODE11

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetCODE11(ref CODE11\_PARAMS pCode11)

### Example

```
PCODE11_PARAMS pCode11 = new CODE11_PARAMS();  
pCode11->bEnable = m_bEnable;  
pCode11->bCDV = m_bCDV;  
pCode11->nMinLen = m_nMinLen;  
pCode11->nMaxLen = m_nMaxLen;  
if(IMAGER_SetCODE11(pCode11) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetCODE11()", NULL, MB_TOPMOST);  
delete pCode11;
```

## 4.2.65 IMAGER\_SetCODE128

### Description

Sets the option of CODE128 Barcode.

### Syntax

IMAGER\_API BOOL IMAGER\_SetCODE128(PCODE128\_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

IMAGER\_GetCODE128

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetCODE128(ref CODE128\_PARAMS pCode128)

### Example

```
PCODE128_PARAMS pCode128 = new CODE128_PARAMS();
pCode128->bEnable = m_bEnable;
pCode128->nMinLen = m_nMinLen;
pCode128->nMaxLen = m_nMaxLen;
if(!IMAGER_SetCODE128(pCode128) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCODE128()", NULL, MB_TOPMOST);
delete pCode128;
```

## 4.2.66 IMAGER\_SetCODE16K

### Description

Sets the option of CODE16K Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetCODE16K(PCODE16K_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

IMAGER\_GetCODE16K

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetCODE16K(ref CODE16K\_PARAMS pCode16k);

### Example

```
PCODE16K_PARAMS pCode16k = new CODE16K_PARAMS();
pCode16k->bEnable = m_bEnable;
pCode16k->nMinLen = m_nMinLen;
pCode16k->nMaxLen = m_nMaxLen;
if(!IMAGER_SetCODE16K(pCode16k) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_SetCODE16K()", NULL, MB_TOPMOST);
delete pCode16k;
```

## 4.2.67 IMAGER\_SetCODE39

### Description

Sets the option of CODE39 Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetCODE39(PCODE39_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

IMAGER\_GetCODE39

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetCODE39(ref CODE39\_PARAMS pCode39);

### Example

```
PCODE39_PARAMS pCode39 = new CODE39_PARAMS();
pCode39->bEnable = m_bEnable;
pCode39->nFormat = m_nFormat;
pCode39->bCDV = m_bCDV;
pCode39->bXCD = m_bXCD;
pCode39->bFullASCII = m_bFullASCII;
pCode39->nMinLen = m_nMinLen;
pCode39->nMaxLen = m_nMaxLen;
if(IMAGER_SetCODE39(pCode39) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetCODE39()", NULL, MB_TOPMOST);
delete pCode39;
```

## 4.2.68 IMAGER\_SetCODE49

### Description

Sets the option of CODE49 Barcode.

### Syntax



IMAGER\_API BOOL IMAGER\_SetCODE49(PCODE49\_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

IMAGER\_GetCODE49

#### For .Net

Namespace : ImagerNet.Imager

Function : bool SetCODE49(ref CODE49\_PARAMS pCode49)

#### Example

```
PCODE49_PARAMS pCode49 = new CODE49_PARAMS();
pCode49->bEnable = m_bEnable;
pCode49->nMinLen = m_nMinLen;
pCode49->nMaxLen = m_nMaxLen;
if(IMAGER_SetCODE49(pCode49) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCODE49()", NULL, MB_TOPMOST);
delete pCode49;
```

### 4.2.69 IMAGER\_SetCODE93

#### Description

Sets the option of CODE93 Barcode.

#### Syntax

IMAGER\_API BOOL IMAGER\_SetCODE93(PCODE93\_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

IMAGER\_GetCODE93

## For .Net

Namespace : ImagerNet.Imager

Function : bool SetCODE93(ref CODE93\_PARAMS pCode93);

## Example

```
PCODE93_PARAMS pCode93 = new CODE93_PARAMS();
pCode93->bEnable = m_bEnable;
pCode93->nMinLen = m_nMinLen;
pCode93->nMaxLen = m_nMaxLen;
if(IMAGER_SetCODE93(pCode93) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCODE93()", NULL, MB_TOPMOST);
delete pCode93;
```

## 4.2.70 IMAGER\_SetCOMPOSITE

### Description

Sets the option of COMPOSITE Barcode.

### Syntax

IMAGER\_API BOOL IMAGER\_SetCOMPOSITE(PCOMPOSITE\_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

IMAGER\_GetCOMPOSITE

## For .Net

Namespace : ImagerNet.Imager

Function : bool SetCOMPOSITE(ref COMPOSITE\_PARAMS pComposite)

## Example

```
PCOMPOSITE_PARAMS pComposite = new COMPOSITE_PARAMS();
pComposite->bEnable = m_bEnable;
```

```

pComposite->bComposite_Upc = m_bComposite_Upc;
pComposite->nMinLen = m_nMinLen;
pComposite->nMaxLen = m_nMaxLen;
if(IMAGER_SetCOMPOSITE(pComposite) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetCOMPOSITE()", NULL, MB_TOPMOST);
delete pComposite;

```

## 4.2.71 IMAGER\_SetDATAMATRIX

### Description

Sets the option of DATAMATRIX Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetDATAMATRIX(PDATAMATRIX_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

IMAGER\_GetDATAMATRIX

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetDATAMATRIX(ref DATAMATRIX\_PARAMS pDataMatrix)

### Example

```

PDATAMATRIX_PARAMS  pDataMatrix = new DATAMATRIX_PARAMS();
pDataMatrix->bEnable = m_bEnable;
pDataMatrix->nMinLen = m_nMinLen;
pDataMatrix->nMaxLen = m_nMaxLen;
if(IMAGER_SetCOMPOSITE(pComposite) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_GetCOMPOSITE()", NULL, MB_TOPMOST);
delete pDataMatrix;

```

## 4.2.72 IMAGER\_SetDecOption

### Description

Sets the option of Decoder.

### Syntax

```
IMAGER_API BOOL IMAGER_SetDecOption(PDECOPTION_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

IMAGER\_GetDecOption

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetDecOption(ref DECOPTION\_PARAMS pDecOption)

### Example

```
PDECOPTION_PARAMS pDecOption = new DECOPTION_PARAMS();
pDecOption->nDecodeMode = m_ctrlComboDecodeMode.GetCurSel();
pDecOption->nLinearRange = m_nLinearRange;
pDecOption->nPrintWeight = m_nPrintWeight;
pDecOption->nMaxDecode = m_nMaxDecode;
pDecOption->nMaxSearch = m_nMaxSearch;
pDecOption->bVideoReverse = m_bVideoReverse;
if(IMAGER_SetDecOption(pDecOption) == FALSE)
{
    ::MessageBox(NULL, L"ERROR : IMAGER_SetDecOption()", NULL, MB_TOPMOST);
    return FALSE;
}
delete pDecOption;
```

## 4.2.73 IMAGER\_SetEAN13

### Description

Sets the option of EAN-13 Barcode.

#### **Syntax**

```
IMAGER_API BOOL IMAGER_SetEAN13(PEAN13_PARAMS pEan13);
```

#### **Parameters**

*pEan13*

Pointer to an EAN13\_PARAMS structure holding the EAN-13 common parameters.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

IMAGER\_GetEAN13

#### **For .Net**

Namespace : ImagerNet.Imager

Function : bool SetEAN13(ref EAN13\_PARAMS pEan13)

#### **Example**

```
PEAN13_PARAMS pEan13 = new EAN13_PARAMS();  
pEan13->bEnable = m_bEnable;  
pEan13->bXCD = m_bXCD;  
pEan13->bAddOn = m_bAddOn;  
if(IMAGER_SetEAN13(pEan13) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetEAN13()", NULL, MB_TOPMOST);  
delete pEan13;
```

### **4.2.74 IMAGER\_SetEAN8**

#### **Description**

Sets the option of EAN-8 Barcode.

#### **Syntax**

```
IMAGER_API BOOL IMAGER_SetEAN8(PEAN8_PARAMS pEan8);
```

#### **Parameters**

*pEan8*

Pointer to an EAN8\_PARAMS structure holding the EAN-8 common parameters.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

### See Also

IMAGER\_GetEAN8

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetEAN8(ref EAN8\_PARAMS pEan8)

### Example

```
PEAN8_PARAMS pEan8 = new EAN8_PARAMS();
pEan8->bEnable = m_bEnable;
pEan8->bXCD = m_bXCD;
pEan8->bAddOn = m_bAddOn;
if(IMAGER_SetEAN8(pEan8) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetEAN8()", NULL, MB_TOPMOST);
delete pEan8;
```

## 4.2.75 IMAGER\_SetIATA25

### Description

Sets the option of IATA25 Barcode.

### Syntax

IMAGER\_API BOOL IMAGER\_SetIATA25(PIATA25\_PARAMS plata25);

### Parameters

*plata25*

Pointer to an IATA25\_PARAMS structure holding the IATA25 common parameters.

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

IMAGER\_GetIATA25

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetIATA25(ref IATA25\_PARAMS plata25)

### Example

```
PIATA25_PARAMS plata25 = new IATA25_PARAMS();
```

```

plata25->bEnable = m_bEnable;
plata25->nMinLen = m_nMinLen;
plata25->nMaxLen = m_nMaxLen;
if(IMAGER_SetIATA25(plata25) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_SetIATA25()", NULL, MB_TOPMOST);
delete plata25;

```

#### 4.2.76 IMAGER\_SetINT25

##### Description

Sets the option of INT25 Barcode.

##### Syntax

```
IMAGER_API BOOL IMAGER_SetINT25(PINT25_PARAMS pInt25);
```

##### Parameters

*pInt25*

Pointer to an INT25\_PARAMS structure holding the INT25 common parameters.

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

IMAGER\_GetINT25

##### For .Net

Namespace : ImagerNet.Imager

Function : bool SetINT25(ref INT25\_PARAMS pInt25)

##### Example

```

PINT25_PARAMS pInt25 = new INT25_PARAMS();
pInt25->bEnable = m_bEnable;
pInt25->bCDV = m_bCDV;
pInt25->bXCD = m_bXCD;
pInt25->nMinLen = m_nMinLen;
pInt25->nMaxLen = m_nMaxLen;
if(IMAGER_SetINT25(pInt25) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetINT25()", NULL, MB_TOPMOST);
delete pInt25;

```

## 4.2.77 IMAGER\_SetKOREAPOST

### Description

Sets the option of KOREAPOST Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetKOREAPOST(PKOREAPOST_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

IMAGER\_GetKOREAPOST

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetKOREAPOST(ref KOREAPOST\_PARAMS pKoreaPost)

### Example

```
PKOREAPOST_PARAMS pKoreapost = new KOREAPOST_PARAMS();  
pKoreapost->bEnable = m_bEnable;  
pKoreapost->nMinLen = m_nMinLen;  
pKoreapost->nMaxLen = m_nMaxLen;  
if(IMAGER_SetKOREAPOST(pKoreapost) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_SetKOREAPOST()", NULL, MB_TOPMOST);  
delete pKoreapost;
```

## 4.2.78 IMAGER\_SetMATRIX25

### Description

Sets the option of MATRIX25 Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetMATRIX25(PMATRIX25_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**

IMAGER\_GetMATRIX25

#### **For .Net**

Namespace : ImagerNet.Imager

Function : bool SetMATRIX25(ref MATRIX25\_PARAMS pMatrix25)

#### **Example**

```
PMATRIX25_PARAMS pMatrix25 = new MATRIX25_PARAMS();  
pMatrix25->bEnable = m_bEnable;  
pMatrix25->nMinLen = m_nMinLen;  
pMatrix25->nMaxLen = m_nMaxLen;  
if (IMAGER_SetMATRIX25(pMatrix25) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_SetMATRIX25()", NULL, MB_TOPMOST);  
delete pMatrix25;
```

### **4.2.79 IMAGER\_SetMAXICODE**

#### **Description**

Sets the option of MAXICODE Barcode.

#### **Syntax**

IMAGER\_API BOOL IMAGER\_SetMAXICODE(PMAXICODE\_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**

IMAGER\_GetMAXICODE

#### **For .Net**

Namespace : ImagerNet.Imager

Function : bool SetMAXICODE(ref MAXICODE\_PARAMS pMaxiCode)

#### Example

```
PMAXICODE_PARAMS pMaxicode= new MAXICODE_PARAMS();  
pMaxicode ->bEnable = m_bEnable;  
pMaxicode ->nMinLen = m_nMinLen;  
pMaxicode ->nMaxLen = m_nMaxLen;  
if(IMAGER_SetMAXICODE(pMaxicode) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_SetMAXICODE()", NULL, MB_TOPMOST);  
delete pMaxicode;
```

### 4.2.80 IMAGER\_SetMICROPDF

#### Description

Sets the option of MICROPDF Barcode.

#### Syntax

IMAGER\_API BOOL IMAGER\_SetMICROPDF(PMICROPDF\_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

IMAGER\_GetMICROPDF

#### For .Net

Namespace : ImagerNet.Imager

Function : bool SetMICROPDF(ref MICROPDF\_PARAMS pMicroPdf)

#### Example

```
PMICROPDF_PARAMS pMicropdf = new MICROPDF_PARAMS();  
pMicropdf->bEnable = m_bEnable;  
pMicropdf->nMinLen = m_nMinLen;  
pMicropdf->nMaxLen = m_nMaxLen;  
if(IMAGER_SetMICROPDF(pMicropdf) == FALSE)
```

```
::MessageBox(NULL, L"Error : IMAGER_SetMICROPDF()", NULL, MB_TOPMOST);  
delete pMicropdf;
```

#### 4.2.81 IMAGER\_SetMSI

##### Description

Sets the option of MSI Barcode.

##### Syntax

```
IMAGER_API BOOL IMAGER_SetMSI(PMSI_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

IMAGER\_GetMSI

##### For .Net

Namespace : ImagerNet.Imager

Function : bool SetMSI(ref MSI\_PARAMS pMsi)

##### Example

```
PMSI_PARAMS pMsi = new MSI_PARAMS();  
pMsi->bEnable = m_bEnable;  
pMsi->bXCD = m_bXCD;  
pMsi->nMinLen = m_nMinLen;  
pMsi->nMaxLen = m_nMaxLen;  
if(IMAGER_SetMSI(pMsi) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetMSI()", NULL, MB_TOPMOST);  
delete pMsi;
```

#### 4.2.82 IMAGER\_SetOCR

##### Description

Sets the option of OCR Barcode.

##### Syntax

IMAGER\_API BOOL IMAGER\_SetOCR(POCR\_PARAMS pOcr);

#### Parameters

*pOcr*

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

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

IMAGER\_GetOCR

#### For .Net

Namespace : ImagerNet.Imager

Function : bool SetOCR(ref OCR\_PARAMS pOcr)

#### Example

```
POCR_PARAMS pOcr = new OCR_PARAMS();
pOcr->bEnable = m_bEnable;
pOcr->nMode = (OCR_MODE)m_ctrlComboMode.GetCurSel();
wsprintf(pOcr->szTemplate, L"%s", m_strEditTemplate);
wsprintf(pOcr->szGroupG, L"%s", m_strEditGroupG);
wsprintf(pOcr->szGroupH, L"%s", m_strEditGroupH);
wsprintf(pOcr->szCheckChar, L"%s", m_strCheckChar);
if(IMAGER_SetOCR(pOcr) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_SetOcr()", NULL, MB_TOPMOST);
delete pOcr;
```

### 4.2.83 IMAGER\_SetOption

#### Description

Sets the option of imager.

#### Syntax

IMAGER\_API BOOL IMAGER\_SetOption(PDECODER\_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

IMAGER\_GetOption

#### For .Net

Namespace : ImagerNet.Imager

Function : bool SetOption(ref DECODER\_PARAMS pOption)

#### Example

```
PDECODER_PARAMS pOption = new DECODER_PARAMS();
pOption->nSound = m_nSound;
pOption->bCentering = m_bCentering;
pOption->bVibrate = m_bVibrate;
pOption->bXmitAimID = m_bAimID;
pOption->bContinueMode = m_bContinueMode;
pOption->bHexMode = m_bHexMode;
pOption->nTimeOut = m_ctrlComboTimeOut.GetCurSel() + 1;
pOption->nLightMode = m_ctrlComboLightMode.GetCurSel();
if(IMAGER_SetOption(pOption) == FALSE)
{
    ::MessageBox(NULL, L"ERROR : IMAGER_SetOption()", NULL, MB_TOPMOST);
    return FALSE;
}
delete pOption;
```

### 4.2.84 IMAGER\_SetPDF417

#### Description

Sets the option of PDF417 Barcode.

#### Syntax

IMAGER\_API BOOL IMAGER\_SetPDF417(PPDF417\_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

IMAGER\_GetPDF417

#### For .Net

Namespace : ImagerNet.Imager

Function : bool SetPDF417(ref PDF417\_PARAMS pPdf417)

#### Example

```
PPDF417_PARAMS      pPdf417 = new PDF417_PARAMS();
pPdf417->bEnable = m_bEnable;
pPdf417->nMinLen = m_nMinLen;
pPdf417->nMaxLen = m_nMaxLen;
if(IMAGER_SetPDF417(pPdf417) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_SetPDF417()", NULL, MB_TOPMOST);
delete pPdf417;
```

## 4.2.85 IMAGER\_SetPLANET

#### Description

Sets the option of PLANET Barcode.

#### Syntax

IMAGER\_API BOOL IMAGER\_SetPLANET(PPLANET\_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

IMAGER\_GetPLANET

#### For .Net

Namespace : ImagerNet.Imager

Function : bool SetPLANET(ref PLANET\_PARAMS pPlanet)

### Example

```
PPLANET_PARAMS pPlanet = new PLANET_PARAMS();  
pPlanet->bEnable = m_bEnable;  
pPlanet->bXCD = m_bXCD;  
if(IMAGER_SetPLANET(pPlanet) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetPLANET()", NULL, MB_TOPMOST);  
delete pPlanet;
```

## 4.2.86 IMAGER\_SetPLESSEY

### Description

Sets the option of PLESSEY Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetPLESSEY(PPLESSEY_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

IMAGER\_GetPLESSEY

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetPLESSEY(ref PLESSEY\_PARAMS pPlessey)

### Example

```
PPLESSEY_PARAMS pPlessey = new PLESSEY_PARAMS();  
pPlessey->bEnable = m_bEnable;  
pPlessey->nMinLen = m_nMinLen;  
pPlessey->nMaxLen = m_nMaxLen;  
if(IMAGER_SetPLESSEY(pPlessey) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_SetPLESSEY()", NULL, MB_TOPMOST);  
delete pPlessey;
```

## 4.2.87 IMAGER\_SetPOSICODE

### Description

Sets the option of POSICODE Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetPOSICODE(PPOSICODE_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

IMAGER\_GetPOSICODE

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetPOSICODE(ref POSICODE\_PARAMS pPosiCode)

### Example

```
PPOSICODE_PARAMS pPosicode = new POSICODE_PARAMS();
pPosicode->bEnable = m_bEnable;
pPosicode->bPosi_Lim1 = m_bPosi_Lim1;
pPosicode->bPosi_Lim2 = m_bPosi_Lim2;
pPosicode->nMinLen = m_nMinLen;
pPosicode->nMaxLen = m_nMaxLen;
if(IMAGER_SetPOSICODE(pPosicode) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetPLANET()", NULL, MB_TOPMOST);
delete pPosicode;
```

## 4.2.88 IMAGER\_SetPOSTNET

### Description

Sets the option of POSTNET Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetPOSTNET(PPOSTNET_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**

IMAGER\_GetPOSTNET

#### **For .Net**

Namespace : ImagerNet.Imager

Function : bool SetPOSTNET(ref POSTNET\_PARAMS pPostNet)

#### **Example**

```
PPOSTNET_PARAMS pPostnet = new POSTNET_PARAMS();  
pPostnet->bEnable = m_bEnable;  
pPostnet->bXCD = m_bXCD;  
if(IMAGER_SetPOSTNET(pPostnet) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetPOSTNET()", NULL, MB_TOPMOST);  
delete pPostnet;
```

### **4.2.89 IMAGER\_SetQR**

#### **Description**

Sets the option of QR Barcode.

#### **Syntax**

IMAGER\_API BOOL IMAGER\_SetQR(PQR\_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**

IMAGER\_GetQR

#### **For .Net**

Namespace : ImagerNet.Imager

Function : bool SetQR(ref QR\_PARAMS pQr)

#### Example

```
PQR_PARAMS pQr = new QR_PARAMS();  
pQr->bEnable = m_bEnable;  
pQr->nMinLen = m_nMinLen;  
pQr->nMaxLen = m_nMaxLen;  
if(IMAGER_SetQR(pQr) == FALSE)  
    ::MessageBox(NULL, L"Error : IMAGER_SetQR()", NULL, MB_TOPMOST);  
delete pQr;
```

## 4.2.90 IMAGER\_SetRSS

### Description

Sets the option of RSS Barcode.

### Syntax

```
IMAGER_API BOOL IMAGER_SetRSS(PRSS_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

IMAGER\_GetRSS

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetRSS(ref RSS\_PARAMS pRss)

#### Example

```
PRSS_PARAMS pRss = new RSS_PARAMS();  
pRss->bEnable = m_bEnable;  
pRss->bRssLim = m_bRssLim;  
pRss->bRssExp = m_bRssExp;  
pRss->nMinLen = m_nMinLen;
```

```

pRss->nMaxLen = m_nMaxLen;
if(IMAGER_SetRSS(pRss) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetRSS()", NULL, MB_TOPMOST);
delete pRss;

```

### 4.2.91 IMAGER\_SetSTRT25

#### Description

Sets the option of STRT25 Barcode.

#### Syntax

```
IMAGER_API BOOL IMAGER_SetSTRT25(PSTRT25_PARAMS pStrt25);
```

#### Parameters

*pStrt25*

ex

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

IMAGER\_GetSTRT25

#### For .Net

Namespace : ImagerNet.Imager

Function : bool SetSTRT25(ref STRT25\_PARAMS pStrt25)

#### Example

```

PSTRT25_PARAMS pStrt25 = new STRT25_PARAMS();
pStrt25->bEnable = m_bEnable;
pStrt25->nMinLen = m_nMinLen;
pStrt25->nMaxLen = m_nMaxLen;
if(IMAGER_SetSTRT25(pStrt25) == FALSE)
    ::MessageBox(NULL, L"Error : IMAGER_SetSTRT25()", NULL, MB_TOPMOST);
delete pStrt25;

```

### 4.2.92 IMAGER\_SetSymbology

#### Description

Sets the Enable/Disable of Symbologies.

## Syntax

```
IMAGER_API BOOL IMAGER_SetSymbology(PDECODER 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

IMAGER\_GetSymbology

## For .Net

Namespace : ImagerNet.Imager

Function : bool SetSymbology(ref DECODER pSymbology)

## Example

```
PDECODER pSym = new DECODER();
BOOL bEnable[47] = {0, };
int i = 0;
for(i=0; i<47; i++)
{
    bEnable[i] = m_ctrlListSymbology.GetCheck(i);
}
pSym->bAZTEC = bEnable[0];
pSym->bCODABAR = bEnable[1];
pSym->bCODE11 = bEnable[2];
pSym->bCODE128 = bEnable[3];
pSym->bCODE39 = bEnable[4];
pSym->bCODE49 = bEnable[5];
pSym->bCODE93 = bEnable[6];
pSym->bCOMPOSITE = bEnable[7];
pSym->bDATAMATRIX = bEnable[8];
pSym->bEAN8 = bEnable[9];
pSym->bEAN13 = bEnable[10];
pSym->bINT25 = bEnable[11];
```

pSym->bMAXICODE = bEnable[12];  
pSym->bMICROPDF = bEnable[13];  
pSym->bOCR = bEnable[14];  
pSym->bPDF417 = bEnable[15];  
pSym->bPOSTNET = bEnable[16];  
pSym->bQR = bEnable[17];  
pSym->bRSS = bEnable[18];  
pSym->bUPCA = bEnable[19];  
pSym->bUPCE = bEnable[20];  
pSym->bISBT = bEnable[21];  
pSym->bBPO = bEnable[22];  
pSym->bCANPOST = bEnable[23];  
pSym->bAUSPOST = bEnable[24];  
pSym->bIATA25 = bEnable[25];  
pSym->bCODABLOCK = bEnable[26];  
pSym->bJAPOST = bEnable[27];  
pSym->bPLANET = bEnable[28];  
pSym->bDUTCHPOST = bEnable[29];  
pSym->bMSI = bEnable[30];  
pSym->bTLCODE39 = bEnable[31];  
pSym->bSTRT25 = bEnable[32];  
pSym->bMATRIX25 = bEnable[33];  
pSym->bPLESSEY = bEnable[34];  
pSym->bCHINAPOST = bEnable[35];  
pSym->bKOREAPOST = bEnable[36];  
pSym->bTELEPEN = bEnable[37];  
pSym->bCODE16K = bEnable[38];  
pSym->bPOSICODE = bEnable[39];  
pSym->bCOUPONCODE = bEnable[40];  
pSym->bUSPS4CB = bEnable[41];  
pSym->bIDTAG = bEnable[42];  
pSym->bLABEL = bEnable[43];  
pSym->bGS1\_128 = bEnable[44];  
pSym->bHANXIN = bEnable[45];

```
pSym->bGRIDMATRIX = bEnable[46];  
if(IMAGER_SetSymbology(pSym) == FALSE)  
    ::MessageBox(NULL, L"ERROR : IMAGER_SetSymbology()", NULL, MB_TOPMOST);  
delete pSym;
```

### 4.2.93 IMAGER\_SetSymbologyAll

#### Description

Enables all of Symbologies.

#### Syntax

```
IMAGER_API BOOL IMAGER_SetSymbologyAll();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

IMAGER\_SetSymbologyDefault

#### For .Net

Namespace : ImagerNet.Imager

Function : bool SetSymbologyAll()

#### Example

None

### 4.2.94 IMAGER\_SetSymbologyDefault

#### Description

Initializes all option of scanner.

#### Syntax

```
IMAGER_API BOOL IMAGER_SetSymbologyDefault();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

IMAGER\_SetSymbologyAll

#### For .Net

Namespace : ImagerNet.Imager

Function : bool SetSymbologyDefault()

#### Example

None

### 4.2.95 IMAGER\_SetTELEPEN

#### Description

Sets the option of TELEPEN Barcode

#### Syntax

```
IMAGER_API BOOL IMAGER_SetTELEPEN(PTELEPEN_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

IMAGER\_GetTELEPEN

#### For .Net

Namespace : ImagerNet.Imager

Function : bool SetTELEPEN(ref TELEPEN\_PARAMS pTelepen)

#### Example

```
PTELEPEN_PARAMS pTelepen = new TELEPEN_PARAMS();  
pTelepen->bEnable = m_bEnable;  
pTelepen->bNumeric = m_bNumeric;  
pTelepen->nMinLen = m_nMinLen;  
pTelepen->nMaxLen = m_nMaxLen;  
if(IMAGER_SetTELEPEN(pTelepen) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetTELEPEN()", NULL, MB_TOPMOST);
```

delete pTelepen;

## 4.2.96 IMAGER\_SetUPCA

### Description

Sets the option of UPC-A Barcode

### Syntax

```
IMAGER_API BOOL IMAGER_SetUPCA(PUPCA_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

IMAGER\_GetUPCA

### For .Net

Namespace : ImagerNet.Imager

Function : bool SetUPCA(ref UPCA\_PARAMS pUpca)

### Example

```
PUPCA_PARAMS pUpca = new UPCA_PARAMS();  
pUpca->bEnable = m_bEnable;  
pUpca->bXCD = m_bXCD;  
pUpca->bXNum = m_bXNum;  
pUpca->bAddOn = m_bAddOn;  
if(IMAGER_SetUPCA(pUpca) == FALSE)  
    ::MessageBox(NULL, L"Error : SCAN_SetUPCA()", NULL, MB_TOPMOST);  
delete pUpca;
```

## 4.2.97 IMAGER\_SetUPCE

### Description

Sets the option of UPC-E Barcode

### Syntax

```
IMAGER_API BOOL IMAGER_SetUPCE(PUPCE_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

IMAGER\_GetUPCE

## For .Net

Namespace : ImagerNet.Imager

Function : bool SetUPCE(ref UPCE\_PARAMS pUpce)

## Example

```
PUPCE_PARAMS pUpce = new UPCE_PARAMS();
pUpce->bEnable = m_bEnable;
pUpce->bXCD = m_bXCD;
pUpce->bXNum = m_bXNum;
pUpce->bAddOn = m_bAddOn;
if(IMAGER_SetUPCE(pUpce) == FALSE)
    ::MessageBox(NULL, L"Error : SCAN_SetUPCE()", NULL, MB_TOPMOST);
delete pUpce;
```

## 4.3 CAMERA (CE)

### Status Return value

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

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

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

```

    AF_STATUS_START = 1,
    AF_STATUS_FINISH = 0
};

enum MODEL_TYPE
{
    MODEL_GREEN_13M,
    MODEL_GREEN,
    MODEL_T,
    MODEL_POS,
    MODEL_SMART,
    MODEL_ORANGE_CE,
    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

```

typedef struct
{
    INT    nResolution;      // Resolution
    INT    nSaveFormat;      // SaveFormat
    INT    nJpegQuality;     // Jpeg Quality
    INT    nNamePrefix;      // Image File Name Prefix
    INT    nImgEffect;       // Image Effect
    INT    nImgbalance;      // Image Balance
    INT    nImgRotatesave;   //Image Rotate
    INT    nImgRotateview;   //Image Rotate
} CAMERA_OPTION, *LPCAMERA_OPTION;

typedef struct _ExifInfo
{
    TCHAR TitleName[21];
}

```

```
TCHAR Make[21];
```

```
TCHAR Model[21];
```

```
}ExifInfo;
```

## Functions for Camera (CE)

Name	Description
<a href="#">CAM_Open</a>	Opens the Camera.
<a href="#">CAM_Close</a>	Closes the Camera.
<a href="#">CAM_Capture</a>	Captures the Still Picture.
<a href="#">CAM_PreviewStart</a>	Starts viewing the Preview screen.
<a href="#">CAM_PreviewStop</a>	Stops viewing the Preview screen.
<a href="#">CAM_GetLastSaveFilePath</a>	Gets name of the latest file.
<a href="#">CAM_AutoFocus</a>	Focuses automatically.
<a href="#">CAM_EnableAutoAF</a>	Performs AF by automatically recognizing preview image changes.
<a href="#">CAM_Zoom</a>	Zooms in and/or out.
<a href="#">CAM_SetCameraOption</a>	Sets the Camera options.
<a href="#">CAM_GetCameraOption</a>	Gets currently set Options of Camera.
<a href="#">CAM_Brightness</a>	Changes the Brightness of Camera.
<a href="#">CAM_FlashOn</a>	Turns on the Flash.
<a href="#">CAM_FlashOff</a>	Turns off the Flash.
<a href="#">CAM_RawData</a>	Gets the raw data of Preview screen.
<a href="#">CAM_InsertExifInformation</a>	Insert EXIF information to Jpeg image.
<a href="#">CAM_UseGPSEXifData</a>	Insert GPS information in EXIF Information.
<a href="#">CAM_RegisterMsgWnd</a>	Register the window handle getting Window Message from Camera.
<a href="#">CAM_GetScannerType</a>	Gets Scanner Type of device.
<a href="#">CAM_GetVersion</a>	Gets dll version.

### 4.3.1 CAM\_Open

#### Description

Opens the Camera.

#### Syntax

```
MODEL_TYPE CAM_Open(HWND hMainWnd, HWND 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

CAM\_Close

#### For .Net

Namespace : CamNet.Cam

Function : Cam.MODEL\_TYPE Open(IntPtr hMainWnd, IntPtr hPreviewWnd)

#### Example

```
MODEL_TYPE m_Model;

m_Model = CAM_Open(m_hWnd, m_ctrPreview.m_hWnd) ;

if(m_Model == MODEL_UNKNOWN)
{
    ::MessageBox(NULL, L"Open error", L"Open", NULL);
    return;
}
```

### 4.3.2 CAM\_Close

#### Description

Closes the Camera.

#### Syntax

```
CAM_EXPORTS BOOL CAM_Close()
```

#### Parameters

None

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

CAM\_Open

#### **For .Net**

Namespace : CamNet.Cam

Function : bool Close()

#### **Example**

```
CAM_Close();
```

### **4.3.3 CAM\_Capture**

#### **Description**

Captures the Still Picture.

#### **Syntax**

```
CAM_EXPORTS BOOL CAM_Capture(LPCTSTR strFilePath);
```

#### **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 .Net**

Namespace : CamNet.Cam

Function : bool Capture(string strPath)

#### **Example**

```
TCHAR m_tzSavePath[256] = {0x00};
```

```
if(!m_bCapturing)
```

```
{
```

```
CAM_Capture(m_tzSavePath);  
}
```

#### 4.3.4 CAM\_PreviewStart

##### Description

Starts viewing the Preview screen.

##### Syntax

```
CAM_EXPORTS BOOL CAM_PreviewStart();
```

##### Parameters

None

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

CAM\_PreviewStop

##### For .Net

Namespace : CamNet.Cam

Function : bool PreviewStart()

##### Example

```
CAM_PreviewStart();
```

#### 4.3.5 CAM\_PreviewStop

##### Description

Stops viewing the Preview screen.

##### Syntax

```
CAM_EXPORTS BOOL CAM_PreviewStop();
```

##### Parameters

None

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also



CAM\_PreviewStart

#### **For .Net**

Namespace : CamNet.Cam

Function : bool PreviewStop()

#### **Example**

```
CAM_PreviewStop();
```

### **4.3.6 CAM\_GetLastSaveFilePath**

#### **Description**

Gets name of the latest file.

#### **Syntax**

```
CAM_EXPORTS BOOL CAM_GetLastSaveFilePath(LPTSTR strOutFilePath);
```

#### **Parameters**

*strOutFilePath*

Obtains the last captured picture's name.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

CAM\_Capture

#### **For .Net**

Namespace : CamNet.Cam

Function : bool GetLastSaveFilePath(out string strFilePath);

#### **Example**

```
TCHAR m_tzSavePath[256] = {0x00};
```

```
CAM_GetLastSaveFilePath(m_tzSavePath);
```

### **4.3.7 CAM\_AutoFocus**

#### **Description**

Focuses automatically.

#### **Syntax**

```
CAM_EXPORTS BOOL CAM_AutoFocus();
```

#### **Parameters**

None

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

CAM\_EnableAutoAF

#### **For .Net**

Namespace : CamNet.Cam

Function : bool AutoFocus()

#### **Example**

```
CAM_AutoFocus();
```

### **4.3.8 CAM\_EnableAutoAF**

#### **Description**

Performs AF by automatically recognizing preview image changes.

#### **Syntax**

```
CAM_EXPORTS BOOL CAM_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**

None

#### **See Also**

CAM\_AutoFocus

#### **For .Net**

Namespace : CamNet.Cam

Function : bool EnableAutoAF(bool bEnable)

#### **Example**

```
CAM_EnableAutoAF(TRUE);
```

### 4.3.9 CAM\_Zoom

#### Description

Zooms in and/or out.

#### Syntax

```
CAM_EXPORTS BOOL CAM_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 .Net

Namespace : CamNet.Cam

Function : int Zoom(int index)

#### Example

```
CAM_Zoom(1);
```

### 4.3.10 CAM\_SetCameraOption

#### Description

Sets the Camera options.

#### Syntax

```
CAM_EXPORTS BOOL CAM_SetCameraOption(LPCAMERA_OPTION option, LPTSTR strSaveFolder);
```

#### Parameters

*option*

LPCAMERA\_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

CAM\_GetCameraOption

## For .Net

Namespace : CamNet.Cam

Function : bool SetCameraOption(ref Cam.CAMERA\_OPTION option, string strSavePath)

## Example

```
CAMERA_OPTION CamOption;  
CamOption.nImgEffect = OPTION_IMAGE_NOMAL_EFFECT;  
CamOption.nResolution = OPTION_RESOLUTION_1280X1024;  
CAM_SetCameraOption(&m_CamOption, L"Flash Disk\\Camera");
```

### 4.3.11 CAM\_GetCameraOption

#### Description

Gets currently set Options of Camera.

#### Syntax

```
CAM_EXPORTS BOOL CAM_GetCameraOption(LPCAMERA_OPTION option, LPTSTR strSaveFolder);
```

#### Parameters

*option*

LPCAMERA\_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

CAM\_SetCameraOption

## For .Net

Namespace : CamNet.Cam

Function : bool SetCameraOption(ref Cam.CAMERA\_OPTION option, string strSavePath)

## Example

```
TCHAR tzSavePath[260] = {0x00};  
CAMERA_OPTION CamOption;  
CAM_GetCameraOption(&CamOption, m_tzSavePath);
```

### 4.3.12 CAM\_Brightness

#### Description

Changes the Brightness of Camera.

#### Syntax

```
CAM_EXPORTS BOOL CAM_Brightness(int nBrightness);
```

#### Parameters

*nBrightness*

Sets the brightness.

#### Return Value

Nonzero indicates success. Zero indicates failure

#### Remarks

None

#### See Also

CAM\_SetCameraOption, CAM\_GetCameraOption

#### For .Net

Namespace : CamNet.Cam

Function : bool Brightness(int nBrightness);

#### Example

```
CAM_Brightness(2);
```

### 4.3.13 CAM\_FlashOn

#### Description

Turns on the Flash.

#### Syntax

```
CAM_EXPORTS BOOL CAM_FlashOn();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

CAM\_FlashOff

#### For .Net

Namespace : CamNet.Cam

Function : bool FlashOn()

#### **Example**

```
CAM_FlashOn();
```

### **4.3.14 CAM\_FlashOff**

#### **Description**

Turns off the Flash.

#### **Syntax**

```
CAM_EXPORTS BOOL CAM_FlashOff();
```

#### **Parameters**

None

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

CAM\_FlashOn

#### **For .Net**

Namespace : CamNet.Cam

Function : bool FlashOff()

#### **Example**

```
CAM_FlashOff();
```

### **4.3.15 CAM\_RawData**

#### **Description**

Obtains data of Preview screen.

#### **Syntax**

```
CAM_EXPORTS void CAM_RawData(byte* data);
```

#### **Parameters**

*data*

Obtains data of current preview screen.

#### **Return Value**

void

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : CamNet.Cam

Function : void RawData(byte[] data)

**Example**

None

### 4.3.16 CAM\_InsertExifInformation

**Description**

Inserts EXIF information to Jpeg image.

**Syntax**

```
CAM_API BOOL CAM_InsertExifInformation(TCHAR* FileName, ExifInfo info);
```

**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**

CAM\_UseGPSExifData

**For .Net**

Namespace : CamNet.Cam

Function : bool InsertExifInformation(string FileName, Cam.ExifInfo e)

**Example**

```
ExifInfo info;
```

```
_tcscopy(info.TitleName, _T("Picture0.jpg"));
```

```
_tcscopy(info.Make, _T("M3 Mobile Co.Ltd"));
```

```
_tcscopy(info.Model, _T("M3 SMART"));
```

```
CAM_InsertExifInformation(L"Flash Disk\\Camera\\Photo\\Picture0.jpg", info);
```

### 4.3.17 CAM\_UseGPSExifData

#### Description

Inserts GPS information to EXIF Information.

#### Syntax

```
CAM_API BOOL CAM_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

CAM\_InsertExifInformation

#### For .Net

Namespace : CamNet.Cam

Function : bool UseGPSExifData(bool bUse)

#### Example

```
CAM_UseGPSExifData(TRUE);
```

### 4.3.18 CAM\_RegisterMsgWnd

#### Description

Register the window handle getting Window Message from Camera.

#### Syntax

```
CAM_EXPORTS void CAM_RegisterMsgWnd(HWND hWnd);
```

#### Parameters

*hWnd*

Window Handle that gets messages from library

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None



**See Also**

None

**For .Net**

Namespace : CamNet.Cam

Function : void RegisterMsgWnd(IntPtr hWnd);

**Example**

None

### 4.3.19 CAM\_GetScannerType

**Description**

Gets Scanner Type of device.

**Syntax**

```
CAM_EXPORTS SCANNER_TYPE CAM_GetScannerType();
```

**Parameters**

None

**Return Value**

SCANNER\_TYPE is Enum type value. Gets scanner type of device.

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : CamNet.Cam

Function : bool GetScannerType()

**Example**

```
SCANNER_TYPE type = CAM_GetScannerType();
```

### 4.3.20 CAM\_GetVersion

**Description**

Gets current version of Camera DLL.

**Syntax**

```
CAM_EXPORTS BOOL CAM_GetVersion(TCHAR* tzDllVersion, TCHAR* tzReleaseDate, TCHAR*  
tzPixels);
```

**Parameters**

*tzDllVersion*

Obtains DLL version.

*tzReleaseDate*

Obtains released date of DLL.

*tzPixels*

Obtains pixels of Camera.

### **Return Value**

Nonzero indicates success. Zero indicates failure.

### **Remarks**

None

### **See Also**

None

### **For .Net**

Namespace : CamNet.Cam

Function : bool GetVersion()

### **Example**

None

## 4.4 CAMERA (WM)

### Status Return value

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

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

Define
<pre>#define WM_STATUS_AF WM_USER + 916 #define WM_CONVERT WM_USER + 232 #define WM_GPS_ONWM_USER + 1020</pre>
Enum
<pre>typedef enum {     MODEL_SKY,     MODEL_MM3,     MODEL_ORANGE,     MODEL_SMART,     MODEL_UNKONWN,     MODEL_ORANGE_PLUS } MODEL_TYPE;  typedef enum {     SCANNER_OPTICON,     SCANNER_SYMBOL,     SCANNER_INTERMEC,     SCANNER_HHP } SCANNER_TYPE;  typedef enum {     STILL_MODE,     VIDEO_MODE,     CLOSE_MODE } CAMERA_MODE;  typedef enum {     VIDEO_ASF,     VIDEO_WMV } VIDEO_TYPE;  typedef enum {     AF_MANUAL,     AF_ALWAYS,     AF_AUTO } AF_MODE;</pre>

```
typedef enum{
    AF_STATUS_READY,
    AF_STATUS_START,
    AF_STATUS_FINISH
} AF_STATUS;
typedef enum{
    WB_Auto,
    WB_Sunny,
    WB_Cloudy,
    WB_Fluorescent,
    WB_Incandescent
}WHITE_BALANCE;
typedef enum{
    SAVE_DATE=0,
    SAVE_CUSTIM=1
}SAVE_TYPE;
```

#### Structure

```
typedef struct _ExifInfo
{
    TCHAR TitleName[21];
    TCHAR Make[21];
    TCHAR Model[21];
}ExifInfo;
```

## Functions for Camera (WM)

Name	Description
<a href="#">CAM_Open</a>	Opens the Camera.
<a href="#">CAM_Close</a>	Closes the Camera.
<a href="#">CAM_SetPreviewWindow</a>	Sets the size of Preview screen.
<a href="#">CAM_PreviewStart</a>	Starts viewing the Preview screen.
<a href="#">CAM_PreviewStop</a>	Stops viewing the Preview screen.
<a href="#">CAM_GetRegCapturePath</a>	Get save filename. (old version)
<a href="#">CAM_GetRegCapturePathEx</a>	Get save filename.
<a href="#">CAM_SetRegCapturePath</a>	Set save filename.
<a href="#">CAM_Capture</a>	Captures the Still Picture.
<a href="#">CAM_CaptureEx</a>	Captures the Still Picture.
<a href="#">CAM_EnableShutterSound</a>	Set shutter sound enable/disable.
<a href="#">CAM_VideoStart</a>	Starts Videoing.
<a href="#">CAM_VideoStop</a>	Stops Videoing.
<a href="#">CAM_VideoStopEx</a>	Stops Videoing.
<a href="#">CAM_GetFlashState</a>	Sets whether the flash is on or not when capturing picture.
<a href="#">CAM_AlwaysFlash</a>	Turns on the Flash all the time.
<a href="#">CAM_CaptureFlash</a>	Sets whether the flash is on or not when capturing picture.
<a href="#">CAM_AutoFocus</a>	Focuses automatically.
<a href="#">CAM_SetAfMode</a>	Sets AF Mode.
<a href="#">CAM_Zoom</a>	Zooms in and/or out.
<a href="#">CAM_SetResolution</a>	Sets Resolution of picture.
<a href="#">CAM_GetResolution</a>	Gets setting value of resolution.
<a href="#">CAM_SetQuality</a>	Sets quality of Jpeg Still Shot.
<a href="#">CAM_GetQuality</a>	Gets quality value of Jpeg Still shot.
<a href="#">CAM_SetBrightness</a>	Sets brightness of camera.
<a href="#">CAM_GetBrightness</a>	Gets brightness of camera.
<a href="#">CAM_SetWhiteBalance</a>	Sets white balance of camera.
<a href="#">CAM_GetWhiteBalance</a>	Gets white balance of camera.
<a href="#">CAM_SetContrast</a>	Sets contrast of camera.
<a href="#">CAM_GetContrast</a>	Gets contrast of camera.

<a href="#"><u>CAM_SetSharpness</u></a>	Sets sharpness of camera.
<a href="#"><u>CAM_GetSharpness</u></a>	Gets sharpness of camera.
<a href="#"><u>CAM_SetHistoEqual</u></a>	Sets whether performing of Histogram Equalization when taking pictures.
<a href="#"><u>CAM_GetHistoEqual</u></a>	Gets whether performing of Histogram Equalization when taking pictures.
<a href="#"><u>CAM_RegisterPreview</u></a>	Registers Callback function to get the preview screen.
<a href="#"><u>CAM_RegisterMsgWnd</u></a>	Registers Window to recive the message from Camera DLL.
<a href="#"><u>CAM_InsertExifInformation</u></a>	Insert EXIF information to Jpeg image.
<a href="#"><u>CAM_GetRegExifEnable</u></a>	Get use EXIF function.
<a href="#"><u>CAM_UseGPSExifData</u></a>	Insert GPS information in EXIF Information.
<a href="#"><u>CAM_GetRegExifGpsEnable</u></a>	Get use GPS function.
<a href="#"><u>CAM_GetRegInsertDateTimeEnable</u></a>	Get use insert datetime function.
<a href="#"><u>CAM_SetInsertDateTimeEnable</u></a>	Set insert datetime function enable/disable.
<a href="#"><u>CAM_GetModelType</u></a>	Gets model type of device.
<a href="#"><u>CAM_GetScannerType</u></a>	Gets Scanner Type of device.
<a href="#"><u>CAM_GetVersion</u></a>	Gets dll version.

### 4.4.1 CAM\_Open

#### Description

Opens the Camera.

#### Syntax

```
CAM_API BOOL CAM_Open(HWND 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

CAM\_Close

#### For .Net

Namespace : CamNet.Cam

Function : bool Open()

#### Example

```
MODEL_TYPE m_Model;  
m_Model = CAM_Open(m_hWnd, m_ctrPreview.m_hWnd) ;  
if(m_Model == MODEL_UNKNOWN)  
{  
    ::MessageBox(NULL, L"Open error", L"Open", NULL);  
    return;  
}
```

### 4.4.2 CAM\_Close

#### Description

Closes the Camera.

**Syntax**

```
CAM_API BOOL CAM_Close();
```

**Parameters**

None

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

CAM\_Open

**For .Net**

Namespace : CamNet.Cam

Function : bool Close()

**Example**

```
CAM_Close();
```

### 4.4.3 CAM\_SetPreviewWindow

**Description**

Sets the size of Preview screen.

**Syntax**

```
CAM_API BOOL CAM_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**

CAM\_PreviewStart, CAM\_PreviewStop

**For .Net**

Namespace : CamNet.Cam

Function : bool SetPreviewWindow(long left, long top, long width, long height)

**Example**

```
CAM_SetPreviewWindow(80,90,320,240); // in MM3
```

#### 4.4.4 CAM\_PreviewStart

**Description**

Starts viewing the Preview screen.

**Syntax**

```
CAM_API BOOL CAM_PreviewStart();
```

**Parameters**

None

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

CAM\_PreviewStop

**For .Net**

Namespace : CamNet.Cam

Function : bool PreviewStart()

**Example**

```
CAM_PreviewStart();
```

#### 4.4.5 CAM\_PreviewStop

**Description**

Stops viewing the Preview screen.

**Syntax**

```
CAM_API BOOL CAM_PreviewStop();
```

**Parameters**

None

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

CAM\_PreviewStart

**For .Net**

Namespace : CamNet.Cam

Function : bool PreviewStop()

**Example**

```
CAM_PreviewStop();
```

## 4.4.6 CAM\_GetRegCapturePath

**Description**

Get save filename..

**Syntax**

```
CAM_API BOOL CAM_GetRegCapturePath(TCHAR* tzFullName);
```

**Parameters**

*tzFullName*

Filename in capture.

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

CAM\_SetRegCapturePath

**For .Net**

Namespace : CamNet.Cam

Function : bool GetRegCapturePath(StringBuilder strPath)

**Example**

```
TCHAR m_filename[260]={0,};
```

```
CAM_GetRegCapturePath(m_filename);
```

## 4.4.7 CAM\_GetRegCapturePathEx

### Description

Get save filename.

### Syntax

```
CAM_API BOOL CAM_GetRegCapturePathEx(TCHAR* tzPath, TCHAR* tzName);
```

### Parameters

*tzPath*

Save path in capture.

*tzName*

Save name in capture.

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

CAM\_SetRegCapturePath

### For .Net

Namespace : CamNet.Cam

Function : bool GetRegCapturePathEx(StringBuilder strPath, StringBuilder strName)

### Example

```
TCHAR m_szPath[260]={0,};
```

```
TCHAR m_szName[260]={0,};
```

```
CAM_GetRegCapturePathEx(m_szPath, m_szName);
```

## 4.4.8 CAM\_SetRegCapturePath

### Description

Set save filename.

### Syntax

```
CAM_API BOOL CAM_SetRegCapturePath(TCHAR* tzPath, TCHAR* tzName);
```

### Parameters

*strPath*

Set save filepath.

*strName*

Set save filename.

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

CAM\_GetRegCapturePathEx

**For .Net**

Namespace : CamNet.Cam

Function : bool SetRegCapturePath(StringBuilder strPath, StringBuilder strName)

**Example**

```
TCHAR* strPath=_T("\\Flash Disk\\Camera\\");
```

```
TCHAR* strName=_T("123.jpg");
```

```
CAM_SetRegCapturePath(strPath, strName);
```

## 4.4.9 CAM\_Capture

**Description**

Captures the Still Picture. This is an old version. New version uses CAM\_CaptureEx.

**Syntax**

```
CAM_API BOOL CAM_Capture(const TCHAR* tzInFileName, TCHAR* tzOutFileName, BOOL  
bAutoInit);
```

**Parameters**

*tzInFileName*

Input file name.

*btzOutFileName*

Obtains file name.

*bAutoInit*

Auto preview enable/disable after capture.

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : CamNet.Cam

Function : `bool Capture(string tzInFileName, char[] tzOutFileName, bool bAutoInit)`

#### Example

```
TCHAR tzFile[MAX_PATH] = {0x00};  
TCHAR tzOutFile[MAX_PATH] = {0x00};  
memset(tzFile, 0x00, sizeof(TCHAR) * MAX_PATH);  
memset(tzOutFile, 0x00, sizeof(TCHAR) * MAX_PATH);  
wsprintf(tzFile, L"\\Flash Disk\\Cam\\Picture1.jpg");  
m_bAutoInit = TRUE;  
CAM_Capture(tzFile, tzOutFile, m_bAutoInit);
```

### 4.4.10 CAM\_CaptureEx

#### Description

Captures the Still Picture.

#### Syntax

`CAM_API BOOL CAM_CaptureEx(SAVE_TYPE nSaveType, BOOL bAutoInit, TCHAR* 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 .Net

Namespace : CamNet.Cam

Function : `bool CaptureEx(string tzInFileName, char[] tzOutFileName, bool bAutoInit)`

#### Example

```
TCHAR tzOutFile[MAX_PATH] = {0x00};  
memset(tzOutFile, 0x00, sizeof(TCHAR) * MAX_PATH);
```

```
m_bAutoInit = TRUE;  
CAM_CaptureEx(SAVE_DATE, bAutoInit, m_tzOutFile);
```

#### 4.4.11 CAM\_EnableShutterSound

##### Description

Enable/disable shutter sound in capture.

##### Syntax

```
CAM_API BOOL CAM_EnableShutterSound(BOOL bEnable);
```

##### Parameters

*bEnable*.

TRUE is enable, False is disable.

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

None.

##### For .Net

Namespace : CamNet.Cam

Function : bool EnableShutterSound()

##### Example

```
CAM_EnableShutterSound(TRUE);
```

#### 4.4.12 CAM\_VideoStart

##### Description

Starts Video capturing.

##### Syntax

```
CAM_API BOOL CAM_VideoStart();
```

##### Parameters

None.

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

## See Also

CAM\_VideoStop

## For .Net

Namespace : CamNet.Cam

Function : bool VideoStart()

## Example

```
CAM_VideoStart();
```

## 4.4.13 CAM\_VideoStop

### Description

Stops Video capturing

### Syntax

```
CAM_API BOOL CAM_VideoStop(const TCHAR* tzInFileName, TCHAR* tzOutFileName);
```

### Parameters

*tzInFileName*

Input filename.

*tzOutFileName*

Obtains file name.

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

## See Also

CAM\_VideoStart

## For .Net

Namespace : CamNet.Cam

Function : bool VideoStop(char[] tzInFileName, char[] tzOutFileName)

## Example

```
TCHAR tzFile[MAX_PATH] = {0x00};  
TCHAR tzOutFile[MAX_PATH] = {0x00};  
memset(tzFile, 0x00, sizeof(TCHAR) * MAX_PATH);  
memset(tzOutFile, 0x00, sizeof(TCHAR) * MAX_PATH);  
wsprintf(tzFile, L"\\Flash Disk\\Camera\\ ");  
CAM_VideoStop(tzFile, tzOutFile);
```

#### 4.4.14 CAM\_VideoStopEx

##### Description

Stops Video capturing

##### Syntax

```
CAM_API BOOL CAM_VideoStopEx(SAVE_TYPE nSaveType, TCHAR* tzOutFileName);
```

##### Parameters

*nSaveType*

SAVE\_TYPE.

*tzOutFileName*

Obtains file name.

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

CAM\_VideoStart

##### For .Net

Namespace : CamNet.Cam

Function : bool VideoStopEx(SAVE\_TYPE nSaveType, char[] tzOutFileName)

##### Example

```
TCHAR tzFile[MAX_PATH] = {0x00};  
TCHAR tzOutFile[MAX_PATH] = {0x00};  
memset(tzFile, 0x00, sizeof(TCHAR) * MAX_PATH);  
memset(tzOutFile, 0x00, sizeof(TCHAR) * MAX_PATH);  
wsprintf(tzFile, L"\\Flash Disk\\Camera\\");  
CAM_VideoStopEx(tzFile, tzOutFile);
```

#### 4.4.15 CAM\_GetFlashState

##### Description

Get Flash Status.

##### Syntax

```
CAM_API BOOL CAM_GetFlashState();
```

##### Parameters



None.

#### **Return Value**

Nonzero indicates flash on. Zero indicates flash off.

#### **Remarks**

None.

#### **See Also**

None.

#### **For .Net**

Namespace : CamNet.Cam

Function : bool GetFlashState()

#### **Example**

```
BOOL bFlashOn = CAM_GetFlashState();
```

### **4.4.16 CAM\_AlwaysFlash**

#### **Description**

Turns on the Flash all the time.

#### **Syntax**

```
CAM_API BOOL CAM_AlwaysFlash(BOOL bOn);
```

#### **Parameters**

*bOn*

Sets the flash mode.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

Flash should be turned off before the Camera program is closed.

#### **See Also**

CAM\_CaptureFlash

#### **For .Net**

Namespace : CamNet.Cam

Function : bool AlwaysFlash(bool bOn)

#### **Example**

```
CAM_AlwaysFlash(true);
```

#### 4.4.17 CAM\_CaptureFlash

##### Description

Sets whether the flash is on or not when capturing picture

##### Syntax

```
CAM_API BOOL CAM_CaptureFlash(BOOL bOn);
```

##### Parameters

*bOn*

Sets the flash mode.

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

CAM\_AlwaysFlash

##### For .Net

Namespace : CamNet.Cam

Function : bool CaptureFlash(int bOn)

##### Example

```
CAM_CaptureFlash(true);
```

#### 4.4.18 CAM\_AutoFocus

##### Description

Focuses automatically.

##### Syntax

```
CAM_API BOOL CAM_AutoFocus();
```

##### Parameters

None

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

CAM\_SetAfMode

##### For .Net

Namespace : CamNet.Cam

Function : bool AutoFocus()

#### **Example**

```
CAM_AutoFocus();
```

### **4.4.19 CAM\_SetAfMode**

#### **Description**

Sets AF Mode.

#### **Syntax**

```
CAM_API AF_MODE CAM_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**

CAM\_AutoFocus

#### **For .Net**

Namespace : CamNet.Cam

Function : Cam.AF\_MODE SetAfMode(Cam.AF\_MODE mode)

#### **Example**

```
CAM_SetAfMode(0);
```

### **4.4.20 CAM\_Zoom**

#### **Description**

Zooms in and/or out.

#### **Syntax**

```
CAM_API BOOL CAM_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 .Net**

Namespace : CamNet.Cam

Function : int Zoom(int index)

#### **Example**

```
CAM_Zoom(1);
```

### **4.4.21 CAM\_SetResolution**

#### **Description**

Sets Resolution of picture.

#### **Syntax**

```
CAM_API BOOL CAM_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**

CAM\_GetResolution

#### **For .Net**

Namespace : CamNet.Cam

Function : int SetResolution(int nResolution)

#### **Example**

```
CAM_SetResolution(0);
```

#### 4.4.22 CAM\_GetResolution

##### Description

Gets setting value of resolution.

##### Syntax

```
CAM_API int CAM_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

CAM\_SetResolution

##### For .Net

Namespace : CamNet.Cam

Function : int CAM\_GetResolution ()

##### Example

```
int nResolution = CAM_GetResolution();
```

#### 4.4.23 CAM\_SetQuality

##### Description

Sets quality of Jpeg Still Shot.

##### Syntax

```
CAM_API BOOL CAM_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

CAM\_GetQuality

**For .Net**

Namespace : CamNet.Cam

Function : bool SetQuality(int nQuality)

**Example**

CAM\_SetQuality(0);

#### 4.4.24 CAM\_GetQuality

**Description**

Gets quality value of Jpeg Still shot.

**Syntax**

CAM\_API int CAM\_GetQuality();

**Parameters**

None

**Return Value**

Gets quality value of Jpeg Still Shot.

**Remarks**

None

**See Also**

CAM\_SetQuality

**For .Net**

Namespace : CamNet.Cam

Function : int GetQuality()

**Example**

int nQuality = CAM\_GetQuality();

#### 4.4.25 CAM\_SetBrightness

**Description**

Sets brightness of camera.

**Syntax**

CAM\_API BOOL CAM\_SetBrightness(int nBrightness);

**Parameters**

*nBrightness*

Value's range is +4~-4.

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

CAM\_GetBrightness

**For .Net**

Namespace : CamNet.Cam

Function : bool SetBrightness(int nBrightness)

**Example**

```
CAM_SetBrightness(0);
```

## 4.4.26 CAM\_GetBrightness

**Description**

Gets brightness of camera.

**Syntax**

```
CAM_API int CAM_GetBrightness();
```

**Parameters**

None

**Return Value**

Gets birhgtness of camera.

**Remarks**

None

**See Also**

CAM\_SetBrightness

**For .Net**

Namespace : CamNet.Cam

Function : int GetBrightness()

**Example**

```
int nBrightness = CAM_GetBrightness();
```

## 4.4.27 CAM\_SetWhiteBalance

**Description**

Sets white balance of camera.

### Syntax

```
CAM_API BOOL CAM_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

CAM\_GetWhiteBalance

### For .Net

Namespace : CamNet.Cam

Function : bool SetWhiteBalance(Cam.WHITE\_BALANCE nWhiteBalance)

### Example

```
CAM_SetWhiteBalance(WB_Auto);
```

## 4.4.28 CAM\_GetWhiteBalance

### Description

Gets white balance of camera.

### Syntax

```
CAM_API WHITE_BALANCE CAM_GetWhiteBalance();
```

### Parameters

None

### Return Value

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

### Remarks

None

### See Also

CAM\_SetWhiteBalance

### For .Net

Namespace : CamNet.Cam

Function : Cam.WHITE\_BALANCE GetWhiteBalance()

### Example



```
WHITE_BALANCE wbValue = CAM_GetWhiteBalance();
```

#### 4.4.29 CAM\_SetContrast

##### Description

Sets contrast of camera.

##### Syntax

```
CAM_API BOOL CAM_SetContrast(int nContrast);
```

##### Parameters

*nContrast*

Inputs contrast value of camera.

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

CAM\_GetContrast

##### For .Net

Namespace : CamNet.Cam

Function : bool SetContrast(int nContrast)

##### Example

```
CAM_SetContrast(1);
```

#### 4.4.30 CAM\_GetContrast

##### Description

Gets contrast of camera.

##### Syntax

```
CAM_API int CAM_GetContrast();
```

##### Parameters

none

##### Return Value

Gets contrast of camera.

##### Remarks

None

##### See Also

CAM\_SetContrast

#### **For .Net**

Namespace : CamNet.Cam

Function : int GetContrast()

#### **Example**

```
int nContrast = CAM_GetContrast();
```

### **4.4.31 CAM\_SetSharpness**

#### **Description**

Sets sharpness of camera.

#### **Syntax**

```
CAM_API BOOL CAM_SetSharpness(int nSharpness);
```

#### **Parameters**

*nSharpness*

Inputs sharpness value of camera.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

CAM\_GetSharpness

#### **For .Net**

Namespace : CamNet.Cam

Function : bool SetSharpness(int nSharpness)

#### **Example**

```
CAM_SetSharpness(0);
```

### **4.4.32 CAM\_GetSharpness**

#### **Description**

Gets sharpness of camera.

#### **Syntax**

```
CAM_API int CAM_GetSharpness();
```

#### **Parameters**

none

**Return Value**

Gets sharpness value of camera.

**Remarks**

None

**See Also**

CAM\_SetSharpness

**For .Net**

Namespace : CamNet.Cam

Function : int GetSharpness()

**Example**

```
Int nSharpness = CAM_GetSharpness();
```

### 4.4.33 CAM\_SetHistoEqual

**Description**

Sets whether performing of Histogram Equalization when taking pictures.

**Syntax**

```
CAM_API void CAM_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**

CAM\_GetHistoEqual

**For .Net**

Namespace : CamNet.Cam

Function : void SetHistoEqual(bool Enable)

**Example**

```
CAM_SetHistoEqual(TRUE);
```

### 4.4.34 CAM\_GetHistoEqual

**Description**

Gets whether performing of Histogram Equalization when taking pictures.

#### **Syntax**

```
CAM_API BOOL CAM_GetHistoEqual();
```

#### **Parameters**

None

#### **Return Value**

Gets whether performing of Histogram Equalization when taking pictures.

#### **Remarks**

None

#### **See Also**

CAM\_SetHistoEqual

#### **For .Net**

Namespace : CamNet.Cam

Function : int GetHistoEqual()

#### **Example**

```
BOOL bEnableHisto = CAM_GetHistEqual();
```

### **4.4.35 CAM\_RegisterPreview**

#### **Description**

Registers Callback function to get the preview screen.

#### **Syntax**

```
CAM_API BOOL CAM_RegisterPreview(MANAGED_SAMPLEPROCESSEDPROC fpPreview);
```

#### **Parameters**

*fpPreview*

CallBack Procedure formed a MANAGED\_SAMPLEPROCESSEDPROC function.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

MANAGED\_SAMPLEPROCESSEDPROC

#### **Example**

None

#### 4.4.36 CAM\_RegisterMsgWnd

##### Description

Registers Window to receive the message from Camera DLL.

##### Syntax

```
CAM_API void CAM_RegisterMsgWnd(HWND hWnd);
```

##### Parameters

*hWnd*

Windows Handle that receives message from the library.

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

None

##### For .Net

Namespace : CamNet.Cam

Function : void RegisterMsgWnd(IntPtr hWnd);

##### Example

None

#### 4.4.37 CAM\_InsertExifInformation

##### Description

Insert EXIF information to Jpeg image.

##### Syntax

```
CAM_API BOOL CAM_InsertExifInformation(TCHAR* FileName, ExifInfo info);
```

##### 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**

CAM\_GetRegExifEnable

**For .Net**

Namespace : CamNet.Cam

Function : bool InsertExifInformation(string FileName, Cam.ExifInfo e)

**Example**

ExifInfo info;

\_tcscpy(info.TitleName, \_T("Picture0.jpg"));

\_tcscpy(info.Make, \_T("M3 Mobile Co.Ltd"));

\_tcscpy(info.Model, \_T("M3 SKY"));

CAM\_InsertExifInformation(L"Flash Disk\\Camera\\Photo\\Picture0.jpg", info);

### 4.4.38 CAM\_GetRegExifEnable

**Description**

Get use EXIF function.

**Syntax**

CAM\_API BOOL CAM\_GetRegExifEnable();

**Parameters**

None.

**Return Value**

Nonzero indicates enable. Zero indicates disable.

**Remarks**

None

**See Also**

CAM\_InsertExifInformation

**For .Net**

Namespace : CamNet.Cam

Function : bool GetRegExifEnable()

**Example**

BOOL bExifEnable = CAM\_GetRegExifEnable();

### 4.4.39 CAM\_UseGPSExifData

**Description**

Insert GPS information in EXIF Information.

**Syntax**

```
CAM_API BOOL CAM_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**

CAM\_GetRegExifGpsEnable

**For .Net**

Namespace : CamNet.Cam

Function : bool UseGPSExifData(bool bUse)

**Example**

```
CAM_UseGPSExifData(TRUE);
```

#### 4.4.40 CAM\_GetRegExifGpsEnable

**Description**

Get use GPS function.

**Syntax**

```
CAM_API BOOL CAM_GetRegExifGpsEnable();
```

**Parameters**

None.

**Return Value**

Nonzero indicates enable. Zero indicates disable.

**Remarks**

None.

**See Also**

CAM\_UseGpsExifData

**For .Net**

Namespace : CamNet.Cam

Function : bool GetRegExifGpsEnable()

**Example**

```
BOOL bGpsEnable = CAM_GetRegExifGpsEnable();
```

#### 4.4.41 CAM\_GetRegInsertDateTimeEnable

##### Description

Get use GPS function.

##### Syntax

```
CAM_API BOOL CAM_GetRegInsertDateTimeEnable();
```

##### Parameters

None.

##### Return Value

Nonzero indicates enable. Zero indicates disable.

##### Remarks

None.

##### See Also

CAM\_SetInsertDateTimeEnable

##### For .Net

Namespace : CamNet.Cam

Function : bool GetRegInsertDateTimeEnable()

##### Example

```
BOOL bDateTimeEnable = CAM_GetRegInsertDateTimeEnable();
```

#### 4.4.42 CAM\_SetInsertDateTimeEnable

##### Description

Get use GPS function.

##### Syntax

```
CAM_API BOOL CAM_SetInsertDateTimeEnable(int bEnable);
```

##### Parameters

*bEnable*

1 is enable, 0 is disable.

##### Return Value

Nonzero indicates enable. Zero indicates disable.

##### Remarks

None.

##### See Also



CAM\_GetRegInsertDateTimeEnable

**For .Net**

Namespace : CamNet.Cam

Function : bool SetInsertDateTimeEnable

**Example**

CAM\_SetInsertDateTimeEnable(TRUE);

### 4.4.43 CAM\_GetModelType

**Description**

Gets model type of device.

**Syntax**

CAM\_API MODEL\_TYPE CAM\_GetModelType();

**Parameters**

None

**Return Value**

MODEL\_TYPE is Enum Type value. Obtains model type of device.

**Remarks**

None

**See Also**

CAM\_GetScannerType

**For .Net**

Namespace : CamNet.Cam

Function : MODEL\_TYPE GetModelType();

**Example**

MODEL\_TYPE type = CAM\_GetModelType();

### 4.4.44 CAM\_GetScannerType

**Description**

Gets Scanner Type of device.

**Syntax**

CAM\_API SCANNER\_TYPE CAM\_GetScannerType();

**Parameters**

None

**Return Value**

SCANNER\_TYPE is Enum type value. Obtains scanner type of device.

**Remarks**

None

**See Also**

CAM\_GetModelType

**For .Net**

Namespace : CamNet.Cam

Function : bool GetScannerType()

**Example**

SCANNER\_TYPE type = CAM\_GetScannerType();

### 4.4.45 CAM\_GetVersion

**Description**

Gets dll version.

**Syntax**

CAM\_API BOOL CAM\_GetVersion(TCHAR\* tzDllVersion, TCHAR\* tzDllRelease);

**Parameters**

*tzDllVersion*

Obtains DLL version.

*tzDllRelease*

Obtains released date of DLL.

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : CamNet.Cam

Function : bool GetVersion()

**Example**

None

## 4.5 RFID (LF/HF)

### Status Return value

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

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

Enum
<pre>enum TAG_TYPE_HF {     ISO_14443_TYPE_A,     ISO_14443_TYPE_B,     ICODE_UID,     ICODE_EPC,     ICODE,     SR176,     ACTIVATE_ALL_TAG,     ISO_15693 };  enum TAG_TYPE_LF {     ACTIVATE_ALL_TAGS,     EM4x02,     EM4x05,     EM4x50,     HITAG1_S,     HITAG2,     TI_RFID_SYSTEMS };  enum RFID_TYPE {     RFID_LF = 0,     RFID_HF,     RFID_NOTHING };  enum READ_MODE {     READ_ASYNC,     READ_SYNC,     READ_CONTINUOUS };  enum STATE_SOUND</pre>

```
{  
    STATE_MSG,  
    STATE_READ,  
    STATE_WRITE  
};  
enum RESULT_MODE  
{  
    MODE_HEX,  
    MODE_DEC,  
    MODE_CHAR  
};  
enum SOUND_MODE  
{  
    SOUND_NO = 0,  
    SOUND_1,  
    SOUND_2,  
    SOUND_3,  
    VIB_1,  
    VIB_2  
};
```

## Functions for RFID (LF/HF)

Name	Description
<a href="#">RFID_Open</a>	Opens RFID.
<a href="#">RFID_Close</a>	Closes RFID
<a href="#">RFID_PowerSupply</a>	Supplies power to RFID module.
<a href="#">RFID_GetType</a>	Gets RFID Radio Type of device.
<a href="#">RFID_SelectTag</a>	Searches and Selects one tag within Antenna field.
<a href="#">RFID_HighSelectTag</a>	Searches one tag within Antenna field and selects as High baudrate.
<a href="#">RFID_LoginTag</a>	Login the tag if necessary.
<a href="#">RFID_ReadBlock</a>	Reads memory block of the tag.
<a href="#">RFID_WriteBlock</a>	Writes data in memory block of the tag.
<a href="#">RFID_ReadMultiBlock</a>	Reads multiple data blocks on a card.
<a href="#">RFID_WriteMultiBlock</a>	Writes multiple data blocks on a card.
<a href="#">RFID_SetAntenna</a>	Controls antenna power of the RFID Module.
<a href="#">RFID_GetVersion</a>	Gets DLL information and FW information of reader.
<a href="#">RFID_EnableMultiTag</a>	Enables Anti-Collision function so that multi tags can be read.
<a href="#">RFID_SendReadMultiTag</a>	Sends commander reading multi tags to reader.
<a href="#">RFID_SendTransferCommand</a>	Allows card-specific communication.
<a href="#">RFID_SendContinuousRead</a>	Reads and displays serial numbers continuously while one or more tags remain in the field.
<a href="#">RFID_SendCommand</a>	Sends a command to a reader specified by its handle
<a href="#">RFID_SendCommandGetData</a>	Sends a command to a reader specified by its handle and receives data..
<a href="#">RFID_GetData</a>	Gets the result that performs commander stored in reader.
<a href="#">RFID_CheckResult</a>	Checks obtained result from reader if it is valid.
<a href="#">RFID_SoundPlay</a>	Plays sound and vibration.
<a href="#">RFID_SetTagType</a>	Sets up the reader for a specific tag type.
<a href="#">RFID_GetTagType</a>	Gets tag type.
<a href="#">RFID_GetTagTypeToString</a>	Gets tag type as string.
<a href="#">RFID_TagItInventory</a>	Performs Inventory commander of TagIt tag.
<a href="#">RFID_TagItReadBlock</a>	Reads memory block of TagIt tag.
<a href="#">RFID_TagItWriteBlock</a>	Writes memory block of TagIt tag.

## 4.5.1 RFID\_Open

### Description

Opens RFID.

### Syntax

```
RFID_API RFID_TYPE RFID_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. (in CFReader.dll version 4.1.1)

### See Also

RFID\_PowerSupply

### For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool PowerSupply(bool bOn);

### Example

```
RFID_TYPE m_RfType = RFID_Open();
TCHAR tzRadioType[256] = {0x00};
if(m_RfType == RFID_LF)
{
    wsprintf(tzRadioType, L"Low Frequency");
}
else if(m_RfType == RFID_HF)
{
    wsprintf(tzRadioType, L"High Frequency");
}
else
{
    return;
}
```

## 4.5.2 RFID\_Close

### Description

Close RFID.

### Syntax

```
RFID_API BOOL RFID_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

RFID\_PowerSupply

### For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool Close(bool bOn);

### Example

```
RFID_Close();
```

## 4.5.3 RFID\_PowerSupply

### Description

Supplies power to RFID module.

### Syntax

```
RFID_API BOOL RFID_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 .Net

Namespace : RFIDNet.RFIDCommon

Function : bool PowerSupply(bool bOn)

#### **Example**

```
RFID_PowerSupply(FALSE);
```

### **4.5.4 RFID\_GetType**

#### **Description**

Gets RFID Radio Type of device.

#### **Syntax**

```
RFID_API RFID_TYPE RFID_GetType();
```

#### **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 .Net**

Namespace : RFIDNet.RFIDCommon

Function : RFID\_TYPE GetRadioType();

#### **Example**

```
RFID_TYPE m_RfType = RFID_GetType();
TCHAR tzRadioType[256] = {0x00};
if(m_RfType == RFID_LF)
{
    wsprintf(tzRadioType, L"Low Frequency");
}
else if(m_RfType == RFID_HF)
{
    wsprintf(tzRadioType, L"High Frequency");
}
else
{
    return;
```



}

### 4.5.5 RFID\_SelectTag

#### Description

Searches and Selects one tag within Antenna field.

#### Syntax

```
RFID_API BOOL RFID_SelectTag(LPWSTR strSerial);
```

#### Parameters

*strSerial*

Obtains serial number of selected tag.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

RFID\_HighSelectTag, RFID\_ReadBlock, RFID\_WriteBlock

#### For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool SelectTag(StringBuilder strSerial)

#### Example

```
TCHAR tzSerial[100] = {0x00};
```

```
Rfid_SelectTag(tzSerial);
```

### 4.5.6 RFID\_HighSelectTag

#### Description

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

#### Syntax

```
RFID_API BOOL RFID_HighSelectTag(LPWSTR 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

RFID\_SelectTag, RFID\_ReadBlock, RFID\_WriteBlock

#### For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool HighSelectTag(StringBuilder strSerial)

#### Example

```
TCHAR tzSerial[100] = {0x00};  
Rfid_HighSelectTag(tzSerial);
```

### 4.5.7 RFID\_LoginTag

#### Description

Login the tag if necessary.

#### Syntax

```
RFID_API TCHAR RFID_LoginTag(LPCWSTR strLogin);
```

#### Parameters

*strLogin*

Inputs password to login.

#### Return Value

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

#### Remarks

None

#### See Also

RFID\_ReadBlock, RFID\_WriteBlock

#### For .Net

Namespace : RFIDNet.RFIDCommon

Function : char LoginTag(string strLogin)

#### Example

```
TCHAR tzSerial[32] = {0x00};  
TCHAR tzData[nMaxData] = {0x00};  
memset(tzSerial, 0x00, sizeof(TCHAR) * 32);  
memset(tzData, 0x00, sizeof(TCHAR) * nMaxData);  
if(Rfid_SelectTag(tzSerial))  
{
```

```

TCHAR tRet = RFID_LoginTag(L"01AFFFFFFFFFFFFF");
if(tRet == 'L' || tRet == 'I')
{
    Rfid_ReadBlock(L"01", tzData);
}
}

```

### 4.5.8 RFID\_ReadBlock

#### Description

Reads memory block of the tag.

#### Syntax

RFID\_API BOOL RFID\_ReadBlock(LPCWSTR strBlock, LPWSTR 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

RFID\_WriteBlock

#### For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool ReadBlock(string strBlock, StringBuilder strData)

#### Example

```

TCHAR tzSerial[32] = {0x00};
TCHAR tzData[nMaxData] = {0x00};
memset(tzSerial, 0x00, sizeof(TCHAR) * 32);
memset(tzData, 0x00, sizeof(TCHAR) * nMaxData);
if(Rfid_SelectTag(tzSerial))
{
    Rfid_ReadBlock(L"01", tzData);
}

```

```
}
```

### 4.5.9 RFID\_WriteBlock

#### Description

Writes data in memory block of the tag.

#### Syntax

```
RFID_API BOOL RFID_WriteBlock(LPCWSTR strBlock, LPCWSTR strInData, LPWSTR 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

RFID\_ReadBlock, RFID\_WriteMultiBlock

#### For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool WriteBlock(string strBlock, string strInData, StringBuilder strOutData)

#### Example

```
TCHAR tzSerial[32] = {0x00};
TCHAR tzOutData[nMaxData] = {0x00};
memset(tzSerial, 0x00, sizeof(TCHAR) * 32);
memset(tzOutData, 0x00, sizeof(TCHAR) * 1024);
if(Rfid_SelectTag(tzSerial))
{
    Rfid_WriteBlock(L"01", L"00112233", tzOutData);
}
```

### 4.5.10 RFID\_ReadMultiBlock

#### Description

Reads multiple data blocks on a card.

#### Syntax

RFID\_API void RFID\_ReadMultiBlock(LPCWSTR strStartBlock, LPCWSTR strNumBlock, LPWSTR strData)

#### 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

RFID\_WriteBlock

#### For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool ReadBlock(string strBlock, StringBuilder strData)

#### Example

None

### 4.5.11 RFID\_WriteMultiBlock

#### Description

Writes multiple data blocks on a card.

#### Syntax

RFID\_API void RFID\_WriteMultiBlock(LPCWSTR strStartBlock, LPCWSTR strNumBlock, LPCWSTR strWriteData, LPWSTR strOut);

#### 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**

RFID\_ReadBlock, RFID\_ReadMultiBlock

### **For .Net**

Namespace : RFIDNet.RFIDCommon

Function : bool WriteBlock(string strBlock, string strInData, StringBuilder strOutData)

### **Example**

```
TCHAR tzSerial[32] ={0x00};
TCHAR tzOutData[nMaxData] = {0x00};
memset(tzSerial, 0x00, sizeof(TCHAR) * 32);
memset(tzOutData, 0x00, sizeof(TCHAR) * 1024);
if(Rfid_SelectTag(tzSerial))
{
    RFID_WriteMultiBlock(L"01", L"02", L"0011223344556677", tzOutData);
}
```

## **4.5.12 RFID\_SetAntenna**

### **Description**

Controls antenna power of the RFID Module.

### **Syntax**

RFID\_API void RFID\_SetAntenna(BOOL bSet);

### **Parameters**

*bSet*

Sets the condition of Antenna.

### **Return Value**

None

### **Remarks**

Main battery can be saved by turning off the antenna.

#### See Also

None

#### For .Net

Namespace : RFIDNet.RFIDCommon

Function : void SetAntenna(bool bOn)

#### Example

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

### 4.5.13 RFID\_GetVersion

#### Description

Gets DLL information and FW information of reader.

#### Syntax

```
RFID_API BOOL RFID_GetVersion(LPWSTR szFwVersion, LPWSTR szReaderDllVersion, LPWSTR szRfidDllVersion);
```

#### Parameters

*szFwVersion*

Obtains FW version of reader.

*szReaderDllVersion*

Obtains version of CFReader.dll.

*szRfidDllVersion*

Obtains version of RFID.dll.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

None

## For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool GetVersion(StringBuilder strFwVersion, StringBuilder strReaderDllVersion, StringBuilder strRfidDllVersion)

### Example

```
TCHAR tzVersion[32] = {0x00};  
TCHAR tzVersion2[32] = {0x00};  
TCHAR tzVersion3[32] = {0x00};  
RFID_GetVersion(tzVersion, tzVersion2, tzVersion3);
```

## 4.5.14 RFID\_EnableMultiTag

### Description

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

### Syntax

RFID\_API BOOL RFID\_EnableMultiTag(BOOL bEnable);

### Parameters

*bEnable*

Enables Anti-Collision function of reader.

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

RFID\_SendReadMultiTag , RFID\_GetData

## For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool EnableMultiTag(bool bEnable)

### Example

```
RFID_EnableMultiTag(TRUE);
```

## 4.5.15 RFID\_SendReadMultiTag

### Description

Sends commander reading multi tags to reader.

### Syntax

RFID\_API void RFID\_SendReadMultiTag();



**Parameters**

None

**Return Value**

None

**Remarks**

MultiTag can be read with Rfid\_SendContinuousRead function after RFID\_EnableMultiTag.

**See Also**

RFID\_EnableMultiTag, RFID\_GetData, Rfid\_SendContinuousRead

**For .Net**

Namespace : RFIDNet.RFIDCommon

Function : void SendReadMultiTag()

**Example**

None

### 4.5.16 RFID\_SendTransferCommand

**Description**

This command allows card-specific communication.

**Syntax**

RFID\_API void RFID\_SendTransferCommand(LPWSTR 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**

RFID\_GetData

**For .Net**

Namespace : RFIDNet.RFIDCommon

Function : void SendTransferCommand(string strData);

## Example

None

### 4.5.17 RFID\_SendContinuousRead

#### Description

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

#### Syntax

```
RFID_API void RFID_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

RFID\_GetData , RFID\_EnableMultiTag

#### For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool SendContinuousRead(bool bStart)

#### Example

```
RFID_SendContinuousRead(TRUE);  
BOOL bRead = TRUE;  
while(bRead)  
{  
    TCHAR szSerial[512] = {0x00,};  
    memset(szSerial, 0x00, sizeof(TCHAR) * 512);  
    RFID_GetData(szSerial);  
    if(wcslen(szSerial) > 2 && bRead)  
    {  
        // Print szSerial  
    }  
}  
RFID_SendContinuousRead(FALSE);
```

### 4.5.18 RFID\_SendCommand

#### Description

The RFID\_SendCommand function sends a command to a reader specified by its handle.

#### Syntax

```
RFID_API void RFID_SendCommand(LPCWSTR strCommand, LPCWSTR strData);
```

#### Parameters

*strCommand*

Zero terminated string with the command

*strData*

Zero terminated string containing the data

#### Return Value

None

#### Remarks

None

#### See Also

RFID\_GetData

#### For .Net

Namespace : RFIDNet.RFIDCommon

Function : void SendCommand(string strCommand, string strData)

#### Example

```
RFID_SendCommand(L"s", L"");  
BOOL bRead = TRUE;  
while(bRead)  
{  
    TCHAR szSerial[512] = {0x00,};  
    memset(szSerial, 0x00, sizeof(TCHAR) * 512);  
    RFID_GetData(szSerial);  
    if(wcslen(szSerial) > 2 && bRead)  
    {  
        // Print szSerial  
    }  
}
```

### 4.5.19 RFID\_SendCommandGetData

#### Description

The RFID\_SendCommandGetData function sends a command to a reader specified by its handle and receives data..

#### Syntax

RFID\_API void RFID\_SendCommandGetData(LPCWSTR strCommand, LPCWSTR strInputData, LPWSTR 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

RFID\_SendCommand , RFID\_GetData

#### For .Net

Namespace : RFIDNet.RFIDCommon

Function : void SendCommandGetData(string strCommand, string strInputData, StringBuilder strOutputData)

#### Example

```
TCHAR szSerial[512] = {0x00,};  
memset(szSerial, 0x00, sizeof(TCHAR) * 512);  
RFID_SendCommandGetData(L"s", L"", szSerial);
```

### 4.5.20 RFID\_GetData

#### Description

Gets the result that performs commander stored in reader.

#### Syntax

RFID\_API BOOL RFID\_GetData(LPWSTR 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

RFID\_SendReadMultiTag, RFID\_SendTransferCommand, RFID\_SendContinuousRead

### For .Net

Namespace : RFIDNet.RFIDCommon

Function : bool GetData(StringBuilder strData)

### Example

```
RFID_SendContinuousRead(TRUE);  
BOOL bRead = TRUE;  
while(bRead)  
{  
    TCHAR szSerial[512] = {0x00,};  
    memset(szSerial, 0x00, sizeof(TCHAR) * 512);  
    RFID_GetData(szSerial);  
    if(wcslen(szSerial) > 2 && bRead)  
    {  
        // Print szSerial  
    }  
}  
RFID_SendContinuousRead(FALSE);
```

## 4.5.21 RFID\_CheckResult

### Description

Checks obtained result from reader if it is valid.

### Syntax

RFID\_API BOOL RFID\_CheckResult(TCHAR\* tzInputResult, TCHAR\* tzOutputResult)

### Parameters

*tzInputResult*

Inputs result obtained from reader.

*tzOutputResult*

Obtains message if it is valid result from reader.

### **Return Value**

Nonzero indicates success. Zero indicates failure.

### **Remarks**

None

### **See Also**

RFID\_GetData

### **For .Net**

Namespace : RFIDNet.RFIDCommon

Function : bool CheckResult(string strInputResult, StringBuilder strOutputResult);

### **Example**

```
TCHAR tzSerial[32] = {0x00};
TCHAR tzData[nMaxData] = {0x00};
memset(tzSerial, 0x00, sizeof(TCHAR) * 32);
memset(tzData, 0x00, sizeof(TCHAR) * nMaxData);
if(Rfid_SelectTag(tzSerial))
{
    Rfid_ReadBlock(L"01", tzData);
}
TCHAR tzMsg[1024] = {0x00};
memset(tzMsg, 0x00, sizeof(TCHAR) * 1024);
if(RFID_CheckResult(tzOutData, tzMsg))
{
    // "Result: Write Success"
    // print tzMsg
}
```

## **4.5.22 RFID\_SoundPlay**

### **Description**

Plays sound and vibration.

### **Syntax**

RFID\_API BOOL RFID\_SoundPlay(SOUND\_MODE Sound)

### **Parameters**

*Sound*

SOUND\_MODE is enum value type. Plays sound and vibration.

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : RFIDNet.RFIDCommon

Function : bool SoundPlay(SOUND\_MODE sound)

**Example**

RFID\_SoundPlay(SOUND\_1);.

### 4.5.23 RFID\_SetTagType

**Description**

This Function sets up the reader for a specific tag type.

**Syntax**

RFID\_API int RFID\_SetTagType(int nType)

**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**

RFID\_GetTagType, RFID\_GetTagTypeToString

**For .Net**

Namespace : RFIDNet.RFIDCommon

Function : int SetTagType(int Type)

**Example**

RFID\_SetTagType(ISO\_14443\_TYPE\_B);.

## 4.5.24 RFID\_GetTagType

### Description

Gets tag type.

### Syntax

```
RFID_API int RFID_GetTagType();
```

### Parameters

None

### Return Value

TAG\_TYPE\_HF is applied to HF RFID and TAG\_TYPE\_LF is applied to LF RFID.

### Remarks

None

### See Also

RFID\_GetTagTypeToString, RFID\_SetTagType

### For .Net

Namespace : RFIDNet.RFIDCommon

Function : int GetTagType();

### Example

```
int nType = GetTagType();
```

## 4.5.25 RFID\_GetTagTypeToString

### Description

Gets tag type as string.

### Syntax

```
RFID_API void RFID_GetTagTypeToString(TCHAR* tzTagType);
```

### Parameters

*tzTagType*

Obtains current TagType as string.

### Return Value

None

### Remarks

None

### See Also

RFID\_SetTagType, RFID\_GetTagType

### For .Net



Namespace : RFIDNet.RFIDCommon

Function : void GetTagTypeToString(StringBuilder strTagType)

#### Example

```
TCHAR tzType[260] = {0x00};  
RFID_GetTagTypeToString(tzType);.
```

### 4.5.26 RFID\_TagItInventory

#### Description

Performs Inventory commander of TagIt tag.

#### Syntax

RFID\_API BOOL RFID\_TagItInventory(TCHAR\* tzUID)

#### Parameters

*tzUID*

Obtains UID of tag.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

RFID\_TagItReadBlock, RFID\_TagItWriteBlock

#### For .Net

Namespace : RFIDNet. RFIDTagIt

Function : bool Inventory(StringBuilder strUID);

#### Example

```
TCHAR tzInventory[256] = {0x00};  
memset(tzInventory, 0x00, sizeof(TCHAR) * 256);  
if(!RFID_TagItInventory(tzInventory))  
    return;  
  
TCHAR tzResult[256] = {0x00};  
memset(tzResult, 0x00, sizeof(TCHAR) * 256);  
RFID_TagItReadBlock(1, tzResult);.
```

### 4.5.27 RFID\_TagItReadBlock

#### Description

Reads memory block of TagIt tag.

### Syntax

```
RFID_API BOOL RFID_TagItReadBlock(int nBlockNo, TCHAR* tzBlockData);
```

### 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

RFID\_TagItInventory, RFID\_TagItWriteBlock

### For .Net

Namespace : RFIDNet. RFIDTagIt

Function : bool ReadBlock(int nBlockNo, StringBuilder strBlockData)

### Example

```
TCHAR tzInventory[256] = {0x00};
memset(tzInventory, 0x00, sizeof(TCHAR) * 256);
if(!RFID_TagItInventory(tzInventory))
    return;

TCHAR tzResult[256] = {0x00};
memset(tzResult, 0x00, sizeof(TCHAR) * 256);
RFID_TagItReadBlock(1, tzResult);.
```

## 4.5.28 RFID\_TagItWriteBlock

### Description

Writes memory block of TagIt tag.

### Syntax

```
RFID_API BOOL RFID_TagItWriteBlock(int nBlockNo, CONST TCHAR* tzWriteBlockData, TCHAR*
tzWriteResult)
```

### 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**

RFID\_TagItInventory, RFID\_TagItReadBlock

### **For .Net**

Namespace : RFIDNet. RFIDTagIt

Function : bool WriteBlock(int nBlockNo, string strWriteBlockData, StringBuilder strWriteResult)

### **Example**

```
TCHAR tzInventory[256] = {0x00};
memset(tzInventory, 0x00, sizeof(TCHAR) * 256);
if(!RFID_TagItInventory(tzInventory))
    return;

TCHAR tzResult[256] = {0x00};
memset(tzResult, 0x00, sizeof(TCHAR) * 256);
RFID_TagItWriteBlock(1, L"00112233",tzResult);.
```

## 4.6 UHF GUN READER

### Status Return value

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

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

Define
<pre>typedef unsigned __int8  INT8U; typedef unsigned __int16 INT16U; typedef unsigned __int32 INT32U; typedef signed  __int32  BOOL32; typedef unsigned __int32 HANDLE32; typedef HANDLE32  RFID_RADIO_HANDLE;  #define WM_MSG_INVENTORY  0x5001 #define WM_MSG_ACCESS    0x5002 #define WM_MSG_POWER     0x5010  #define UHF_OPEN_EVENT    L"UHF_OPEN_EVENT"</pre>
Enum
<pre>typedef enum {     BATTERY_ERROR,     BATTERY_0,     BATTERY_1,     BATTERY_2,     BATTERY_3,     BATTERY_FULL }BATTERY_STATUS;  typedef enum {     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, }RFIDErrorCode;  typedef enum {     TAG_RESERVED,     TAG_EPC,     TAG_TID,     TAG_USER }RFIDTagBank;</pre>

```
typedef enum
```

```
{  
    RFID_REGION_KOREA_NEW,  
    RFID_REGION_KOREA_WEAK,  
    RFID_REGION_KOREA_OLD,  
    RFID_REGION_USA,  
    RFID_REGION_EURO,  
    RFID_REGION_EURO_NEW,  
    RFID_REGION_JAPAN,  
    RFID_REGION_CHINA,  
    RFID_REGION_AUSTRALIA,  
    RFID_REGION_BRAZIL,  
    RFID_REGION_MALAYSIA,  
    RFID_REGION_TAIWAN  
}
```

```
}RFIDRegion;
```

```
typedef enum
```

```
{  
    SELECT_EPC = 1,  
    SELECT_TID = 2,  
    SELECT_USER = 3  
}
```

```
}RFIDSelectBank;
```

```
typedef enum
```

```
{  
    PERMISSION_ACCESSIBLE,  
    PERMISSION_ALWAYS_ACCESSIBLE,  
    PERMISSION_SECURED_ACCESSIBLE,  
    PERMISSION_ALWAYS_NOT_ACCESSIBLE,  
    PERMISSION_NO_CHANGE  
}
```

```
}RFIDLockPermission;
```

```
enum
```

```
{  
    /* 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 */
}typedef RFID_STATUS;

```

## Structure

```

typedef struct {
    /* The major version (i.e., in 1.x.x.x, the 1) */
    INT32U major;
    /* The minor version (i.e., in x.1.x.x, the 1) */
    INT32U minor;
    /* The maintenance number (i.e., in x.x.1.x, the 1) */
    INT32U maintenance;
    /* The release number (i.e., in x.x.x.1, the 1) */
    INT32U release;
} RFID_VERSION;

typedef struct
{
    HWND hWnd; //Diag handle Receive for Message
    RFIDTagBank bank;
    INT8U wlength;
    INT8U offset;
    INT32U accpwd;
}RFIDReadCmd;

typedef struct
{
    HWND hWnd; //Diag handle Receive for Message
    RFIDTagBank bank;
    INT8U wlength;
    INT8U offset;
    INT16U wdata[32];
    INT32U accpwd;
}RFIDWriteCmd;

typedef struct
{

```

```
HWND hWnd; //Diag handle Receive for Message
RFIDLockPermission lockkillpwd;
RFIDLockPermission lockaccesspwd;
RFIDLockPermission lockepc;
RFIDLockPermission locktid;
RFIDLockPermission lockuser;
INT32U accpwd;
}RFIDLockCmd;

typedef struct
{
    HWND hWnd; //Diag handle Receive for Message
    INT32U killpwd;
}RFIDKillCmd;
```

## Functions for UHF GUN READER

Name	Description
<a href="#">UHF_GetError</a>	Check the last error
<a href="#">UHF_IsReady</a>	Check UHF GUN READER's availability by checking the power
<a href="#">UHF_Init</a>	Initialize RFID
<a href="#">UHF_Close</a>	Close RFID
<a href="#">UHF_Inventory</a>	Start Inventory by reading EPC data
<a href="#">UHF_InventoryStop</a>	Stop Inventory
<a href="#">UHF_Read</a>	Read memory from tag
<a href="#">UHF_Write</a>	Write data to tag
<a href="#">UHF_Lock</a>	Limit access to tag
<a href="#">UHF_Kill</a>	Kill the tag (disabling the tag)
<a href="#">UHF_GetData</a>	Get tag data from Inventory and Read
<a href="#">UHF_SetRegionFrequency</a>	Set RFID frequency to suit regional regulation
<a href="#">UHF_GetRegionFrequency</a>	Check current regional frequency setting
<a href="#">UHF_ReadBattery</a>	Check battery voltage and ADC value
<a href="#">UHF_ReadBatteryStatus</a>	Check battery level (0~4 level)
<a href="#">UHF_Version</a>	Check DLL and F/W version



### 4.6.1 UHF\_GetError

#### Description

Check the last error

#### Syntax

```
RFIDErrorCode UHF_GetError();
```

#### Parameters

None

#### Return Value

Retuens RFIDErrorCode of enum type

#### Remarks

None

#### See Also

None

#### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFIDErrorCode GetError();

#### Example

```
length = UHF_GetData(data);
if(length == 0)
{
    err = UHF_GetError();
    switch(err)
    {
        case TAG_OTHERERROR:
            MessageBox(_T("Tag Other Error!"));
            break;
        case TAG_SUCCESS:
            MessageBox(_T("SUCCESS!"));
            break;
        case TAG_MEMORYOVERRUN:
            MessageBox(_T("Tag Memory Overrun!"));
            break;
        case MAC_OUTOFRETRIES:
            MessageBox(_T("Out of Retries Error!"));
```

```

        break;
    default:
        MessageBox(_T("Unknown Error!"));
        break;
    }
}

```

## 4.6.2 UHF\_IsReady

### Description

Check UHF GUN READER's availability by checking the power

### Syntax

```
BOOL UHF_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 .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : bool IsReady();

### Example

```

if(!UHF_IsReady())
    return;

```

## 4.6.3 UHF\_Init

### Description

Initialize RFID

### Syntax

```
RFID_STATUS UHF_Init (HWND hDlgWnd);
```

### Parameters

*hDlgWnd*

Reader power status message is passed to registered Windows Handle

### Return Value

Returns RFID\_STATUS of enum type

### Remarks

When initializing, WM\_MSG\_POWER message is passed to registered Windows Handle. On change of power status, this message will return the status. If FALSE is returned, reader should be closed using UHF\_Close. If it changes to TRUE, then UHF\_Init should be used to re-initialize.

### See Also

UHF\_Close

### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFID\_STATUS Init();

### Example

```
RFID_STATUS status = UHF_Init(m_hWnd);  
if(status != RFID_STATUS_OK)  
{  
    strstatus.Format(_T("RFID Init Error-[%x]"),status);  
    return FALSE;  
}
```

## 4.6.4 UHF\_Close

### Description

Close RFID

### Syntax

```
RFID_STATUS UHF_Close();
```

### Parameters

None

### Return Value

Returns RFID\_STATUS of enum type

### Remarks

None

### See Also

UHF\_Init

### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFID\_STATUS Close();

### Example

```
RFID_STATUS status;  
CString strstatus;  
status = UHF_Close();  
if(status != RFID_STATUS_OK)  
{  
    strstatus.Format(_T("%x"),status);  
    return FALSE;  
}
```

## 4.6.5 UHF\_Inventory

### Description

Start Inventory by reading EPC data

### Syntax

```
void UHF_Inventory(HWND hWnd);
```

### Parameters

*hWnd*

Register Windows Handle to receive data through inventory

### Return Value

None

### Remarks

EPC data is passed to registered Windows Handle through WM\_MSG\_INVENTORY

### See Also

UHF\_InventoryStop

### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : void Inventory();

### Example

```
UHF_Inventory(m_hWnd);
```

#### 4.6.6 UHF\_InventoryStop

##### Description

Stop Inventory

##### Syntax

```
void UHF_InventoryStop();
```

##### Parameters

None

##### Return Value

None

##### Remarks

None

##### See Also

UHF\_Inventory

##### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : void InventoryStop();

Example

```
UHF_InventoryStop();
```

#### 4.6.7 UHF\_Read

##### Description

Read memory from tag

##### Syntax

```
RFID_STATUS UHF_Read(RFIDReadCmd *cmd);
```

##### Parameters

*cmd*

RFIDReadCmd structure is used to assign tag memory location

##### Return Value

Returns RFID\_STATUS of enum type

##### Remarks

Windows Handle for receiving data reading message must be registered to hWnd of RFIDReadCmd structure. Data can be obtained using UHF\_GetData once WM\_MSG\_ACCESS message is received.

##### See Also

UHF\_GetData

#### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFID\_STATUS Read(ref RFIDUHF.RFIDReadCmd cmd);

#### Example

```
RFID_STATUS status;
RFIDReadCmd readcmd;
INT32U accpwd = 0;
accpwd = wcstoul(m_strRWPwd, 0, 16);
readcmd.hWnd = this->m_hWnd;
readcmd.bank = (RFIDTagBank)m_cmbBank.GetCurSel();
readcmd.wlength = (INT8U)_ttoi(m_strLength);
readcmd.offset = (INT8U)_ttoi(m_strOffset);
readcmd.accpwd = accpwd;
status = UHF_Read(&readcmd);
if (RFID_STATUS_OK != status)
{
    // Fail!!
}
```

### 4.6.8 UHF\_Write

#### Description

Write data to tag

#### Syntax

RFID\_STATUS UHF\_Write(RFIDWriteCmd \*cmd);

#### Parameters

*cmd*

RFIDWriteCmd structure is used to assign tag memory location

#### Return Value

Returns RFID\_STATUS of enum type

#### Remarks

Windows Handle for receiving data reading message must be registered to hWnd of RFIDWriteCmd structure. Result can be obtained using UHF\_GetError once WM\_MSG\_ACCESS message is received.

#### See Also

UHF\_GetError

## For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFID\_STATUS Write(ref RFIDUHF.RFIDWriteCmd cmd);

## Example

```
RFID_STATUS status;
RFIDWriteCmd writecmd;
WCHAR *str;
char str1[32][8] = {0,};
INT16U wdata[32] = {0,};
INT8U wlength;
INT8U offset;
INT32U accpwd = 0;
wlength = (INT8U)_ttoi(L"6");
offset = (INT8U)_ttoi(L"2");
accpwd = wcstoul(L"00000000", 0, 16);
str = (WCHAR*)((LPCWSTR)m_strWriteData); // 6 Word – 12 Byte

for(int i=0;i<wlength;i++)
{
    sprintf(str1[i], "%c%c%c%c", str[i*4], str[i*4+1], str[i*4+2], str[i*4+3]);
    wdata[i] = (INT16U)strtoul(str1[i], 0, 16);
}

writecmd.hWnd = this->m_hWnd;
writecmd.bank = (RFIDTagBank)m_cmbBank.GetCurSel();
writecmd.wlength = wlength;
writecmd.offset = offset;
memcpy(writecmd.wdata, wdata, wlength*2);
writecmd.accpwd = accpwd;

status = UHF_Write(&writecmd);
if (RFID_STATUS_OK != status)
{
    MessageBox(_T("RFID Write Fail!"));
```

}

### 4.6.9 UHF\_Lock

#### Description

Limit access to tag

#### Syntax

```
RFID_STATUS UHF_Lock(RFIDLockCmd *cmd);
```

#### Parameters

*cmd*

RFIDLockCmd structure is used to limit access to tag

#### Return Value

Returns RFID\_STATUS of enum type

#### Remarks

Windows Handle for receiving data reading message must be registered to hWnd of RFIDLockCmd structure. Result can be obtained using UHF\_GetError once WM\_MSG\_ACCESS message is received.

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

UHF\_GetError

#### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFID\_STATUS Lock(ref RFIDUHF.RFIDLockCmd cmd);

#### Example

```
RFID_STATUS status;
```

```
RFIDLockCmd lock;
```

```
INT32U accpwd;
```



```

accpwd = wcstoul(m_strLockPwd, 0, 16);

lock.hWnd = this->m_hWnd;
lock.lockkillpwd = (RFIDLockPermission)m_cmbKillPwd.GetCurSel();
lock.lockaccesspwd = (RFIDLockPermission)m_cmbAccPwd.GetCurSel();
lock.lockepc = (RFIDLockPermission)m_cmbEPC.GetCurSel();
lock.locktid = (RFIDLockPermission)m_cmbTID.GetCurSel();
lock.lockuser = (RFIDLockPermission)m_cmbUser.GetCurSel();
lock.accpwd = accpwd;

status = UHF_Lock(&lock);

if(RFID_STATUS_OK != status)
{
    MessageBox(_T("RFID Lock Fail!"));
}

```

#### 4.6.10 UHF\_Kill

##### Description

Kill the tag (disabling the tag)

##### Syntax

```
RFID_STATUS UHF_Kill(RFIDKillCmd *cmd);
```

##### Parameters

*cmd*

RFIDKillCmd structure is used to disable tag

##### Return Value

Returns RFID\_STATUS of enum type

##### Remarks

Windows Handle for receiving data reading message must be registered to hWnd of RFIDKillCmd structure. Result can be obtained using UHF\_GetError once WM\_MSG\_ACCESS message is received.

##### See Also

UHF\_GetError

##### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFID\_STATUS Kill(ref RFIDUHF.RFIDKillCmd cmd);

#### Example

```
RFID_STATUS status;
RFIDKillCmd kill;
INT32U killpwd;
killpwd = wcstoul(m_strKillPwd, 0, 16);
kill.hWnd = this->m_hWnd;
kill.killpwd = killpwd;
status = UHF_Kill(&kill);
if(RFID_STATUS_OK != status)
{
    MessageBox(_T("RFID Kill Fail!"));
}
```

### 4.6.11 UHF\_GetData

#### Description

Get tag data from Inventory and Read

#### Syntax

```
int UHF_GetData (INT8U *data);
```

#### Parameters

*\*data*

[out] get current data

#### Return Value

Returns the length of the data

#### Remarks

Data obtained by UHF\_Inventory or UHF\_Read can be checked. If return value is 0, error can be checked using UHF\_GetError.

#### See Also

UHF\_Inventory, UHF\_Read, UHF\_GetError

#### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : int GetData(StringBuilder strTagData);

#### Example

```
int length = 0;
unsigned char data[128] = {0,};
```

```

RFIDErrorCode err;
CString str;
length = UHF_GetData(data);
if(length == 0)
{
    err = UHF_GetError();

}
else
{
    for(int i=0;i<length;i++)
    {
        str.AppendFormat(_T("%02X"), data[i]);
    }
    m_strReadData = str;
}

```

#### 4.6.12 UHF\_SetRegionFrequency

##### Description

Set RFID frequency to suit regional regulation

##### Syntax

```
RFID_STATUS UHF_SetRegionFrequency(RFIDRegion region);
```

##### Parameters

*region*

Set regional frequency by assigning Enum type variable

##### Return Value

Returns RFID\_STATUS of enum type

##### Remarks

None

##### See Also

UHF\_GetRegionFrequency

##### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFID\_STATUS SetRegionFrequency(RFIDUHF.RFIDRegion region);

### Example

```
RFIDRegion region = RFID_REGION_EURO_NEW;  
RFID_STATUS status = UHF_SetRegionFrequency(region);  
if(status != RFID_STATUS_OK)  
{  
    MessageBox(_T("UHF_SetRegionFrequency Fail!"));  
    return ;  
}
```

## 4.6.13 UHF\_GetRegionFrequency

### Description

Check current regional frequency setting

### Syntax

```
RFIDRegion UHF_GetRegionFrequency();
```

### Parameters

None

### Return Value

Returns RFID\_STATUS of enum type

### Remarks

None

### See Also

UHF\_SetRegionFrequency

### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFIDRegion GetRegionFrequency();

### Example

```
RFIDRegion region = UHF_GetRegionFrequency();
```

## 4.6.14 UHF\_ReadBattery

### Description

Check battery voltage and ADC value

### Syntax

```
RFID_STATUS UHF_ReadBattery(INT32U *nADCValue, 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

UHF\_ReadBatteryStatus

### For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFID\_STATUS ReadBattery(ref uint nADCValue, ref float fVolt);

### Example

```
RFID_STATUS status;
```

```
INT32U nADC = 0;
```

```
float fVolt;
```

```
BATTERY_STATUS battstatus;
```

```
UHF_ReadBattery(&nADC, &fVolt);
```

```
status = UHF_ReadBatteryStatus(&battstatus);
```

```
m_strStatus.Format(_T("%.2f(%d)"),fVolt, battstatus);
```

## 4.6.15 UHF\_ReadBatteryStatus

### Description

Check battery level (0~4 level)

### Syntax

```
RFID_STATUS UHF_ReadBatteryStatus(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

UHF\_ReadBattery

## For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : RFIDUHF.RFID\_STATUS ReadBatteryStatus(ref RFIDUHF.BATTERY\_STATUS step);

## Example

```
RFID_STATUS status;
INT32U nADC = 0;
float fVolt;
BATTERY_STATUS battstatus;
UHF_ReadBattery(&nADC, &fVolt);
status = UHF_ReadBatteryStatus(&battstatus);
m_strStatus.Format(_T("%.2f(%d)"),fVolt, battstatus);
```

## 4.6.16 UHF\_Version

### Description

Check DLL and F/W version

### Syntax

```
BOOL UHF_Version(RFID_VERSION *LibVer, RFID_VERSION *MacVer, TCHAR *tzDllVersion);
```

### Parameters

*\*LibVer*

[out] Check version of NRFMCE.dll

*\*MacVer*

[out] Check reader firmware version

*\*tzDllVersion*

[out] Check RFDI\_UHF.dll version

### Return Value

TRUE = Success

FALSE = Fail

### Remarks

None

## See Also

None

## For .Net

Namespace : RFID\_UHF\_Net.RFIDUHF

Function : bool Version(ref RFIDUHF.RFID\_VERSION LibVer, ref RFIDUHF.RFID\_VERSION MacVer, StringBuilder strDllVersion);

### **Example**

```
RFID_VERSION LibVer;
```

```
RFID_VERSION MacVer;
```

```
TCHAR tzDllVersion[260] = {0x00};
```

```
CString strstatus;
```

```
UHF_Version(&LibVer, &MacVer, tzDllVersion);
```

```
strstatus.Format(_T("Firmware: %u.%u.%u APP: %s"), MacVer.major, MacVer.minor, MacVer.maintenance, VER_APP);
```

## 4.7 WLAN

### Status Return value

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

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

Define
#define MAX_AP 30
Enum
<pre>typedef enum {     SUCCESS=0,     FAIL,     INVALID_NAME,     INVALID_CONFIG,     INVALID_DELETE,     POWERCYCLE_REQUIRED,     INVALID_PARAMETER,     INVALID_EAP_TYPE,     INVALID_WEP_TYPE,     INVALID_FILE }WLAN_ERROR_RESULT;</pre>
Structure
<pre>typedef struct _WLAN_STATUS {     DWORD      channel;     int        rssi;     double     bitRate;     int        txPower;     DWORD      DTIM;     DWORD      beaconPeriod;     DWORD      beaconsReceived;     TCHAR      SSID[32];     TCHAR      CardState[36];     byte       client_MAC[6];     byte       client_IP[4];     byte       AP_MAC[6];     byte       AP_IP[4]; }WLAN_STATUS, *P_WLAN_STATUS;</pre>



```
typedef struct _WLAN_SSID
{
    TCHAR    SSID[32];
    BOOL     Privacy;
    int      Rssi;
} WLAN_SSID;

typedef struct _WLAN_SSID_LIST
{
    int      m_NumberOfItems;
    WLAN_SSID*  m_SSID;
} WLAN_SSID_LIST;

typedef struct _WLAN_CONFIG_NAME
{
    TCHAR    ConfigName[32];
} WLAN_CONFIG_NAME;

typedef struct _WLAN_CONFIG_NAME_LIST
{
    int      m_NumberOfItems;
    WLAN_CONFIG_NAME*  m_ConfigName;
}WLAN_CONFIG_NAME_LIST;
```

## Functions for WLAN

Name	Description
<a href="#">WLAN_ActivateConfig</a>	Activate the configuration with the given name.
<a href="#">WLAN_Close</a>	Free WLAN.dll.
<a href="#">WLAN_ConnectAP</a>	Connecting to AP.
<a href="#">WLAN_DeleteConfig</a>	This function deletes the configuration matching 'name'.
<a href="#">WLAN_ExportConfig</a>	This function exports configurations.
<a href="#">WLAN_ImportConfig</a>	This function imports configurations.
<a href="#">WLAN_Init</a>	WLAN.dll initiation
<a href="#">WLAN_GetAllConfigName</a>	This function retrieves all of the configurations.
<a href="#">WLAN_GetCurrentAPInfo</a>	Get current AP information
<a href="#">WLAN_GetBssidList</a>	Scans and lists connectable APs.
<a href="#">WLAN_GetPowerStatus</a>	Get power status.
<a href="#">WLAN_PowerOn</a>	Inserts WLAN card.
<a href="#">WLAN_PowerOff</a>	Removes WLAN card.

### 4.7.1 WLAN\_ActivateConfig

#### Description

Activate the configuration with the given name.

#### Syntax

```
WLAN_API BOOL WLAN_ActivateConfig(TCHAR* szName);
```

#### Parameters

*szName*

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 .Net

Namespace : WLANNet.WLANNet

Function : bool ActivateConfig(string ConfigName)

#### Example

```
if(WLAN_ActivateConfig("Config Name") == FALSE)
{
    AfxMessageBox(_T("Fail To WLAN_ActivateConfig()"));
}
```

### 4.7.2 WLAN\_Close

#### Description

Free WLAN module.

#### Syntax

```
WLAN_API BOOL WLAN_Close();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

## See Also

WLAN\_Init

## For .Net

Namespace : WLANNet.WLANNet

Function : bool Close()

## Example

```
if(WLAN_Close() == FALSE)
{
    AfxMessageBox(_T("Fail To WLAN_Close()");
}
```

## 4.7.3 WLAN\_ConnectAP

### Description

Connecting to AP.

### Syntax

WLAN\_API int WLAN\_ConnectAP(TCHAR\* szBssid, TCHAR\* 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

Return WLAN\_ERROR\_RESULT

### Remarks

None

## See Also

None

## For .Net

Namespace : WLANNet.WLANNet

Function : bool ConnectAP(string szBssid, string password, int encryptionType)

### Example

```
int result=WLAN_ConnectAP(_T("SSID NAME"), _T("Password"), EncryptionType);  
if(result==0)  
    AfxMessageBox(_T("Success To WLAN_ConnectAP()"));
```

## 4.7.4 WLAN\_ConnectAPEX

### Description

Connecting to AP.

### Syntax

WLAN\_API int WLAN\_ConnectAPEX(TCHAR\* szBssid, TCHAR\* 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

Return WLAN\_ERROR\_RESULT

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : WLANNet.WLANNet

Function : bool ConnectAPEX(string szBssid, string password, int encryptionType, int WepKeyType)

**Example**

```
int result=WLAN_ConnectAPEX(_T("SSID NAME"), _T("Password"), EncryptionType, 2);  
if(result==0)  
    AfxMessageBox(_T("Success To WLAN_ConnectAPEX()"));
```

### 4.7.5 WLAN\_DeleteConfig

**Description**

This function deletes the configuration matching 'name'.

**Syntax**

```
WLAN_API BOOL WLAN_DeleteConfig(TCHAR *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**

None

**For .Net**

Namespace : WLANNet.WLANNet

Function : bool ActivateConfig(string ConfigName)

**Example**

```
If(WLAN_DeleteConfig(_T("SSID NAME"))==0)  
    AfxMessageBox(_T("Fail To WLAN_ DeleteConfig ( )"));
```

### 4.7.6 WLAN\_ExportConfig

#### Description

This function exports configurations.

#### Syntax

```
WLAN_API BOOL WLAN_ExportConfig(TCHAR* szExportPath);
```

#### Parameters

*szExportPath*

File name and route to be stored.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

WLAN\_ImportConfig

#### For .Net

Namespace : WLANNet.WLANNet

Function : bool ExportConfig (string ExportPath)

#### Example

```
If(WLAN_ExportConfig(_T("Flash Disk\\WLAN\\Summit.sdc"))==0)
    AfxMessageBox(_T("Fail To WLAN_ExportConfig ("));
```

### 4.7.7 WLAN\_ImportConfig

#### Description

This function imports configurations.

#### Syntax

```
WLAN_API BOOL WLAN_ImportConfig(TCHAR* szImportPath);
```

#### Parameters

*szImportPath*

File name and route to get.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

WLAN\_ExportConfig

#### **For .Net**

Namespace : WLANNet.WLANNet

Function : bool ImportConfig (string ImportPath)

#### **Example**

```
If(WLAN_ImportConfig(_T("Flash Disk\\WLAN\\Summit.sdc")==0)
    AfxMessageBox(_T("Fail To WLAN_ ImportConfig (")");
```

### **4.7.8 WLAN\_Init**

#### **Description**

WLAN module initiation.

#### **Syntax**

WLAN\_API BOOL WLAN\_Init();

#### **Parameters**

None

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

WLAN\_Close

#### **For .Net**

Namespace : WLANNet.WLANNet

Function : bool Init ()

#### **Example**

```
If(WLAN_Init()==0)
    AfxMessageBox(_T("Fail To WLAN_ Init());
```

### **4.7.9 WLAN\_GetAllConfigName**

#### **Description**

This function retrieves all of the configurations.

#### **Syntax**

WLAN\_API int WLAN\_GetAllConfigName(WLAN\_CONFIG\_NAME\_LIST\* pstConfigList);

#### **Parameters**



WLAN\_CONFIG\_NAME\_LIST\* pstConfigList.

#### Return Value

Return WLAN\_ERROR\_RESULT

#### Remarks

Except ThirdPartyConfig.

#### See Also

None

#### For .Net

Namespace : WLANNet.WLANNet

Function : int GetAllConfigName ()

#### Example

```
WLAN_CONFIG_NAME_LIST* NameList=new WLAN_CONFIG_NAME_LIST();  
int resut=WLAN_GetAllConfigName(NameList);  
if(result==0)  
    AfxMessageBox(_T("Success To WLAN_GetAllConfigName()"));
```

### 4.7.10 WLAN\_GetCurrentAPInfo

#### Description

Get current AP information except txPower .

#### Syntax

```
WLAN_API int WLAN_GetCurrentAPInfo(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 .Net

Namespace : WLANNet.WLANNet

Function : int GetCurrentAPInfo (ref WLAN\_STATUS st)

#### Example

```

WLAN_STATUS st;
memset(&st, 0, sizeof(st));
int result=WLAN_GetCurrentAPInfo(&st);
if(result==0)
    AfxMessageBox(_T("Success To WLAN_GetCurrentAPInfo"));

```

#### 4.7.11 WLAN\_GetBssidList

##### Description

Scans and lists connectable APs.

##### Syntax

```
WLAN_API BOOL WLAN_GetBssidList(WLAN_SSID_LIST* pstSsidList);
```

##### Parameters

*WLAN\_SSID\_LIST\* pstSsidList*

Pointer to a WLAN\_SSID\_LIST structure to be filled in SSIDs.

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

None

##### For .Net

Namespace : WLANNet.WLANNet

Function : int GetBssidList (ref WLAN\_SSID[] ssidlist)

##### Example

```

WLAN_SSID_LIST* SSIDList=new WLAN_SSID_LIST();
SSIDList->m_SSID=new WLAN_SSID[40];
if(WLAN_GetBssidList(SSIDList)==0)
    AfxMessageBox(_T("Fail To WLAN_GetBssidList()");

```

#### 4.7.12 WLAN\_GetPowerStatus

##### Description

Get power status.

##### Syntax

```
WLAN_API BOOL WLAN_GetPowerStatus();
```

**Parameters**

None

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : WLANNet.WLANNet

Function : int GetPowerStatus ()

**Example**

```
if(WLAN_GetPowerStatus())  
    AfxMessageBox(_T("Power On!"));  
else  
    AfxMessageBox(_T("Power On!"));
```

### 4.7.13 WLAN\_PowerOn

**Description**

Inserts WLAN card.

**Syntax**

```
WLAN_API BOOL WLAN_PowerOn();
```

**Parameters**

None

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

WLAN\_PowerOff

**For .Net**

Namespace : WLANNet.WLANNet

Function : bool PowerOn ()

**Example**

```
If(WLAN_PowerOn()==0)
    AfxMessageBox(_T("Fail To WLAN_PowerOn()"));
```

#### 4.7.14 WLAN\_PowerOff

##### Description

Removing WLAN card.

##### Syntax

```
WLAN_API BOOL WLAN_PowerOff();
```

##### Parameters

None

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

WLAN\_PowerOn

##### For .Net

Namespace : WLANNet.WLANNet

Function : bool PowerOff ()

##### Example

```
If(WLAN_PowerOff()==0)
    AfxMessageBox(_T("Fail To WLAN_PowerOff()"));
```

## 4.8 BLUETOOTH

### Status Return value & API Error Codes

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

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

Define
<pre>// Message Queue Commands #define PING 0 #define BTP_FIND_FIRST_DEVICE          1 #define BTP_FIND_NEXT_DEVICE          2 #define BTP_FIND_DEVICE_CLOSE         3 #define BTP_FIND_FIRST_SERVICE        4 #define BTP_FIND_NEXT_SERVICE        5 #define BTP_FIND_SERVICE_CLOSE        6 #define BTP_FIND_FIRST_CONNECTION     7 #define BTP_FIND_NEXT_CONNECTION     8 #define BTP_FIND_CONNECTION_CLOSE     9 #define BTP_CREATE_CONNECTION        10 #define BTP_DELETE_CONNECTION        11 #define BTP_CONNECT                  12 #define BTP_DISCONNECT               13 #define BTP_SET_AUTHENTICATION_CALLBACK 14 #define BTP_SET_CONNECTION_CALLBACK   15 #define BTP_FIND_LOCAL_DEVICE         16 #define BTP_SET_INCOMING_PIN          17 #define BTP_SET_OUTGOING_PIN          18 #define BTP_PERFORM_ACTION            19 #define BTP_SET_SECURITY_MODE         20 #define BTP_GET_SECURITY_MODE         21 #define BTP_SET_SCO_CONNECTION_STATE  22 #define BTP_GET_SCO_CONNECTION_STATE  23 // Maximum message size allowed by the message queues. #define MAX_MSG_SIZE      1024 #define MAX_NAME_LENGTH   248 // Maximum name length, including delimiter. #define BLUETOOTH_MAX_NAME_SIZE      (MAX_NAME_LENGTH + 1) // Error codes. #define BTP_ERROR      0// Indicates the command executed successfully. #define BTP_ERROR_SUCCESS      1// Indicates that the command was unsuccessful; used when no other more specific error code is applicable. #define BTP_ERROR_INVALID_PARAMETER      2// One of the parameters is not valid.</pre>

<pre> #define BTP_ERROR_INVALID_HANDLE      3// A handle is not valid. #define BTP_ERROR_NO_MORE             4// Indicates there are no more elements in the list. #define BTP_ERROR_MSG_SEND            5// Indicates sending of the command to BTE Explorer failed. #define BTP_ERROR_MSG_RECEIVE         6// Indicates BTE Explorer failed to respond to the command. #define BTP_ERROR_NO_COM_PORT         7// Indicates there are no available COM ports with which to connect. #define BTP_ERROR_BTEXP_NOT_RUNNING   8// Indicates BTE Explorer could not be started. #define BTP_ERROR_NO_KEY_AVAILABLE    9// Returned by authentication callbacks, indicating the callback cannot provide the pass key for the specified device.  // Device States #define BTP_DEVICE_STATE_OFF          0 #define BTP_DEVICE_STATE_ON           1  // SCO Connection States #define BTP_SCO_STATE_DISCONNECTED    0 #define BTP_SCO_STATE_CONNECTED       1  //BTP_Perform_Action API #define BTP_ACTION_SHOW_SETTINGS      0x00000001 #define BTP_ACTION_DELETE_ALL_DEVICES 0x00000002  // Searching for remote devices #define BTP_DEVICE_NONE                0x0001 #define BTP_DEVICE_AUTHENTICATED       0x0002 #define BTP_DEVICE_REMEMBERED          0x0004 #define BTP_DEVICE_CONNECTED           0x0008 #define BTP_DEVICE_ALL                 0x8000  // Flags for the BTP_Connection_Query_t structure. #define BTP_CONNECTION_NONE            0 #define BTP_CONNECTION_REMEMBERED      1 #define BTP_CONNECTION_ACTIVE          2 #define BTP_CONNECTION_ALL             3 #define MESSAGE_TO_BTE_HEADER_SIZE    12 #define MESSAGE_FROM_BTE_HEADER_SIZE   4  //ListType #define DeviceFind                     1 #define ServiceFind                    2 #define ConnectionFind                 3 </pre>	
Enum	
<pre> typedef enum {     BTP_PROFILE_SPP,     BTP_PROFILE_DUN,     BTP_PROFILE_FAX,     BTP_PROFILE_LAN,     BTP_PROFILE_FILE_TRANSFER,     BTP_PROFILE_HEADSET,     BTP_PROFILE_HEADSET_AUDIO_GATEWAY,     BTP_PROFILE_HANDS_FREE, </pre>	

```

    BTP_PROFILE_HANDS_FREE_AUDIO_GATEWAY,
    BTP_PROFILE_HID_HOST,
    BTP_PROFILE_HID_DEVICE,
    BTP_PROFILE_UNKNOWN = -1
} BTP_Profile_Type;

typedef enum
{
    bdAll, //returns all devices
    bdHID //returns only HID devices
} BTP_DeviceType_t;

typedef enum
{
    deNIL,
    deNULL,
    deUnsignedInteger1Byte,
    deUnsignedInteger2Bytes,
    deUnsignedInteger4Bytes,
    deUnsignedInteger8Bytes,
    deUnsignedInteger16Bytes,
    deSignedInteger1Byte,
    deSignedInteger2Bytes,
    deSignedInteger4Bytes,
    deSignedInteger8Bytes,
    deSignedInteger16Bytes,
    deTextString,
    deBoolean,
    deURL,
    deUUID_16,
    deUUID_32,
    deUUID_128,
    deSequence,
    deAlternative
} SDP_Data_Element_Type_t;

typedef enum
{
    BTP_AUTHENTICATION_CALLBACK,
    BTP_CONNECTION_CALLBACK
} BTP_Callback_Type;

typedef enum
{
    BTP_SECURITYMODE_NONE,
    BTP_SECURITYMODE_AUTHENTICATE,
    BTP_SECURITYMODE_AUTHENTICATE_AND_ENCRYPT
} BTP_Security_Mode_Type;

```

## Structure

```

typedef struct
{
    Byte_t BD_ADDR0;
    Byte_t BD_ADDR1;
    Byte_t BD_ADDR2;
    Byte_t BD_ADDR3;
    Byte_t BD_ADDR4;
    Byte_t BD_ADDR5;
} BD_ADDR_t;

typedef struct
{
    SDP_Data_Element_Type_t Data_Element_Type;
    union
    {
        {
            UUID_16_t    UUID_16;
            UUID_32_t    UUID_32;
            UUID_128_t   UUID_128;
        } UUID_Value;
    }
} SDP_UUID_Entry_t;

typedef struct
{
    Byte_t UUID_Byte0;
    Byte_t UUID_Byte1;
} UUID_16_t;

typedef struct
{
    Byte_t UUID_Byte0;
    Byte_t UUID_Byte1;
    Byte_t UUID_Byte2;
    Byte_t UUID_Byte3;
} UUID_32_t;

typedef struct
{
    Byte_t UUID_Byte0;
    Byte_t UUID_Byte1;
    Byte_t UUID_Byte2;
    Byte_t UUID_Byte3;
    Byte_t UUID_Byte4;
    Byte_t UUID_Byte5;
    Byte_t UUID_Byte6;
    Byte_t UUID_Byte7;
    Byte_t UUID_Byte8;
    Byte_t UUID_Byte9;
    Byte_t UUID_Byte10;
    Byte_t UUID_Byte11;
}

```



```

    Byte_t UUID_Byte12;
    Byte_t UUID_Byte13;
    Byte_t UUID_Byte14;
    Byte_t UUID_Byte15;
} UUID_128_t;

typedef struct
{
    Word_t DeviceAttributes;
    Byte_t InquiryTimeout;
} BTP_Device_Query_t;

typedef struct _tagBTP_Device_Query_Ex_t
{
    unsigned int      Size;
    Byte_t            Version;
    Word_t            DeviceAttributes;
    Byte_t            InquiryTimeout;
    BTP_DeviceType_t  DeviceType;
} BTP_Device_Query_Ex_t;

typedef struct
{
    BD_ADDR_t          BD_ADDR;
    Class_of_Device_t  ClassOfDevice;
    Word_t             DeviceAttributes;
    char               Name[BLUETOOTH_MAX_NAME_SIZE];
} BTP_Device_Info_t;

typedef struct
{
    BD_ADDR_t          BD_ADDR;
    Word_t             NumberServiceUUID;
    SDP_UUID_Entry_t*  Service;
} BTP_Service_Query_t;

typedef struct
{
    BTP_Profile_Type   ProfileType;
    unsigned char      MajorVersion;
    unsigned char      MinorVersion;
    char               ServiceName[BLUETOOTH_MAX_NAME_SIZE];
    unsigned int        RFCOMMPort;
} BTP_Service_Info_t;

typedef struct _tagBTP_Service_Info_Ex_t
{
    unsigned int      Size;
    unsigned int      Version;
    BTP_Profile_Type   ProfileType;
    unsigned char      MajorVersion;
    unsigned char      MinorVersion;

```

```

char                ServiceName[BLUETOOTH_MAX_NAME_SIZE];
unsigned int        ListIndex;
union
{
    BTPSerialPortProfileInfo_t    RemoteSerialPortProfileInfo;
    BTPHIDProfileInfo_t          RemoteHIDProfileInfo;
}ProfileInformation;
} BTP_Service_Info_Ex_t;

```

```
typedef struct _tagBTPSerialPortProfileInfo_t
```

```

{
    unsigned int    RFCOMMServerPort;
    bool            UseActiveSync;
} BTPSerialPortProfileInfo_t;

```

```
typedef struct _tagBTPHIDProfileInfo_t
```

```

{
    unsigned int    L2CAPControlChannel;
    unsigned int    L2CAPInterruptChannel;
    Byte_t          DeviceSubclass;
    bool            VirtualCableSupported;
    bool            DeviceAutomaticReconnect;
    bool            DeviceNormallyConnectable;
} BTPHIDProfileInfo_t;

```

```
typedef struct
```

```

{
    Word_t    ConnectionAttributes;
} BTP_Connection_Query_t;

```

```
typedef struct
```

```

{
    BTP_Connection_ID    ConnectionID;
    BD_ADDR_t            BD_ADDR;
    unsigned int          RFCOMMPort;
    int                   LocalCOMPort;
    unsigned char          MajorVersion;
    unsigned char          MinorVersion;
    Word_t                 ConnectionAttributes;
    BTP_Profile_Type       ProfileType;
} BTP_Connection_Info_t;

```

```
typedef struct _tagBTP_Connection_Info_Ex_t
```

```

{
    unsigned int        Size;
    unsigned int        Version;
    BTP_Connection_ID    ConnectionID;
    BD_ADDR_t            BD_ADDR;
    int                  LocalCOMPort;
    unsigned char        MajorVersion;
    unsigned char        MinorVersion;
}

```

```

Word_t          ConnectionAttributes;
BTP_Profile_Type ProfileType;
HLOCAL          ListHandle;
unsigned int     ListIndex;
union
{
    BTPSerialPortProfileInfo_t RemoteSerialPortProfileInfo;
    BTPHIDProfileInfo_t        RemoteHIDProfileInfo;
}ProfileInformation;
} BTP_Connection_Info_Ex_t;
typedef struct _tagBTPSerialPortProfileInfo_t
{
    unsigned int  RFCOMMServerPort;
    bool          UseActiveSync;
} BTPSerialPortProfileInfo_t;
typedef struct _tagBTPHIDProfileInfo_t
{
    unsigned int  L2CAPControlChannel;
    unsigned int  L2CAPInterruptChannel;
    Byte_t        DeviceSubclass;
    bool          VirtualCableSupported;
    bool          DeviceAutomaticReconnect;
    bool          DeviceNormallyConnectable;
} BTPHIDProfileInfo_t;
typedef struct _tagBTP_PIN_t
{
    unsigned int  PINLength;
    BTP_PIN_Code_t PINCode;
} BTP_PIN_t;
typedef struct _tagBTP_PIN_Code_t
{
    Byte_t PIN_Code0;
    Byte_t PIN_Code1;
    Byte_t PIN_Code2;
    Byte_t PIN_Code3;
    Byte_t PIN_Code4;
    Byte_t PIN_Code5;
    Byte_t PIN_Code6;
    Byte_t PIN_Code7;
    Byte_t PIN_Code8;
    Byte_t PIN_Code9;
    Byte_t PIN_Code10;
    Byte_t PIN_Code11;
    Byte_t PIN_Code12;
    Byte_t PIN_Code13;
    Byte_t PIN_Code14;

```

```
    Byte_t PIN_Code15;  
} BTP_PIN_Code_t;
```

## Functions for Bluetooth

Name	Description
<a href="#">BLUETOOTH_Close</a>	Performs cleanup after the API is no longer needed.
<a href="#">BLUETOOTH_Connect</a>	Activates the specified connection in the connection list.
<a href="#">BLUETOOTH_CreateConnection</a>	Defines a new connection, adding it to a list of connections.
<a href="#">BLUETOOTH_CreateConnectionEx</a>	Defines a new connection, adding it to a list of connections.
<a href="#">BLUETOOTH_DeleteConnection</a>	Deletes a connection from the list of connections.
<a href="#">BLUETOOTH_Disconnect</a>	Disconnects from the specified connection.
<a href="#">BLUETOOTH_FindConnectionClose</a>	Frees remote resources.
<a href="#">BLUETOOTH_FindDeviceClose</a>	Frees remote resources.
<a href="#">BLUETOOTH_FindFirstConnection</a>	Finds the first connection meeting the specified criteria.
<a href="#">BLUETOOTH_FindFirstConnectionEx</a>	Finds the first connection meeting the specified criteria.
<a href="#">BLUETOOTH_FindFirstDevice</a>	Performs a GAP inquiry to discover devices, returning the first device meeting the specified criteria.
<a href="#">BLUETOOTH_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="#">BLUETOOTH_FindFirstService</a>	Performs an SDP query to discover remote services for a specified device, returning the first service in the list.
<a href="#">BLUETOOTH_FindFirstServiceEx</a>	Performs an SDP query to discover remote services for a specified device, returning the first service in the list.
<a href="#">BLUETOOTH_FindLocalDevice</a>	Returns device information for the local device.
<a href="#">BLUETOOTH_FindNextConnection</a>	Returns the next connection meeting the criteria specified in the call to <a href="#">BLUETOOTH_FindFirstConnection</a> .
<a href="#">BLUETOOTH_FindNextConnectionEx</a>	Returns the next connection meeting the criteria specified in the call to <a href="#">BLUETOOTH_FindFirstConnection</a> .
<a href="#">BLUETOOTH_FindNextDevice</a>	Returns the next device in the list meeting the initial criteria, specified in the call to <a href="#">BLUETOOTH_FindFirstDevice</a> .
<a href="#">BLUETOOTH_FindNextService</a>	Returns the next service in the list.
<a href="#">BLUETOOTH_FindNextServiceEx</a>	Returns the next service in the list.
<a href="#">BLUETOOTH_FindServiceClose</a>	Frees remote resources.
<a href="#">BLUETOOTH_GetBluetoothState</a>	Gets the Bluetooth device state.
<a href="#">BLUETOOTH_GetSCOConnectionState</a>	Gets the SCO connection state.
<a href="#">BLUETOOTH_GetSecurityMode</a>	Gets the current authentication and encryption settings.
<a href="#">BLUETOOTH_Open</a>	Initializes the BTE Explorer API module.
<a href="#">BLUETOOTH_PerformAction</a>	Triggers BTE Explorer to take the requested action.
<a href="#">BLUETOOTH_SetAuthenticationCallback</a>	Sets the authentication callback for the specified device.

<a href="#">BLUETOOTH_SetBluetoothState</a>	Sets the Bluetooth device state to either on or off.
<a href="#">BLUETOOTH_SetConnectionCallback</a>	Sets the connection callback for the specified connection.
<a href="#">BLUETOOTH_SetIncomingPIN</a>	Sets or clears a static PIN to be used for any incoming Connections.
<a href="#">BLUETOOTH_SetOutgoingPIN</a>	Sets or clears a static PIN to be used for any outgoing Connections.
<a href="#">BLUETOOTH_SetSecurityMode</a>	Sets the authentication and encryption settings for future connections.

## 4.8.1 BLUETOOTH\_Close

### Description

This function is responsible for cleaning up the module after it is no longer needed by the application.

### Syntax

```
BLUETOOTH_API void BLUETOOTH_Close();
```

### Parameters

None

### Return Value

None

### Remarks

None

### See Also

BLUETOOTH\_Open

### For .Net

Namespace : BlueToothNet Bluetooth

Function : void Close()

### Example

```
BLUETOOTH_Close();
```

## 4.8.2 BLUETOOTH\_Connect

### Description

This function connects to a connection previously defined by a call to BLUETOOTH\_CreateConnection.

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_Connect(BTP_Connection_ID ConnectionID);
```

### Parameters

*ConnectionID*

Unique identifier for a connection which was previously defined by a call to BLUETOOTH\_CreateConnection and BLUETOOTH\_CreateConnectionEx.

### Return Value

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR , BTP\_ERROR\_INVALID\_PARAMETER

### Remarks

None

## See Also

BLUETOOTH\_Disconnect

## For .Net

Namespace : BluetoothNet Bluetooth

Function : Int32 Connect(BT\_Connection\_ID ConnectionID)

## Example

```
hResult = BLUETOOTH_Connect(m_connectionInfo.ConnectionID);
```

## 4.8.3 BLUETOOTH\_CreateConnection

### Description

This function defines a temporary or persistent (Favorite) connection, which may later be connected by a call to BLUETOOTH\_Connect (Supports SPP only).

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_CreateConnection(BTP_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

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_PARAMETER

### Remarks

None

## See Also

BLUETOOTH\_DeleteConnection

## For .Net

Namespace : BluetoothNet Bluetooth

Function : Int32 CreateConnection(ref BT\_Connection\_Info\_t ConnectionInfo)

## Example

None

## 4.8.4 BLUETOOTH\_CreateConnectionEx

### Description

This function defines a temporary or persistent (Favorite) connection, which may later be connected by a call to BLUETOOTH\_Connect (Supports SPP and HID).



## Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_CreateConnectionEx(BTP_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

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR , BTP\_ERROR\_INVALID\_PARAMETER

## Remarks

None

## See Also

BLUETOOTH\_DeleteConnection

## For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 CreateConnectionEx(ref BT\_Connection\_Info\_Ex\_t ConnectionInfo)

## Example

```
hResult = Bluetooth.CreateConnectionEx(ref m_ConnectionInfo);
```

## 4.8.5 BLUETOOTH\_DeleteConnection

### Description

This function discards a previously defined connection. After deleting a connection, its connection ID is no longer valid.

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_DeleteConnection(BTP_Connection_ID ConnectionID);
```

### Parameters

*ConnectionID*

Unique identifier for a connection which was previously defined by a call to BLUETOOTH\_CreateConnection and BLUETOOTH\_CreateConnectionEx.

### Return Value

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR , BTP\_ERROR\_INVALID\_PARAMETER

### Remarks

None

### See Also

BLUETOOTH\_CreateConnection, BLUETOOTH\_CreateConnectionEx

#### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 DeleteConnection(BT\_Connection\_ID ConnectionID)

#### Example

```
hResult = BLUETOOTH_DeleteConnection(m_connectionInfo.ConnectionID);
```

### 4.8.6 BLUETOOTH\_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

```
BLUETOOTH_API HRESULT BLUETOOTH_Disconnect(BTP_Connection_ID ConnectionID);
```

#### Parameters

*ConnectionID*

Unique identifier for a connection which was previously defined by a call to BLUETOOTH\_CreateConnection and BLUETOOTH\_CreateConnectionEx.

#### Return Value

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR , BTP\_ERROR\_INVALID\_ PARAMETER

#### Remarks

None

#### See Also

BLUETOOTH\_Connect

#### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 Disconnect(BT\_Connection\_ID ConnectionID)

#### Example

```
hResult = BLUETOOTH_Connect(m_connectionInfo.ConnectionID);
```

### 4.8.7 BLUETOOTH\_FindConnectionClose

#### Description

This function deletes the remote list of connections and performs any additional cleanup.

#### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_FindConnectionClose(BTP_Connection_Find ConnectionFind);
```

## Parameters

*ConnectionFind*

Handle to the list connections, originally returned by a call to `BLUETOOTH_FindFirstConnection` or `BLUETOOTH_FindFirstConnectionEx`.

## Return Value

`BTP_ERROR_SUCCESS` if successful.

Error codes include the following values: `BTP_ERROR` , `BTP_ERROR_INVALID_HANDLE`

## Remarks

None

## See Also

`BLUETOOTH_FindFirstConnection`, `BLUETOOTH_FindFirstConnectionEx`

## For .Net

Namespace : `BluetoothNet Bluetooth`

Function : `Int32 FindConnectionClose(BT_Connection_Find ConnectionFind)`

## Example

```
BLUETOOTH_FindConnectionClose(connectFind);
```

## 4.8.8 BLUETOOTH\_FindDeviceClose

### Description

This function deletes the remote list of devices and performs any additional cleanup.

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_FindDeviceClose(BTP_Device_Find DeviceFind);
```

## Parameters

*DeviceFind*

Handle to the list of discovered devices, originally returned by a call to `BLUETOOTH_FindFirstDevice`.

## Return Value

`BTP_ERROR_SUCCESS` if successful.

Error codes include the following values: `BTP_ERROR` , `BTP_ERROR_INVALID_HANDLE`

## Remarks

None

## See Also

`BLUETOOTH_FindFirstDevice`

## For .Net

Namespace : `BluetoothNet Bluetooth`

Function : Int32 FindDeviceClose(BT\_Device\_Find DeviceFind)

### Example

```
BLUETOOTH_FindDeviceClose(deviceFind);
```

## 4.8.9 BLUETOOTH\_FindFirstConnection

### Description

This function creates a list of active connections and Favorites to traverse, returning the first connection from the list.

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_FindFirstConnection( BTP_Connection_Find *ConnectionFind,  
Connection_Info_t *ConnectionInfo, const BTP_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

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR , BTP\_ERROR\_NO\_MORE, BTP\_ERROR\_INVALID\_PARAMETER

### Remarks

None

### See Also

BLUETOOTH\_FindConnectionClose

### For .Net

Namespace : BlueToothNet Bluetooth

```
Function : Int32 FindFirstConnection(ref BT_Connection_Find ConnectionFind,ref  
BT_Connection_Info_t ConnectionInfo, ref BT_Connection_Query_t ConnectionQuery)
```

### Example

```
hResult = BLUETOOTH_FindFirstConnection(&connectFind, &connectionInfo, &connectQuery);
```

## 4.8.10 BLUETOOTH\_FindFirstConnectionEx

### Description

This function creates a list of active connections and Favorites to traverse, returning the first connection from the list.

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_FindFirstConnectionEx(BTP_Connection_Find *ConnectionFind, BTP_Connection_Info_Ex_t *ConnectionInfoEx, const BTP_Connection_Query_t *ConnectionQuery);
```

### Parameters

*ConnectionFind*

Handle to the list of connections, needed for subsequent calls to BLUETOOTH\_FindNextConnectionEx and BLUETOOTH\_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

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR , BTP\_ERROR\_NO\_MORE, BTP\_ERROR\_INVALID\_PARAMETER

### Remarks

None

### See Also

BLUETOOTH\_FindConnectionClose

### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 FindFirstConnectionEx(ref BT\_Connection\_Find ConnectionFind,ref BT\_Connection\_Info\_Ex\_t ConnectionInfoEx, ref BT\_Connection\_Query\_t ConnectionQuery)

### Example

None

## 4.8.11 BLUETOOTH\_FindFirstDevice

### Description

This function initializes a GAP inquiry, creating a list of discovered Bluetooth devices. The first device is returned by this function.

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_FindFirstDevice(BTP_Device_Find *DeviceFind, BTP_Device_Info_t *DeviceInfo, const BTP_Device_Query_t *DeviceQuery);
```

### Parameters

*DeviceFind*

Handle to the list of discovered devices, needed for subsequent calls to `BLUETOOTH_FindNextDevice` and `BLUETOOTH_FindDeviceClose`.

*DeviceInfo*

Information defining the first device.

*DeviceQuery*

Provides parameters for the GAP Inquiry and filtering for the devices to retrieve.

### **Return Value**

`BTP_ERROR_SUCCESS` if successful.

Error codes include the following values: `BTP_ERROR` , `BTP_ERROR_NO_MORE`, `BTP_ERROR_INVALID_PARAMETER`

### **Remarks**

None

### **See Also**

`BLUETOOTH_FindDeviceClose`

### **For .Net**

Namespace : `BlueToothNet Bluetooth`

Function : `Int32 FindFirstDevice(ref BT_Device_Find DeviceFind, ref BT_Device_Info_t DeviceInfo, ref BT_Device_Query_t DeviceQuery)`

### **Example**

None

## **4.8.12 BLUETOOTH\_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**

```
BLUETOOTH_API HRESULT BLUETOOTH_FindFirstDeviceEx(BTP_Device_Find *DeviceFind,  
BTP_Device_Info_t *DeviceInfo, const BTP_Device_Query_Ex_t *DeviceQuery);
```

### **Parameters**

*DeviceFind*

Handle to the list of discovered devices, needed for subsequent calls to `BLUETOOTH_FindNextDevice` and `BLUETOOTH_FindDeviceClose`.

*DeviceInfo*

Information defining the first device.

*DeviceQueryEx*

Provides parameters for the GAP Inquiry and filtering for the devices to retrieve.

## Return Value

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR , BTP\_ERROR\_NO\_MORE, BTP\_ERROR\_INVALID\_PARAMETER

## Remarks

None

## See Also

BLUETOOTH\_FindDeviceClose

## For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 FindFirstDeviceEx(ref BT\_Device\_Find DeviceFind, ref BT\_Device\_Info\_t DeviceInfo, ref BT\_Device\_Query\_Ex\_t DeviceQuery)

## Example

```
hResult = BLUETOOTH_FindFirstDeviceEx(&deviceFind, &deviceInfo, &deviceQuery);
```

## 4.8.13 BLUETOOTH\_FindFirstService

### Description

This function performs an SDP service discovery on the specified device, return the first service from the resulting list.

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_FindFirstService(BTP_Service_Find *ServiceFind,  
BTP_Service_Info_t *ServiceInfo, const BTP_Service_Query_t *ServiceQuery);
```

### Parameters

*ServiceFind*

Handle to the list of remote services, needed for subsequent calls to BLUETOOTH\_FindNextService and BLUETOOTH\_FindServiceClose.

*ServiceInfo*

Information defining the first remote service in the list.

*ServiceQuery*

Provides parameters for the SDP query.

### Return Value

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_NO\_MORE, BTP\_ERROR\_INVALID\_HANDLE

### Remarks

None

## See Also

BLUETOOTH\_FindServiceClose

## For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 FindFirstService(ref BT\_Service\_Find ServiceFind, ref BT\_Service\_Info\_t ServiceInfo, ref BT\_Service\_Query\_t ServiceQuery)

## Example

None

### 4.8.14 BLUETOOTH\_FindFirstServiceEx

## Description

This function performs an SDP service discovery on the specified device, return the first service from the resulting list.

## Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_FindFirstServiceEx(BTP_Service_Find *ServiceFind,  
BTP_Service_Info_Ex_t *ServiceInfo, const BTP_Service_Query_t *ServiceQuery);
```

## Parameters

*ServiceFind*

Handle to the list of remote services, needed for subsequent calls to BLUETOOTH\_FindNextServiceEx and BLUETOOTH\_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

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_NO\_MORE, BTP\_ERROR\_INVALID\_HANDLE

## Remarks

None

## See Also

BLUETOOTH\_FindServiceClose

## For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 FindFirstServiceEx(ref BT\_Service\_Find ServiceFind, ref BT\_Service\_Info\_Ex\_t ServiceInfo, ref BT\_Service\_Query\_t ServiceQuery)



## Example

```
hResult = BLUETOOTH_FindFirstServiceEx(&serviceFind, &serviceInfo, &serviceQuery);
```

### 4.8.15 BLUETOOTH\_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

```
BLUETOOTH_API HRESULT BLUETOOTH_FindLocalDevice(BTP_Find_Local_Device_From_BTE_t *DeviceInfo);
```

#### Parameters

*DeviceInfo*

Will receive information defining the local device.

#### Return Value

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR\_INVALID\_PARAMETER, BTP\_ERROR\_MSG\_SEND

#### Remarks

None

#### See Also

None

#### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 FindLocalDevice(ref BT\_Find\_Local\_Device\_From\_BTE\_t DeviceInfo)

#### Example

```
hResult = BLUETOOTH_FindLocalDevice(&device);
```

### 4.8.16 BLUETOOTH\_FindNextConnection

#### Description

This function retrieves the next connection from the list originally created by a call to BLUETOOTH\_FindFirstConnection.

#### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_FindNextConnection(BTP_Connection_Find ConnectionFind, BTP_Connection_Info_t *ConnectionInfo);
```

#### Parameters

*ConnectionFind*

Handle to the list connections, originally returned by a call to BLUETOOTH\_FindFirstConnection.

## *ConnectionInfo*

Information defining a connection from the list.

### **Return Value**

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_NO\_MORE, BTP\_ERROR\_INVALID\_HANDLE

### **Remarks**

None

### **See Also**

BLUETOOTH\_FindConnectionClose

### **For .Net**

Namespace : BlueToothNet Bluetooth

Function : Int32 FindNextConnection(BT\_Connection\_Find ConnectionFind, ref BT\_Connection\_Info\_t ConnectionInfo)

### **Example**

```
hResult = BLUETOOTH_FindNextConnection(connectFind, &connectionInfo);
```

## **4.8.17 BLUETOOTH\_FindNextConnectionEx**

### **Description**

This function retrieves the next connection from the list originally created by a call to BLUETOOTH\_FindFirstConnection.

### **Syntax**

```
BLUETOOTH_API HRESULT BLUETOOTH_FindNextConnectionEx(BT_Connection_Find ConnectionFind, BTP_Connection_Info_Ex_t *ConnectionInfo);
```

### **Parameters**

#### *ConnectionFind*

Handle to the list connections, originally returned by a call to BLUETOOTH\_FindFirstConnection.

#### *ConnectionInfoEx*

Information defining a connection from the list. Supports SPP and HID.

### **Return Value**

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_NO\_MORE, BTP\_ERROR\_INVALID\_HANDLE

### **Remarks**

None

### **See Also**

BLUETOOTH\_FindConnectionClose

#### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 FindNextConnectionEx( ref BT\_Connection\_Find ConnectionFind, ref BT\_Connection\_Info\_Ex\_t ConnectionInfo)

#### Example

None

### 4.8.18 BLUETOOTH\_FindNextDevice

#### Description

This function returns the next device in the list of discovered devices created by a previous call to BLUETOOTH\_FindFirstDevice.

#### Syntax

BLUETOOTH\_API HRESULT BLUETOOTH\_FindNextDevice(BTP\_Device\_Find DeviceFind, BTP\_Device\_Info\_t \*DeviceInfo);

#### Parameters

*DeviceFind*

Handle to the list of discovered devices, originally returned by a call to BLUETOOTH\_FindFirstDevice.

*DeviceInfo*

Information defining a remote device.

#### Return Value

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_NO\_MORE, BTP\_ERROR\_INVALID\_PARAMETER, BTP\_ERROR\_INVALID\_HANDLE

#### Remarks

None

#### See Also

BLUETOOTH\_FindDeviceClose

#### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 FindNextDevice(BT\_Device\_Find DeviceFind, ref BT\_Device\_Info\_t DeviceInfo)

#### Example

hResult = BLUETOOTH\_FindNextDevice(deviceFind, &deviceInfo);

## 4.8.19 BLUETOOTH\_FindNextService

### Description

This function returns the next service in the list of services created by a previous call to BLUETOOTH\_FindFirstService.

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_FindNextService(BTP_Service_Find ServiceFind,  
BTP_Service_Info_t *ServiceInfo);
```

### Parameters

*ServiceFind*

Handle to the list of remote services, originally returned by a call to BLUETOOTH\_FindFirstService.

*ServiceInfo*

Information defining a remote service from the list.

### Return Value

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_NO\_MORE, BTP\_ERROR\_INVALID\_HANDLE

### Remarks

None

### See Also

BLUETOOTH\_FindServiceClose

### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 FindNextService(BT\_Service\_Find ServiceFind, ref BT\_Service\_Info\_t ServiceInfo)

### Example

None

## 4.8.20 BLUETOOTH\_FindNextServiceEx

### Description

This function returns the next service in the list of services created by a previous call to BLUETOOTH\_FindFirstService.

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_FindNextService(BTP_Service_Find ServiceFind,  
BTP_Service_Info_t *ServiceInfo);
```

### Parameters

*ServiceFind*

Handle to the list of remote services, originally returned by a call to BLUETOOTH\_FindFirstService.

## *ServiceInfoEx*

Information defining the first remote service in the list. This currently supports SPP and HID services.

### **Return Value**

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_NO\_MORE, BTP\_ERROR\_INVALID\_HANDLE

### **Remarks**

None

### **See Also**

BLUETOOTH\_FindServiceClose

### **For .Net**

Namespace : BlueToothNet Bluetooth

Function : Int32 FindNextServiceEx(BT\_Service\_Find ServiceFind, ref BT\_Service\_Info\_Ex\_t ServiceInfo)

### **Example**

```
hResult = BLUETOOTH_FindNextServiceEx(serviceFind, &serviceInfo);
```

## **4.8.21 BLUETOOTH\_FindServiceClose**

### **Description**

This function deletes the remote list of services and performs any additional cleanup.

### **Syntax**

```
BLUETOOTH_API HRESULT BLUETOOTH_FindServiceClose(BTP_Service_Find ServiceFind);
```

### **Parameters**

*ServiceFind*

Handle to the list of remote services, originally returned by a call to BLUETOOTH\_FindFirstService or BLUETOOTH\_FindFirstServiceEx.

### **Return Value**

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_HANDLE

### **Remarks**

None

### **See Also**

BLUETOOTH\_FindFirstService, BLUETOOTH\_FindFirstServiceEx, BLUETOOTH\_FindNextService, BLUETOOTH\_FindNextServiceEx

### **For .Net**

Namespace : BlueToothNet Bluetooth

Function : Int32 FindServiceClose(BT\_Service\_Find ServiceFind)

#### Example

```
BLUETOOTH_FindServiceClose(serviceFind);
```

### 4.8.22 BLUETOOTH\_GetBluetoothState

#### Description

This function gets the current Bluetooth device state (either BTP\_DEVICE\_STATE\_ON or BTP\_DEVICE\_STATE\_OFF).

#### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_GetBluetoothState(DWord_t *BluetoothState);
```

#### Parameters

*BluetoothState*

Specifies the current Bluetooth state if the function returns BTP\_ERROR\_SUCCESS

#### Return Value

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_MSG\_SEND

#### Remarks

None

#### See Also

BLUETOOTH\_SetBluetoothState

#### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 GetBluetoothState(ref UInt32 BluetoothState)

#### Example

None

### 4.8.23 BLUETOOTH\_GetSCOConnectionState

#### Description

This function is used to get the SCO connection state for any open Hands-Free connection (either BTP\_SCO\_STATE\_CONNECTED or BTP\_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

```
BLUETOOTH_API HRESULT BLUETOOTH_GetSCOConnectionState(BD_ADDR_t *BD_ADDR, DWord_t *ConnectionState);
```

#### Parameters

*BD\_ADDR*

The Bluetooth address that BTE Explorer has a Hands-Free connection with.

*ConnectionState*

Specifies the current SCO connection state if the function returns BTP\_ERROR\_SUCCESS.

#### **Return Value**

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_PARAMETER, BTP\_ERROR\_MSG\_SEND

#### **Remarks**

None

#### **See Also**

BLUETOOTH\_SetSCOConnectionState

#### **For .Net**

Namespace : BlueToothNet Bluetooth

Function : Int32 GetSCOConnectionState(ref BD\_ADDR\_t BD\_ADDR, ref UInt32 ConnectionState)

#### **Example**

None

### **4.8.24 BLUETOOTH\_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: BTP\_SECURITYMODE\_NONE, BTP\_SECURITYMODE\_AUTHENTICATE, BTP\_SECURITYMODE\_AUTHENTICATE\_AND\_ENCRYPT

#### **Syntax**

BLUETOOTH\_API HRESULT BLUETOOTH\_GetSecurityMode(BTP\_Security\_Mode\_Type \*SecurityMode);

#### **Parameters**

*SecurityMode*

A pointer to a BTP\_Security\_Mode\_Type that will store the current security mode.

#### **Return Value**

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_PARAMETER, BTP\_ERROR\_MSG\_SEND

#### **Remarks**

None

## See Also

BLUETOOTH\_SetSecurityMode

## For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 GetSecurityMode(ref BT\_Security\_Mode\_Type SecurityMode)

## Example

None

## 4.8.25 BLUETOOTH\_Open

### Description

This function initializes the BTExplorer API module, readying it for use by the application.

### Syntax

BLUETOOTH\_API bool BLUETOOTH\_Open();

### Parameters

None

### Return Value

None

### Remarks

None

## See Also

BLUETOOTH\_Close

## For .Net

Namespace : BlueToothNet Bluetooth

Function : bool Open()

## Example

```
if(BLUETOOTH_Open())
{
    m_State.SetWindowText(L"Open Success");
}
else
    m_State.SetWindowText(L"Open Fail");
```

## 4.8.26 BLUETOOTH\_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

```
BLUETOOTH_API HRESULT BLUETOOTH_PerformAction(DWord_t Action, DWord_t Param1, DWord_t Param2, DWord_t *Return1, DWord_t *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

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_PARAMETER

### Remarks

None

### See Also

None

### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 PerformAction(UInt32 Action, UInt32 Param1, UInt32 Param2, UInt32 Return1, UInt32 Return2)

### Example

```
hResult = BLUETOOTH_PerformAction(BTP_ACTION_DELETE_ALL_DEVICES, 0, 0, NULL, NULL);
```

## 4.8.27 BLUETOOTH\_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

```
BLUETOOTH_API HRESULT BLUETOOTH_SetAuthenticationCallback(const BD_ADDR_t *BD_ADDR,  
BTP_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

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_PARAMETER

### Remarks

None

### See Also

None

### For .Net

Namespace : BluetoothNet Bluetooth

Function : Int32 SetAuthenticationCallback(ref BD\_ADDR\_t BD\_ADDR, ref  
BT\_Authentication\_Callback\_t AuthenticationCallback, ref LPVOID CallbackParameter)

### Example

None

## 4.8.28 BLUETOOTH\_SetBluetoothState

### Description

This function sets the Bluetooth device to either BTP\_DEVICE\_STATE\_ON or BTP\_DEVICE\_STATE\_OFF. Turning the device on and off will start and stop the BTE Explorer process.

### Syntax

```
BLUETOOTH_API HRESULT BLUETOOTH_SetBluetoothState(DWord_t BluetoothState);
```

### Parameters

### *BluetoothState*

Specifies the desired Bluetooth state.

#### **Return Value**

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_MSG\_SEND

#### **Remarks**

None

#### **See Also**

BLUETOOTH\_GetBluetoothState

#### **For .Net**

Namespace : BlueToothNet Bluetooth

Function : Int32 SetBLUETOOTHState(ref UInt32 BLUETOOTHState)

#### **Example**

None

## **4.8.29 BLUETOOTH\_SetConnectionCallback**

### **Description**

This function registers a connection callback with BTE Explorer. This function may be called multiple times to associate multiple connections with a callback, or to associate multiple callbacks with a connection.

### **Syntax**

```
BLUETOOTH_API HRESULT BLUETOOTH_SetConnectionCallback(BTP_Connection_ID ConnectionID,  
BTP_Connection_Callback_t ConnectionCallback, LPVOID CallbackParameter);
```

### **Parameters**

*ConnectionID*

Unique identifier for a connection which was previously defined by a call to BLUETOOTH\_CreateConnection.

*ConnectionCallback*

The callback being registered.

*CallbackParameter*

User-defined parameter associated with the callback.

### **Return Value**

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_PARAMETER

### **Remarks**

None

## See Also

None

## For .Net

Namespace : BluetoothNet Bluetooth

Function : Int32 SetConnectionCallback(ref BT\_Connection\_ID ConnectionID, ref BT\_Connection\_Callback\_t ConnectionCallback, ref LPVOID CallbackParameter)

## Example

None

## 4.8.30 BLUETOOTH\_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

```
BLUETOOTH_API HRESULT BLUETOOTH_SetIncomingPIN(BTP_PIN_t *NewPIN, BTP_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

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_PARAMETER

### Remarks

None

## See Also

BLUETOOTH\_SetOutgoingPIN

## For .Net

Namespace : BluetoothNet Bluetooth

Function : Int32 SetIncomingPIN(ref BT\_PIN\_t NewPIN, ref BT\_PIN\_t OldPIN)

## Example

None

### 4.8.31 BLUETOOTH\_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

```
BLUETOOTH_API HRESULT BLUETOOTH_SetOutgoingPIN(BTP_PIN_t *NewPIN, BTP_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

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_PARAMETER

#### Remarks

None

#### See Also

BLUETOOTH\_SetIncomingPIN

#### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 SetOutgoingPIN(ref BT\_PIN\_t NewPIN, ref BT\_PIN\_t OldPIN)

#### Example

None

4.8.32 BLUETOOTH\_SetSCOConnectionState

BLUETOOTH\_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 BTP\_SCO\_STATE\_CONNECTED and BTP\_SCO\_STATE\_DISCONNECTED.

#### Syntax

BLUETOOTH\_API HRESULT BLUETOOTH\_SetSCOConnectionState(BD\_ADDR\_t \*BD\_ADDR, DWord\_t 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

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_PARAMETER, BTP\_ERROR\_MSG\_SEND

#### Remarks

None

#### See Also

BLUETOOTH\_GetSCOConnectionState

#### For .Net

Namespace : BlueToothNet Bluetooth

Function : Int32 SetSCOConnectionState(ref BD\_ADDR\_t BD\_ADDR, ref UInt32 ConnectionState)

#### Example

None

### 4.8.32 BLUETOOTH\_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

BLUETOOTH\_API HRESULT BLUETOOTH\_SetSecurityMode(BTP\_Security\_Mode\_Type SecurityMode);

#### Parameters

*SecurityMode*

Specifies the desired security mode.

#### Return Value

BTP\_ERROR\_SUCCESS if successful.

Error codes include the following values: BTP\_ERROR, BTP\_ERROR\_INVALID\_PARAMETER, BTP\_ERROR\_MSG\_SEND

**Remarks**

None

**See Also**

BLUETOOTH\_GetSecurityMode

**For .Net**

Namespace : BlueToothNet Bluetooth

Function : Int32 SetSecurityMode(ref BT\_Security\_Mode\_Type SecurityMode)

**Example**

None

## 4.9 GPS

### Status Return value & API Error Codes

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

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

Define
<pre>#define SATELLITE_COUNT      12 #define WM_USER_RECVDATA    (WM_USER+10000)</pre>
Enum
<pre>typedef enum {     DEVICE_M3SKY = 0,     DEVICE_M3SKYSAM,     DEVICE_MM3,     DEVICE_M3ORANGE,     DEVICE_M3SMART_CE,     DEVICE_M3SMART_WM,     DEVICE_M3GREEN,     DEVICE_M3T,     DEVICE_M3POS } DEVICE_TYPE;  typedef enum {     MODULE_UNKNOWN = 0,     MODULE_UBLOX,     MODULE_SIRF } MODULE_TYPE;  typedef enum {     GPS_HOT_START = 0,     GPS_WARM_START,     GPS_COLD_START } GPS_START;  typedef enum {     AGPS_1DAY = 0,     AGPS_2DAY,     AGPS_3DAY,     AGPS_5DAY,     AGPS_7DAY,     AGPS_10DAY,     AGPS_14DAY</pre>



```
} AGPS_DAY;
```

## Structure

```
typedef struct GPSDop
```

```
{
```

```
    double dPDop;
```

```
    double dHDop;
```

```
    double dVDop;
```

```
};
```

```
typedef struct GPSSatellite
```

```
{
```

```
    int nID;
```

```
    int nElevation;
```

```
    int nAzimuth;
```

```
    int nSNR;
```

```
};
```

```
typedef struct {
```

```
    struct GPSSatellite mSat[SATELLITE_COUNT];
```

```
    struct GPSDop mDop;
```

```
    int nSatInUse;
```

```
    int nSatNum;
```

```
    BOOL bSatInfo;
```

```
    double dHeading;
```

```
    double dVelocity;
```

```
    BOOL bNorthLatitude;
```

```
    BOOL bEastLongitude;
```

```
    double dLatitude;
```

```
    double dLongitude;
```

```
    double dAltitude;
```

```
    double dUTCDate;
```

```
    double dUTCTime;
```

```
    int nPosFix;
```

```
    int nGPSStatus;
```

```
    TCHAR mNMEAmsg[256];
```

```
}GPSParseInfo;
```

## Void

```
typedef void (*GetGpsInfoProc)(GPSParseInfo info);
```

## Functions for GPS

Name	Description
<a href="#">GPS_AGPSDown</a>	Downloads Almanac Data from AGPS Server.
<a href="#">GPS_AGPSRun</a>	Applies Almanac Data to GPS Module.
<a href="#">GPS_Close</a>	GPS can be closed through simply closing the opened Serial Port.
<a href="#">GPS_EnableStaticMode</a>	Location determination when the receiver's antenna is presumed to be stationary on the earth.
<a href="#">GPS_ModuleRestart</a>	The process of powering up a new GPS receiver for the first time and having it search out and lock onto the satellite by itself, without the benefit of initialization data.
<a href="#">GPS_Open</a>	To open GPS, input the COM port number, Baud Rate for serial communication and Windows HWND for getting GPS data.

### 4.9.1 GPS\_AGPSDown

#### Description

Downloads Almanac Data from AGPS Server.

#### Syntax

```
GPS_API BOOL GPS_AGPSDown(AGPS_DAY day, const TCHAR* tzDestFilePath);
```

#### Parameters

*day*

Date of Almanac Data

*tzDestFilePath*

File path of Almanac Data

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

GPS\_AGPSRun

#### For .Net

Namespace : GPSNet Gps

Function : bool AGPSDown(AGPS\_DAY day, string tzDestFilePath)

#### Example

```
GPS_AGPSDown(day, tzInputFile);
```

### 4.9.2 GPS\_AGPSRun

#### Description

Applies Almanac Data received from GPS\_AGPSDown to GPS module.

#### Syntax

```
GPS_API BOOL GPS_AGPSRun(const TCHAR* tzInputFilePath);
```

#### Parameters

*tzInputFilePath*

Location that Almanac Data is stored.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

### See Also

GPS\_AGPSDown

### For .Net

Namespace : GPSNet Gps

Function : bool AGPSRun(string tzInputFilePath)

### Example

```
GPS_AGPSRun(tzInputFile);
```

## 4.9.3 GPS\_Close

### Description

This function is responsible for cleaning up the module after it is no longer needed by the application.

### Syntax

```
GPS_API BOOL GPS_Close();
```

### Parameters

None

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

GPS\_Open

### For .Net

Namespace : GPSNet Gps

Function : bool Close()

### Example

```
if(GPS_Close())
{
    AfxMessageBox(L"Close");
}
else
{
    AfxMessageBox(L"Close Fail");
}
```

#### 4.9.4 GPS\_EnableStaticMode

##### Description

This function is location determination when the receiver's antenna is presumed to be stationary on the earth. This allows the use of various averaging techniques that improve accuracy by factors of over 1000.

##### Syntax

```
GPS_API BOOL GPS_EnableStaticMode(BOOL bEnable);
```

##### Parameters

*bEnable*

TRUE : Static Mode Support

FALSE : Static Mode not supported

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

None

##### See Also

None

##### For .Net

Namespace : GPSNet Gps

Function : bool EnableStaticMode(bool bEnable)

##### Example

```
if(m_chkStatic.GetCheck() == BST_CHECKED)
{
    GPS_EnableStaticMode(TRUE);
}
else if(m_chkStatic.GetCheck() == BST_UNCHECKED)
{
    GPS_EnableStaticMode(FALSE);
}
```

#### 4.9.5 GPS\_ModuleRestart

##### Description

This function is the process of powering up a new GPS receiver for the first time and having it search out and lock onto the satellite by itself, without the benefit of initialization data.

##### Syntax

GPS\_API BOOL GPS\_ModuleRestart(GPS\_START StartType);

#### Parameters

*StartType*

GPS Restart Type.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

None

#### For .Net

Namespace : GPSNet Gps

Function : bool ModuleRestart(GPS\_START StartType)

#### Example

```
switch(m_cmbStartType.GetCurSel())
{
    case GPS_HOT_START:
        GPS_ModuleRestart(GPS_HOT_START);
        break;
    case GPS_WARM_START:
        GPS_ModuleRestart(GPS_WARM_START);
        break;
    case GPS_COLD_START:
        GPS_ModuleRestart(GPS_COLD_START);
        break;
    default:
        GPS_ModuleRestart(GPS_HOT_START);
        break;
}
```

### 4.9.6 GPS\_Open

#### Description

This function initializes the GPS module, readying it for use by the application.

#### Syntax

```
GPS_API BOOL GPS_Open(HWND hMainWnd, TCHAR* tzComPort, GetGpsInfoProc GetFunc = NULL);
```

### Parameters

*hMainWnd*

Windows Handle of GPS Application.

*tzComPort*

COM Port for GPS Open.

*GetFunc*

Receiving GPS Data.

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

GPS\_Close

### For .Net

Namespace : GPSNet Gps

Function : bool Open(HWND hMainWnd, string tzComPort, GetGpsInfoProc Func)

### Example

```
if(GPS_Open(m_hWnd, tzCom, fnParseGps))
{
    AfxMessageBox(L"Open Fail");
}
else
{
    AfxMessageBox(L"Open Fail");
}
```

# 4.10 SYSTEM

## Status Return value

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

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

Define
<pre>#define DEVICE_M3SKY 0 #define DEVICE_M3SKYSAM 1 #define DEVICE_M3ORANGE 2 #define DEVICE_M3SMARTCE 3 #define DEVICE_M3SMARTWM 4 #define DEVICE_M3T 5 #define DEVICE_M3ORANGEPLUS 6 #define DEVICE_POS 7 #define DEVICE_MM3 8 #define DEVICE_M3ORANGEPLUS_CE 9</pre>
Enum
<pre>typedef enum {     DEVICE_M3SKY = 0,     DEVICE_M3SKYSAM,     DEVICE_M3ORANGE,     DEVICE_M3SMARTCE,     DEVICE_M3SMARTWM,     DEVICE_M3T,     DEVICE_M3ORANGEPLUS,     DEVICE_POS,     DEVICE_MM3,     DEVICE_M3ORANGEPLUS_CE } DEVICE_INFO; typedef enum{     BATTERY_POWER,     EXTERNAL_POWER }POWERTYPE;</pre>
Structruem
<pre>typedef struct _GSensor_PARAMS {     int GsensorEnable;     bool DisplayRotate;</pre>



```
bool  MotionDisplayOff;  
bool  MotionDisplayWakeup;  
bool  MotionSleepOn;  
bool  MotionSleepWakeup;  
bool  MotionSleepOnPrevent;  
bool  MotionStateCheck;  
bool  DisplayMenuCheck;  
bool  SuspendMenuCheck;
```

```
}GSensor_PARAMS, *PGSenor_PARAMS;
```

## Functions for System

Name	Description
<a href="#">SYSTEM_BacklightOn</a>	Backlight On/Off
<a href="#">SYSTEM_Cleanboot</a>	Cleanboot
<a href="#">SYSTEM_GetBacklightlevel</a>	Gets the Backlight Level
<a href="#">SYSTEM_GetBacklightTimeOut</a>	Gets the BacklightTimeOut
<a href="#">SYSTEM_GetBatteryLifePercent</a>	Gets the Battery Life Percent
<a href="#">SYSTEM_GetBatteryState</a>	Gets the Battery State
<a href="#">SYSTEM_GetBluetooth</a>	Gets the Bluetooth
<a href="#">SYSTEM_GetCpuClock</a>	Gets the CPU Clock
<a href="#">SYSTEM_GetDeviceInfo</a>	Gets the Information of Device
<a href="#">SYSTEM_GetGUID</a>	Gets the GUID
<a href="#">SYSTEM_GetGSensorValue</a>	Gets the Gyro Sensor Value
<a href="#">SYSTEM_GetOSVersionInfo</a>	Gets the Information of OS Version
<a href="#">SYSTEM_GetPhoneVolumeLevel</a>	Gets the Phone Volume Level
<a href="#">SYSTEM_GetPowerTimeOut</a>	Gets the PowerTimeOut
<a href="#">SYSTEM_GetSerialNumber</a>	Gets the Serial Number
<a href="#">SYSTEM_GetVersionInfo</a>	Gets the Information of C++ DLL Version
<a href="#">SYSTEM_GetVolumeLevel</a>	Gets the Volume Level
<a href="#">SYSTEM_GetWlan</a>	Gets the Wlan
<a href="#">SYSTEM_KeypadLock</a>	Kaypad Lock/Unlock
<a href="#">SYSTEM_Reboot</a>	Reboot
<a href="#">SYSTEM_SetBacklightlevel</a>	Sets the Backlight Level
<a href="#">SYSTEM_SetBacklightTimeOut</a>	Sets the BacklightTimeOut
<a href="#">SYSTEM_SetBluetooth</a>	Sets the Bluetooth On/Off
<a href="#">SYSTEM_SetCpuClock</a>	Sets the CPU Clock
<a href="#">SYSTEM_SetGSensorValue</a>	Sets the Gyro Sensor Value
<a href="#">SYSTEM_SetPhoneVolumeLevel</a>	Sets the Phone Volume Level
<a href="#">SYSTEM_SetPowerTimeOut</a>	Sets the PowerTimeOut
<a href="#">SYSTEM_SetSleepMode</a>	Sleep On
<a href="#">SYSTEM_SetVolumeLevel</a>	Sets the Volume Level
<a href="#">SYSTEM_SetWlan</a>	Sets the Wlan On/Off

<a href="#">SYSTEM Vibrate</a>	Vibrate On/Off
<a href="#">SYSTEM VolumeMute</a>	Volume Mute

### 4.10.1 SYSTEM\_BacklightOn

#### Description

Backlight On/Off.

#### Syntax

```
SYSTEM_API BOOL    SYSTEM_BacklightOn(BOOL bOn);
```

#### Parameters

*bOn*

The state is either FALSE=Backlight Off, TRUE= Backlight On.

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

This function cannot be used with M3 SMART CE.

#### See Also

None

#### For .Net

Namespace : SystemNet.System

Function : bool BacklightOn(bool bOn)

#### Example

```
SYSTEM_BacklightOn(TRUE);
```

```
SYSTEM_BacklightOn(FALSE);
```

### 4.10.2 SYSTEM\_Cleanboot

#### Description

Cleanboot.

#### Syntax

```
SYSTEM_API BOOL    SYSTEM_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 .Net**

Namespace : SystemNet.System

Function : bool Cleanboot(bool bOn)

#### **Example**

```
BOOL m_bReboot = true;
SYSTEM_Cleanboot(m_bReboot);
SYSTEM_Reboot();
```

### **4.10.3 SYSTEM\_GetBacklightlevel**

#### **Description**

Gets the Backlight Level.

#### **Syntax**

```
SYSTEM_API DWORD SYSTEM_GetBacklightlevel();
```

#### **Parameters**

None

#### **Return Value**

Backlightlevel Value.

#### **Remarks**

None

#### **See Also**

SYSTEM\_SetBacklightlevel

#### **For .Net**

Namespace : SystemNet.System

Function : int GetBacklightlevel()

#### **Example**

```
DWORD dwBacklightlevel_Cur;
dwBacklightlevel_Cur = SYSTEM_GetBacklightlevel();
m_ctlBackLightLevel.SetPos(dwBacklightlevel_Cur);
switch(dwBacklightlevel_Cur)
{
    case 8:
        m_strBacklightLevel.Format(L"100%%");
        break;
```

```

case 7:
    m_strBacklightLevel.Format(L"90%%");
    break;
case 6:
    m_strBacklightLevel.Format(L"80%%");
    break;
case 5:
    m_strBacklightLevel.Format(L"70%%");
    break;
case 4:
    m_strBacklightLevel.Format(L"60%%");
    break;
case 3:
    m_strBacklightLevel.Format(L"40%%");
    break;
case 2:
    m_strBacklightLevel.Format(L"20%%");
    break;
case 1:
    m_strBacklightLevel.Format(L"10%%");
    break;
case 0:
    m_strBacklightLevel.Format(L"5%%");
    break;
default:
    m_strBacklightLevel.Format(L"60%%");
    break;
}

```

#### 4.10.4 SYSTEM\_GetBacklightTimeOut

##### Description

Gets the BacklightTimeOut.

##### Syntax

```
SYSTEM_API DWORD SYSTEM_GetBackLightTimeOut(POWERTYPE type);
```

**Parameters**

*type*

The state is either 0=Battery Status, 1= AC Status.

**Return Value**

BacklightTimeOut Value.

**Remarks**

None

**See Also**

SYSTEM\_SetBackLightTimeOut

**For .Net**

Namespace : SystemNet.System

Function : bool GetBacklightTimeOut(int PowerType)

**Example**

```
typedef enum{  
    BATTERY_POWER,  
    EXTERNAL_POWER  
}POWERTYPE;  
  
DWORD dwBacklightTimeout_Cur;  
  
dwBacklightTimeout_Cur = SYSTEM_GetBackLightTimeOut(BATTERY_POWER);  
  
m_ctlBackLightTimeout.SetCurSel(dwBacklightTimeout_Cur);
```

### 4.10.5 SYSTEM\_GetBatteryLifePercent

**Description**

Gets the Battery Life Percent.

**Syntax**

```
SYSTEM_API int SYSTEM_GetBatteryLifePercent();
```

**Parameters**

None

**Return Value**

Current BatteryLife Value.

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : SystemNet.System

Function : int GetBatteryLifePercent()

**Example**

```
int bBatteryLife;  
bBatteryLife = SYSTEM_GetBatteryLifePercent();
```

### 4.10.6 SYSTEM\_GetBatteryState

**Description**

Gets the Battery State.

**Syntax**

```
SYSTEM_API BOOL    SYSTEM_GetBatteryState();
```

**Parameters**

None

**Return Value**

TRUE indicates AC Power. FALSE indicates Battery Power.

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : SystemNet.System

Function : bool GetBatteryState()

**Example**

```
if(SYSTEM_GetBatteryState() == TRUE)  
    m_strBatteryStatus.Format(L"Power : AC Power");  
else  
    m_strBatteryStatus.Format(L"Power : Battery Power");
```

### 4.10.7 SYSTEM\_GetBluetooth

**Description**

Gets the Bluetooth.

**Syntax**

```
SYSTEM_API BOOL    SYSTEM_GetBluetooth();
```



### Parameters

None

### Return Value

TRUE indicates Bluetooth ON. FALSE indicates Bluetooth OFF.

### Remarks

This function can only be used in M3 Mobile Device installing Windows Mobile OS.

### See Also

SYSTEM\_SetBluetooth

### For .Net

Namespace : SystemNet.System

Function : bool GetCODE93(out CODE93\_PARAMS pCode93)

### Example

```
if(SYSTEM_GetBluetooth() == TRUE)
    m_strBluetooth.Format(L"Bluetooth : ON");
else
    m_strBluetooth.Format(L"Bluetooth : OFF");
```

## 4.10.8 SYSTEM\_GetCpuClock

### Description

Gets the CPU Clock.

### Syntax

SYSTEM\_API int SYSTEM\_GetCpuClock(DWORD\* getclock);

### Parameters

*getclock*

Gets the Current CPU Clock Value.

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

This function can't be used in M3 Smart, MM3, M3 OX10.

### See Also

SYSTEM\_SetCpuClock

### For .Net

Namespace : SystemNet.System

Function : int GetCpuClock(out IntPtr getclock);

### Example

```
nCurClock = SYSTEM_GetCpuClock(&dwCpuClock);  
    switch (nCurClock)  
    {  
        case 1: //208000  
            m_strCpuClock.Format(L"CPU Clock : 208MHz");  
            break;  
        case 2: //416000  
            m_strCpuClock.Format(L"CPU Clock : 416MHz");  
            break;  
        case 3: //624000  
            m_strCpuClock.Format(L"CPU Clock : 624MHz");  
            break;  
        case 4: //806000  
            m_strCpuClock.Format(L"CPU Clock : 806MHz");  
            break;  
        default:  
            m_strCpuClock.Format(L"CPU Clock : 208MHz");  
            break;  
    }
```

### 4.10.9 SYSTEM\_GetDeviceInfo

#### Description

Gets the Information of Device

#### Syntax

```
SYSTEM_API int SYSTEM_GetDeviceInfo();
```

#### Parameters

None

#### Return Value

Nonzero indicates success. Zero indicates failure.

#### Remarks

None

#### See Also

None

## For .Net

Namespace : System.Net.System

Function : int GetDeviceInfo();

## Example

```
nDeviceInfo = SYSTEM_GetDeviceInfo();
switch (nDeviceInfo)
{
    case 0: //DEVICE_M3SKY:
        M3Sky_GetCpu();
        break;
    case 2: //DEVICE_M3ORA NGE:
        M3Orange_GetCpu();
        break;
    case 4: //DEVICE_M3SMARTWM:
        M3SmartWM_GetCpu();
        break;
    case 5: //DEVICE_M3T:
        M3T_GetCpu();
        break;
    case 6: //DEVICE_ORAGEPLUS:

        break;
    case 7: //DEVICE_Pos:
        Pos_GetCpu();
        break;
    case 8: //DEVICE_MM3:
        MM3_GetCpu();
        break;
}
```

### 4.10.10 SYSTEM\_GetGUID

#### Description

Gets the GUID

#### Syntax

```
SYSTEM_API bool SYSTEM_GetGUID(TCHAR* szGUID);
```

**Parameters**

szGUID

Gets the GUID Value.

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

None

**See Also**

SYSTEM\_GetGUID

**For .Net**

Namespace : SystemNet.System

Function : String GetGuid();

**Example**

```
TCHAR strGUID[1024] = {0,};  
SYSTEM_GetGUID(strGUID);  
m_strGuid.Format(L"%s", strGUID);
```

### 4.10.11 SYSTEM\_GetGSensorValue

**Description**

Gets the Gyro Sensor Value

**Syntax**

```
SYSTEM_API bool SYSTEM_GetGSenorValue(PGSensor_PARAMS pGSensor_Params);
```

**Parameters**

*pGSensor\_Params*

Pointer to a GSensor\_PARAMS structure to be filled in with the Gyro Sensor common parameters

**Return Value**

Nonzero indicates success. Zero indicates failure.

**Remarks**

This function can only be used in M3 SMART WM

**See Also**

SYSTEM\_SetGSensorValue

**For .Net**

Namespace : SystemNet.System

Function : Bool GetGSensorValue(out GSensor\_PARAMS pGSensor\_PARAMS);

#### **Example**

```
PGSensor_PARAMS pgSensor = new GSensor_PARAMS();  
SYSTEM_GetGSensorValue(pgSensor)  
m_nGsensorMode = pgSensor->GsensorEnable;  
m_bBacklight_OFF = pgSensor->MotionDisplayOff;  
m_bBacklight_ON = pgSensor->MotionDisplayWakeup;  
m_bPOWER_OFF = pgSensor->MotionSleepOn;  
m_bMotionSleepOnPrevent = pgSensor->MotionSleepOnPrevent;  
m_bDisplay_Rotate = pgSensor->DisplayRotate;  
delete pgSensor
```

### **4.10.12 SYSTEM\_GetOSVersionInfo**

#### **Description**

Gets the Information of OS Version

#### **Syntax**

```
SYSTEM_API bool SYSTEM_GetOSVersionInfo(TCHAR* strVersionInfo);
```

#### **Parameters**

*szGUID*

Gets the GUID Value.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

None

#### **See Also**

None

#### **For .Net**

Namespace : SystemNet.System

Function : String GetGuid();

#### **Example**

```
TCHAR strGUID[1024] = {0,};  
SYSTEM_GetGUID(strGUID);  
m_strGuid.Format(L"%s", strGUID);
```

### 4.10.13 SYSTEM\_GetPhoneVolumeLevel

#### Description

Gets the Phone Volume Level.

#### Syntax

```
SYSTEM_API void SYSTEM_GetVolumeLevel ();
```

#### Parameters

None

#### Return Value

The return value is PhoneVolumeLevel.

#### Remarks

None

#### See Also

SYSTEM\_SetVolumeLevel

#### For .Net

Namespace : SystemNet.System

Function : int GetVolumeLevel()

#### Example

```
DWORD dwVolumeLevel_Cur;  
dwVolumeLevel_Cur = SYSTEM_GetVolumeLevel();  
switch (dwVolumeLevel_Cur)  
{  
case 0:  
    dwVolumeLevel_Cur = 5;  
    break;  
case 1:  
    dwVolumeLevel_Cur = 4;  
    break;  
case 2:  
    dwVolumeLevel_Cur = 3;  
    break;  
case 3:  
    dwVolumeLevel_Cur = 2;  
    break;  
case 4:
```

```

        dwVolumeLevel_Cur = 1;
        break;
case 5:
    dwVolumeLevel_Cur = 0;
    break;
default:
    dwVolumeLevel_Cur = 3;
    break;
}

```

#### 4.10.14 SYSTEM\_GetPowerTimeOut

##### Description

Gets the PowerTimeOut.

##### Syntax

```
SYSTEM_API DWORD SYSTEM_GetPowerTimeOut(PWERTYPE type);
```

##### Parameters

*type*

The state is either 0=Battery Status, 1= AC Status.

##### Return Value

The return value is PowerTimeout.

##### Remarks

None

##### See Also

SYSTEM\_SetPowerTimeOut

##### For .Net

Namespace : SystemNet.System

Function : int GetPowerTimeOut(int PowerType)

##### Example

```

DWORD dwPowerTimeout_Cur;
dwPowerTimeout_Cur = SYSTEM_GetPowerTimeOut(BATTERY_POWER);
m_ctlPowerTimeout.SetCurSel(dwPowerTimeout_Cur);

```

#### 4.10.15 SYSTEM\_GetSerialNumber

##### Description

Gets the Serial Number.

### Syntax

```
SYSTEM_API BOOL SYSTEM_GetSerialNumber(TCHAR* strSerial);
```

### Parameters

*strSerial*

Saving SerialNumber

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

None

### For .Net

Namespace : SystemNet.System

Function : string GetSerialNumber();

### Example

```
TCHAR strSerial[1024] = {0,};
```

```
SYSTEM_GetSerialNumber(strSerial);
```

```
m_strSerial.Format(L"Serial Number : %s", strSerial);
```

## 4.10.16 SYSTEM\_GetVersionInfo

### Description

Gets the Information of C++ dll Version

### Syntax

```
SYSTEM_API bool SYSTEM_GetVersionInfo(TCHAR* pszVersion);
```

### Parameters

*pszVersion*

Gets the Information of C++ dll Version

### Return Value

Nonzero indicates success. Zero indicates failure.

### Remarks

None

### See Also

None



**For .Net**

Namespace : SystemNet.System

Function : int GetVersionInfo ()

**Example**

None

## 4.10.17 SYSTEM\_GetVolumeLevel

**Description**

Gets the Volume Level.

**Syntax**

```
SYSTEM_API DWORD SYSTEM_GetVolumeLevel();
```

**Parameters**

None

**Return Value**

The return value is VolumeLevel.

**Remarks**

None

**See Also**

SYSTEM\_SetVolumeLevel

**For .Net**

Namespace : SystemNet.System

Function : int GetVolumeLevel()

**Example**

```
DWORD dwVolumeLevel_Cur;  
dwVolumeLevel_Cur = SYSTEM_GetVolumeLevel();  
switch (dwVolumeLevel_Cur)  
{  
case 0:  
    dwVolumeLevel_Cur = 5;  
    break;  
case 1:  
    dwVolumeLevel_Cur = 4;  
    break;  
case 2:
```

```

        dwVolumeLevel_Cur = 3;
        break;
case 3:
        dwVolumeLevel_Cur = 2;
        break;
case 4:
        dwVolumeLevel_Cur = 1;
        break;
case 5:
        dwVolumeLevel_Cur = 0;
        break;
default:
        dwVolumeLevel_Cur = 3;
        break;
}

```

#### 4.10.18 SYSTEM\_GetWlan

##### Description

Gets the Wlan.

##### Syntax

```
SYSTEM_API BOOL  SYSTEM_GetWlan();
```

##### Parameters

None

##### Return Value

The return value is Wlan Status.

##### Remarks

None

##### See Also

SYSTEM\_SetWlan

##### For .Net

Namespace : SystemNet.System

Function : bool GetWlan()

##### Example

```
if(SYSTEM_GetWlan() == TRUE)
```

```
m_strWlan.Format(L"WLAN : ON");  
else  
    m_strWlan.Format(L"WLAN : OFF");
```

#### 4.10.19 SYSTEM\_KeypadLock

##### Description

Keypad Lock/Unlock.

##### Syntax

```
SYSTEM_API BOOL    SYSTEM_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 .Net

Namespace : SystemNet.System

Function : bool KeypadLock(bool bOn)

##### Example

```
SYSTEM_KeypadLock(TRUE);  
SYSTEM_KeypadLock(FALSE);
```

#### 4.10.20 SYSTEM\_Reboot

##### Description

Reboot.

##### Syntax

```
SYSTEM_API void SYSTEM_Reboot();
```

##### Parameters

None

##### Return Value

None

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : SystemNet.System

Function : void Reboot()

**Example**

```
SYSTEM_Reboot();
```

## 4.10.21 SYSTEM\_SetBacklightlevel

**Description**

Sets the Backlight Level.

**Syntax**

```
SYSTEM_API void SYSTEM_SetBacklightlevel(DWORD m_dwBright);
```

**Parameters**

*m\_dwBright*

Sets the Backlight Level.

**Return Value**

None

**Remarks**

None

**See Also**

SYSTEM\_GetBacklightlevel

**For .Net**

Namespace : SystemNet.System

Function : void SetBacklightlevel(int m\_nBright)

**Example**

```
DWORD dwPos;
```

```
dwPos = m_ctlBackLightLevel.GetPos();
```

```
SYSTEM_SetBacklightlevel(dwPos);
```

## 4.10.22 SYSTEM\_SetBacklightTimeOut

**Description**

Sets the BacklightTimeout.

### Syntax

```
SYSTEM_API BOOL SYSTEM_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

SYSTEM\_GetBackLightTimeout

### For .Net

Namespace : SystemNet.System

Function : bool SetBackLightTimeout(int PowerType, bool bCheck, int nTimeout)

### Example

```
DWORD dwBright;
```

```
SYSTEM_SetBackLightTimeout(BATTERY_POWER, FALSE, dwBright);
```

```
SYSTEM_SetBackLightTimeout(BATTERY_POWER, TRUE, dwBright);
```

```
SYSTEM_SetBackLightTimeout(EXTERNAL_POWER, FALSE, dwACBright);
```

```
SYSTEM_SetBackLightTimeout(EXTERNAL_POWER, TRUE, dwACBright);
```

## 4.10.23 SYSTEM\_SetBluetooth

### Description

Sets the Bluetooth On/Off.

### Syntax

```
SYSTEM_API BOOL SYSTEM_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**

SYSTEM\_GetBluetooth

#### **For .Net**

Namespace : SystemNet.System

Function : bool SetBluetooth(bool bOn)

#### **Example**

```
SYSTEM_SetBluetooth(TRUE);
```

```
SYSTEM_SetBluetooth(FALSE);
```

### **4.10.24 SYSTEM\_SetCpuClock**

#### **Description**

Sets the CPU Clock.

#### **Syntax**

```
SYSTEM_API BOOL    SYSTEM_SetCpuClock(int cpu);
```

#### **Parameters**

*cpu*

Sets the CPU Clock.

#### **Return Value**

Nonzero indicates success. Zero indicates failure.

#### **Remarks**

This function can't be used in M3 Smart, MM3, M3 OX10.

#### **See Also**

SYSTEM\_GetCpuClock

#### **For .Net**

Namespace : SystemNet.System

Function : bool SetCpuClock(int cpu)

#### **Example**

```
SYSTEM_SetCpuClock(0);
```

```
SYSTEM_SetCpuClock(1);
```

SYSTEM\_SetCpuClock(2);

#### 4.10.25 SYSTEM\_SetGSensorValue

##### Description

Sets the Gyro Sensor Value

##### Syntax

```
SYSTEM_API BOOL    SYSTEM_SetGSensorValue(PGSensor_PARAMS pGSensor_PARAMS);
```

##### Parameters

pGSensor\_PARAMS

Pointer to a GSensor\_PARAMS structure holding the Gyro Sensor common parameters

##### Return Value

Nonzero indicates success. Zero indicates failure.

##### Remarks

This function can only be used with M3 SMART WM.

##### See Also

SYSTEM\_GetGSensorValue

##### For .Net

Namespace : SystemNet.System

Function : bool SetGSensorValue(ref m\_GSensor\_Params)

##### Example

```
PGSensor_PARAMS pgSensor = new GSensor_PARAMS();
pgSensor->GsensorEnable = m_nGsensorMode;
pgSensor->MotionDisplayOff = m_bBacklight_OFF;
pgSensor->MotionDisplayWakeup = m_bBacklight_ON;
pgSensor->MotionSleepOn = m_bPOWER_OFF;
pgSensor->MotionSleepOnPrevent = m_bMotionSleepOnPrevent;
pgSensor->DisplayRotate = m_bDisplay_Rotate;
SYSTEM_SetGSensorValue(pgSensor);
```

#### 4.10.26 SYSTEM\_SetPhoneVolumeLevel

##### Description

Sets the Phone Volume Level.

##### Syntax

```
SYSTEM_API void SYSTEM_SetPhoneVolumeLevel(DWORD m_nVolume);
```

**Parameters**

*nVolume*

Sets the PhoneVolume Level.

**Return Value**

None

**Remarks**

None

**See Also**

SYSTEM\_GetPhoneVolumeLevel

**For .Net**

Namespace : SystemNet.System

Function : void SetPhoneVolumeLevel(int m\_nVolume)

**Example**

```
SYSTEM_SetPhoneVolumeLevel(5);
```

## 4.10.27 SYSTEM\_SetPowerTimeOut

**Description**

Sets the PowerTimeOut.

**Syntax**

```
SYSTEM_API BOOL    SYSTEM_SetPowerTimeOut(POWERTYPE type, int nTimeOut);
```

**Parameters**

*type*

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**

SYSTEM\_GetPowerTimeOut

**For .Net**

Namespace : SystemNet.System

Function : bool SetPowerTimeOut(int PowerType, int nTimeOut);



### Example

```
DWORD dwPower = 0;  
SYSTEM_SetPowerTimeOut(BATTERY_POWER, dwPower);
```

## 4.10.28 SYSTEM\_SetSleepMode

### Description

Sleep On.

### Syntax

```
SYSTEM_API void SYSTEM_SetSleepMode();
```

### Parameters

None

### Return Value

None

### Remarks

None

### See Also

None

### For .Net

Namespace : SystemNet.System

Function : void SetSleepMode()

### Example

```
SYSTEM_SetSleepMode();
```

## 4.10.29 SYSTEM\_SetVolumeLevel

### Description

Sets the Volume Level

### Syntax

```
SYSTEM_API void SYSTEM_SetVolumeLevel(DWORD m_nVolume);
```

### Parameters

*m\_nVolume*

Sets the Volume Level.

### Return Value

None

### Remarks

None

#### See Also

SYSTEM\_GetVolumeLevel

#### For .Net

Namespace : SystemNet.System

Function : void SetVolumeLevel(int m\_nVolume)

#### Example

```
SYSTEM_SetVolumeLevel(1);
```

### 4.10.30 SYSTEM\_SetWlan

#### Description

Sets the Wlan On/Off.

#### Syntax

```
SYSTEM_API BOOL SYSTEM_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

SYSTEM\_GetWlan

#### For .Net

Namespace : SystemNet.System

Function : bool SetWlan(bool bOn)

#### Example

```
SYSTEM_SetWlan(TRUE);
```

```
SYSTEM_SetWlan(FALSE);
```

### 4.10.31 SYSTEM\_Vibrate

#### Description

Vibrate On/Off.

#### Syntax

```
SYSTEM_API BOOL SYSTEM_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 .Net**

Namespace : SystemNet.System

Function : bool Vibrate(bool bOn)

**Example**

```
SYSTEM_Vibrate(TRUE);
```

```
SYSTEM_Vibrate(FALSE);
```

## 4.10.32 SYSTEM\_VolumeMute

**Description**

Volume Mute.

**Syntax**

```
SYSTEM_API void SYSTEM_VolumeMute();
```

**Parameters**

None

**Return Value**

None

**Remarks**

None

**See Also**

None

**For .Net**

Namespace : SystemNet.System

Function : void VolumeMute()

**Example**

```
SYSTEM_VolumeMute();
```

## 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	WM6.1/6.5 CE 5.0	Applicable in Summit module, if 3rd party config is set

#### Precautions

- WLANTest.exe is usually located at \Flash Disk\WLAN.
- WLANTest.exe can be used on devices with Summit WLAN Module.  
To use WlanTray instead of SCU (Summit Utility Client), the Active Profile on SCU must be set to ThirdPartyConfig.

#### 1. First view of the program – AP List View



SSID : Service Set Identifier.

RSSI : Received Signal Strength Indication.

Security

true : Encryption

false : Open.

Power OFF : WLAN tern On/off.

Search : Scan available network.

Connect : Connect to selected network.

## 2. Connecting View for Encryption Type.

WLAN

1

Connect Info

SSID 9F\_Motorola\_30

RSSI -65

Encryption WEP

Password \*\*\*\*\*

Confirm \*\*\*\*\*

Connect to this AP?

Yes No

Connect Info : display View current status of WLAN.

Encryption : Access to an AP with stored settings.

Password : Write password.

Confirm : Confirm password.

Yes : Connect to apply AP info.

No : Cance..

## 3. State View

WLAN

1

ITEM	VALUE
Status	ASSOCIATED
SSID	9F_Motorola_30
Client IP	0.0.0.0
Client Mac	00:17:23:A0:99:BE
AP IP	0.0.0.0
AP Mac	00:15:70:E0:79:10
Beacon	100
DTIM	10
Channel	11
Bit Rate	1 Mbps
TX Power	63 mW

-62 dBm

AP List State Profile About

Status : Current status of WLAN

SSID : Name of connected SSID

Client IP

Client Mac

AP IP

AP Mac

Beacon : Shows periodic time of broad cast signal(beacon) that received from AP as Ksec.

DTIM : Shows how often the AP includes the DTIM(Delivery Traffic Indication Message) when sending signals. If the DTIM is 3, it means that DTIM is included in every third broadcast signal(beacon).

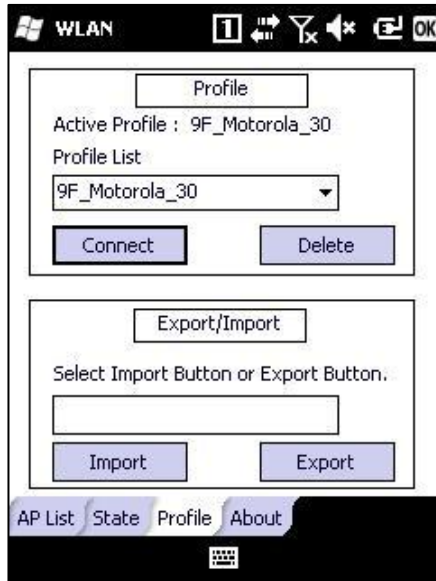
Channel : Connects to channel that AP sets and mostly channel 9 or 11 is used.

Bit Rate : Number of bits proceeded per 1 second.

TX Power

RSSI : Signal Strength,

#### 4. Profile View



Active Profile : Profile of currently connected WLAN.

Profile List : List of stored profiles. Profile is produced automatically if it is connected at least once.

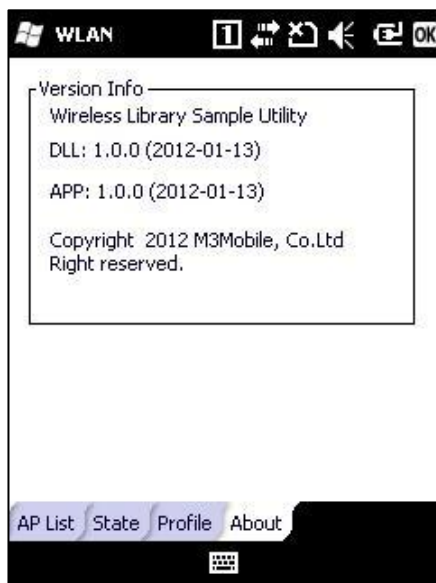
Connect : Connects to selected profile.

Delete : Deletes selected profile

Import : Restores the WLAN connection configuration.

Export : Back up of the WLAN connection configuration.

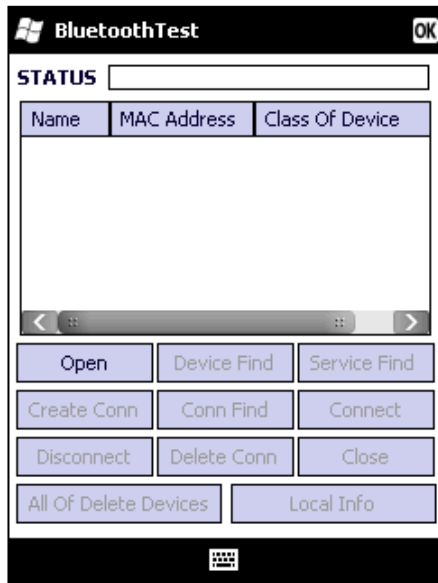
#### 5. About View



Display Version Info

## 5.2 BLUETOOTH

### 1. Main Page



**STATUS** : Status of Bluetooth.

**Open** : Opens Bluetooth COM Port.

**Device Find** : Finds Bluetooth-connectable device.

**Service Find** : Finds service that Bluetooth can be connected.

**Create Conn** : Makes a Connection to connect.

**Conn Find** : Finds a Connection.

**Connect** : Connects Bluetooth devices to each other.

**Disconnect** : Disconnects the connected device.

**Delete Conn** : Deletes the Connection.

**Close** : Closes the Bluetooth COM Port.

**All of Delete Devices** : Deletes all searched devices.

**Local Info** : Provides the device information that runs program.



■ This demo program can be applied to upper level from StonestreetOne BTEplorer Version 2.1.1 and Build Version 27460.

#### ■ Performance procedure

##### 1) If there is no Connection...

Open -> Device Find -> Service Find -> Create Connection -> Connection Find -> Connect -> Disconnect -> (Delete Connection) -> Close

##### 2) If there is Connection...

Open -> Connection Find -> Connect -> Disconnect -> (Delete Connection) -> Close



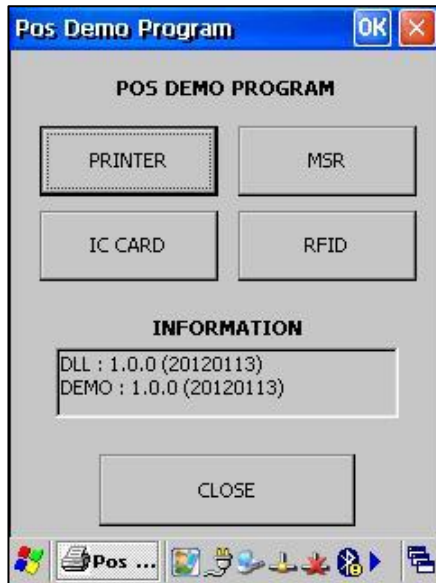
## 5.3 POS

Product	Model	Type	OS	Note
ALL	ALL	Software	ALL	Program for BT printer in M3 Mobile devices

### Precautions

- PosTest.exe is usually located at \Flash Disk\POS.

#### 1. First view of the program



PRINTER : Runs Printer Test Dialog.

MSR : Runs MSR Test Dialog.

IC CARD : Runs IC CARD Test Dialog.

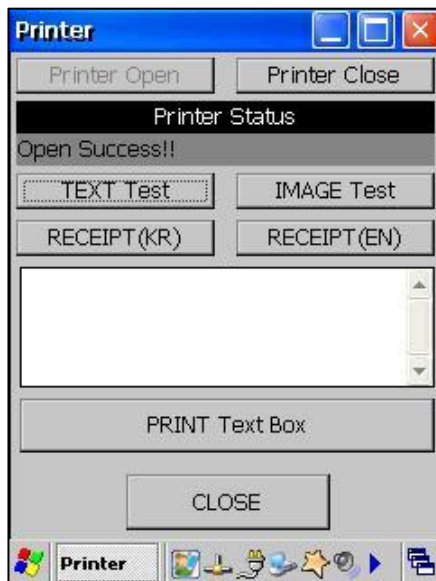
RFID : Runs RFID Test Dialog.

INFORMATION : DLL version info.

Ver : Test program version info

Close : Closes program

#### 2. Printer Dialog.



Printer Open/Close : Printer Module On and Off.

Printer Status : Shows current status of printer.

TEXT Test : Font format can be adjusted but the font cannot.

IMAGE Test : Prints text in the text box.

RECEIPT : Prints receipt sample

PRINT Text Box : Prints text in the text box.

CLOSE : Returns to first Dialog.

### 3. MSR Dialog



MSR Open/Close : MSR module On and Off.

MSR Start/Stop : MSR Reading On and Off.

Card Status : Shows current status of MSR.

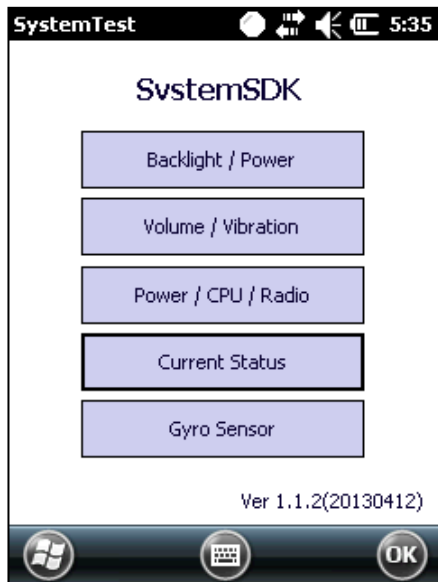
Close : Returns to first Dialog.

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

## 5.4 SYSTEM

Product	Model	Type	OS	Note
M3 OX10	M3 OX10		WM 6.5/CE 6.0	

### 1. Main Page



Backlight / Power

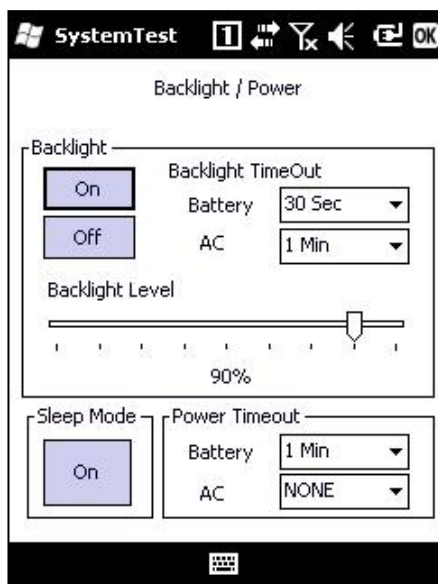
Volume / Vibration

Power / CPU / Radio

Current Status

Gyro Sensor

### 2. Backlight / Power



Backlight

On : Backlight On

Off : Backlight Off

Backlight TimeOut

Battery : Sets the Battery Backlight Timeout

AC : Sets the AC Backlight Timeout

Backlight Level : Backlight Level Control

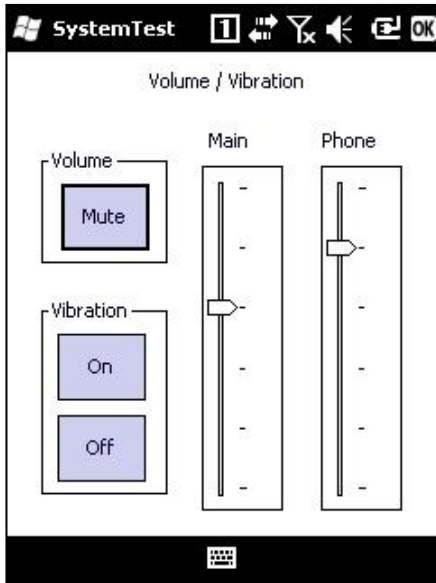
Sleep Mode : Sleep On

Power Timeout

Battery : Sets the Battery Backlight Timeout

AC : Sets the AC Backlight Timeout

### 3. Volume / Vibration



Volume

Mute : Volume Mute

Vibration

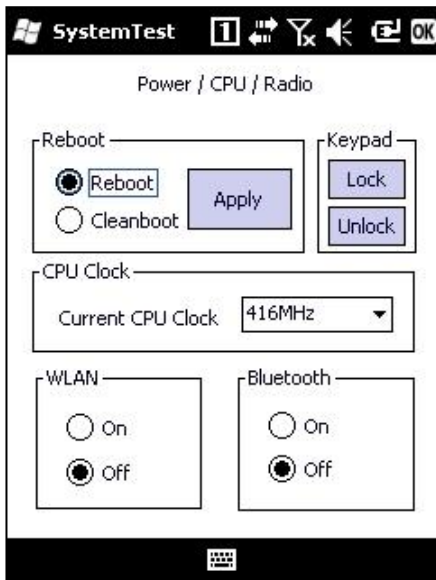
On : Vibration On

Off : Vibration Off

Main Volume : Main Volume Control

Phone Volume : Phone Volume Control

### 4. Power / CPU / Radio



Reboot

Reboot + Apply : Reboot

Cleanboot + Apply : Cleanboot

Keypad

Lock : Keypad Lock

Unlock : Keypad Unlock

CPU Clock : Sets the Current CPU Clock

WLAN

On : Wlan On

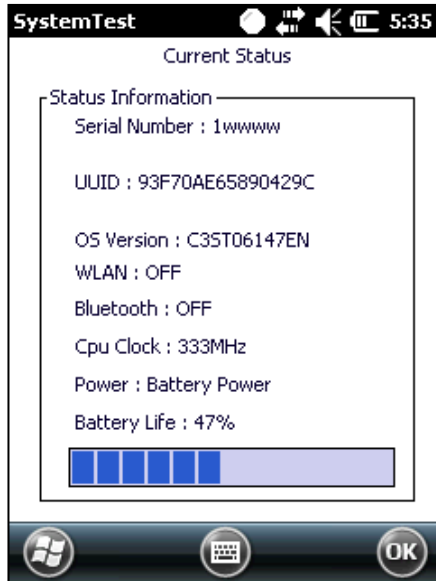
Off : Wlan Off

Bluetooth

On : Bluetooth On

Off : Bluetooth Off

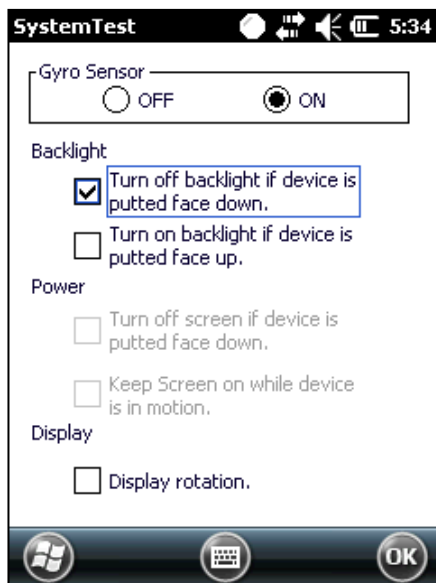
## 5. Current Status



### Status Information

- Serial Number : Gets the Serial Number
- UUID : Gets the UUID
- OS Version : Gets the OS Version
- WLAN : Gets the Wlan Status
- Bluetooth : Gets the Bluetooth Status
- CPU Clock : Gets the Current CPU Clock
- Power : Gets the Power Status(Battery / AC)
- Battery Life : Gets the Current BatteryLife

## 6. Gyro Sensors



### Gyro Sensor

- ON : Gyro Sensor On
- OFF : Gyro Sensor Off

### Backlight

- Turn off backlight
- Turn on backlight

### Power

- Turn off Screen
- Keep Screen

### Display

- Display rotation

## Services

If you experience any trouble while using our product, you can visit **M3 Service center** or send enquires to our **online support web page** (<http://itc.m3mobile.net>), we will do our best to solve your trouble as soon as we can.

M3 FAQ document can help you with troubleshooting.

For any enquires about business program, please contact program provider for faster service.

## Contact Details

### Headquarter

M3 Bldg., 735-45, Yeoksam-Dong, Gangnam-Gu, Seoul, 135-080, Korea  
Tel: +82 2 574 0037 Fax: +82 2 558 1253

[www.m3mobile.net](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>