



# **Programmer's Guide: Using the M3 GREEN SDK**

**Version: 1.2.0.G**

**Release Date: December 06, 2010**

# Table of Contents

Copyright and Agreement .....	5
Development Tools and Requirements .....	5
Release History .....	5
1.0 Introduction.....	6
2.0 Tutorial .....	7
2.1 Bluetooth .....	8
2.1.1 Initialization .....	8
2.1.2 Bluetooth Off.....	10
2.1.3 Device Search.....	11
2.1.4 Service Search .....	12
2.1.5 Connect to Service .....	13
2.2 Camera.....	15
2.2.1 Initialization .....	15
2.2.2 Close Camera .....	15
2.2.3 Capture Image .....	16
2.2.4 Flash On / Off .....	16
2.2.5 Preview Start / Stop.....	17
2.3 CDMA.....	18
2.3.1 PhoneCall.....	18
2.3.2 SendSMS.....	18
2.3.3 HangUp.....	19
2.3.4 DialUp.....	19
2.3.5 GetRasState.....	20
2.3.6 HangUpRas .....	20
2.4 GPS .....	21
2.4.1 Open GPS .....	21
2.4.2 Close GPS.....	21
2.4.3 Receive Message to GPS Module.....	22
2.5 RFID .....	23
2.5.1 Initialization .....	23
2.5.2 Close RFID.....	24
2.5.3 Read Tag.....	25
2.5.4 Write Tag.....	25
2.5.5 Read Mode Change (Continuous or Normal) .....	26
2.6 Scanner 1D .....	28
2.6.1 Initialization .....	28
2.6.2 Default Settings.....	29

2.6.3	ScanRead and GetDecodeData.....	32
2.6.4	Close Scanner .....	34
2.7	Scanner 2D (Imager).....	35
2.7.1	Initialization .....	35
2.7.2	Default Settings .....	36
2.7.3	ScanRead and GetDecodeData.....	36
2.7.4	Close Scanner .....	40
3.0	Samples .....	41
3.1	Bluetooth.....	41
3.2	Camera .....	41
3.3	CDMA .....	41
3.4	GPS .....	41
3.5	RFID .....	41
3.6	Scanner (Software Decoder) .....	42
3.7	Scanner (Hardware Decoder with MC-6400S).....	42
3.8	Imager .....	42
4.0	References (Function Lists).....	43
4.1	Bluetooth .....	44
4.1.1	Reference and Function List for C++.....	45
4.1.2	Reference and Function List for C# .....	54
4.1.3	Error Codes for Bluetooth .....	63
4.2	Camera.....	71
4.2.1	Reference and Function List for C++.....	72
4.2.2	Reference and Function List for C# .....	83
4.3	CDMA .....	91
4.3.1	Reference and Function List for C++.....	92
4.4	GPS .....	102
4.4.1	Reference and Function List for C++.....	103
4.4.2	Reference and Function List for C# .....	107
4.5	RFID .....	111
4.5.1	Reference and Function List for C++.....	112
4.5.2	Reference and Function List for C# .....	119
4.6	Scanner 1D .....	128
4.6.1	Reference and Function List for C++.....	129
4.6.2	Reference and Function List for C# .....	138
4.7	Scanner 2D (Imager).....	162
4.7.1	Reference and Function List for C++.....	163



# 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 and Languages

- Visual Studio 2003 (7.1) (.NET Framework 1.1) – Visual C++, Visual C#, Visual Basic .NET
- Visual Studio 2005 (8.0) (.NET Framework 2.0) – Visual C++, Visual C#, Visual Basic .NET

### Development System Requirements

- Pentium – compatible 500MHz processor or better
- Microsoft Windows 98 / ME / 2000 / XP / 2003
- ActiveSync

### Development Platform Requirements

- M3 GREEN Platform SDK must be installed on your computer to be able to develop softwares using this SDK.
  - After installation, please select 'M3Plus' at the platform selection on EVC.
  - Click [HERE](#) to download M3 GREEN Platform.

## Release History

### M3 GREEN SDK Version 1.2.0

- Re-structured SDK manual (2010.11.03)
  - Separate manual is provided for M3 GREEN
  - Added Tutorial, Samples and References (function lists)

## 1.0 Introduction

This document is a reference guide for the software developer's kit (SDK) for M3 GREEN (WinCE 5.0). This handheld terminal may include up to 7 different modules depending on the specification of the device. For module list and brief description of each module, see below table.

Module Name	Description
Bluetooth	Mainly used to connect to a Bluetooth head set for phone calls or printer. It uses COM4 for BT serial and COM7 for BT connections.
Camera	Used to take a picture of an object.
CDMA	An SMS can be sent or received and it also allows data communication through the network. COM5 is used.
GPS	Gathers information on current location through satellite. GPS can be used as an extension to the compatible vehicle cradle. COM0 is used for vehicle cradle serial.
RFID	Read data from tags or write to the tags. COM8 is used (shared with IrDA)
Scanner	Read 1D and / or 2D barcodes depend on the scanner module option. COM6 is reserved for scanner
WLAN	Enable network connection using Wi-Fi. Samsung or Summit WLAN module is used. <b>(WLAN SDK is NOT included in this document)</b>

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

## 2.0 Tutorial

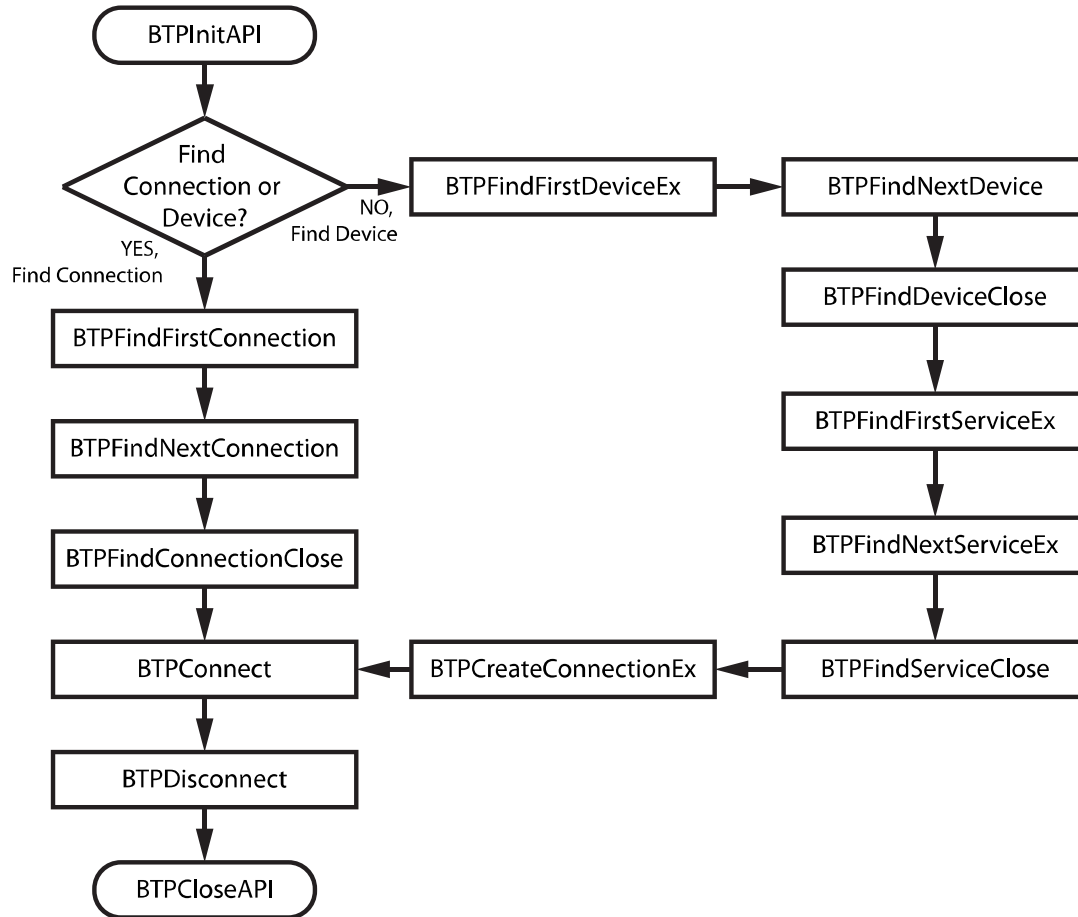
This chapter describes the basic usage of M3 Mobile SDK functions in a step-by-step manner. In this tutorial section, the following topics are treated.

Section	Topic
Bluetooth	Initialization Bluetooth Off Device Search Service Search Connect Service in Current Device
Camera	Initialization Close Camera Capture Image Flash On / Off Preview Start / Stop
CDMA	PhoneCall SendSMS HangUp DialUp GetRasState HangUpRas
GPS	Open GPS Close GPS Receive Message to GPS Module
RFID	Initialization Close RFID Read Tag Write Tag Read Mode Change (Continuous / Normal)
Scanner 1D	Initialization Default Settings ScanRead and GetDecodeData Close Scanner
Scanner 2D	Initialization Default Settings ScanRead and GetDecodeData Close Scanner

## 2.1 Bluetooth

In general, Bluetooth module is included in the PDA as an default option.

Please refer to below flow charts for Bluetooth.



**Bluetooth flow chart**

### 2.1.1 Initialization

In C++, KernelIoControl is used to initialize Bluetooth. In the case of .NET, BTPowerUp and BTOpen functions are used to initialize Bluetooth.

#### C++

```
#include "GenStackInterface.h"
GenStackInterface genStackInterface;

void BluetoothOn()
{
    unsigned char ucSel;
    ucSel = 1;
    KernelIoControl(IOCTL_HAL_BT_POWER_SEL, &ucSel, 1, NULL, 0, NULL);
    // pkfuncs.h has a IOCTL_HAL_BT_POWER_SEL.
}
```



## C#

```
using AveBluetooth; // Name space for atinav Bluetooth

BTCore bTCore;
BT_DEVICE[] devices; //Devices found in the device search
BT_SERVICE[] services; // Services of the selected device found in the service
search

void btOn()
{
    bool retVal = false;
    string strMsg;
    // Initializes the Bluetooth Hardware and the Bluetooth SDK

    // You have to power-up the Bluetooth hardware before using any of
    // the SDK apis, if the Bluetooth hardware is built-in with the device.

    // Power-up Bluetooth

    retVal = bTCore.BTPowerUp();

    if (retVal)
    {
        // Bluetooth hardware successfully powered
        // Now we can initialize the Bluetooth hardware and SDK using the
        // SDK APIs.

        if (bTCore.BTIsInitialized())
        {
            // Open an instance of the SDK
            retVal = bTCore.BTOpen();

            if (retVal)
                strMsg = "BTOpen success";
            else
            {
                strMsg = "BTOpen failed";
            }
        }
        else
        {
            // Bluetooth not initialised...Trying to initialize...
            // Set the Bluetooth transport type of Bluetooth hardware
            // M3 builtin chip is BCSP.
            retVal = bTCore.BTSetTransportLayer(BTCore.TL_BCSP);
            if (retVal)
            {
                strMsg = "BTSetTransportLayer success";
                // Transport Successfully set.
                // Now specify the Transport specific settings,
                // such as port and baudrate to communicate with
                // the Bluetooth hardware(If it has UART/BCSP interface)

                retVal = bTCore.BTSetBCSPDefaultPortParams(PORT_NAME, BT_BAUDRATE);
                if(retVal)
                    strMsg = "BTSetBCSPDefaultPortParams success";
                else
                {
                    strMsg = "BTSetBCSPDefaultPortParams failed";
                }
            }
            else
            {

```

```

        strMsg = "BTSetTransportLayer failed";
    }

    // The above two settings can also failed if it has already been set
    // by some other application. So to confirm this, Get the error code
    // using BTGetErrorCode() API. if the error is 'already initialized',
    // we can proceed... To do.

    // Now try to initialize Bluetooth with the specified settings.
    // In an multi-application scenario, this method is confined to call
    // only by the control application.

    // If the avelinBT application Suite is active in the system,
    // just open an instance of the stack using BTOpen().

    // Initialize Bluetooth SDK
    retVal = bTCore.BTInit();
    if (retVal)
        strMsg = "BTInit success";
    else
    {
        strMsg = "BTInit failed";
    }
}

if(retVal)
{
    // Initialization success
    bTCore.GAPSetDiscoverableMode(true, null, 1);
    //Setting the device to Discoverable mode
    bTCore.GAPSetConnectableMode(true);
    //Setting the device to Connectable mode
    bTCore.GAPSetLocalDeviceName(Device_Name); //Setting the device name
    btOn = true;

    //class representing the message window to receive asynchronous
    //messages from the BT SDK
    //Registering the event for security notififcation

    //Setting the Security Notification function to get security
    //notifications(by invoking SecurityNotifyProc)during authentication
    //and authorization processes.

    BTCORE.SecurityNotifyProc proc = new
    BTCORE.SecurityNotifyProc(SecurityNotifyProc);
    retVal = bTCore.GAPSetSecurityNotifyProc(proc);
    strMsg = "Bluetooth Initialized";
}
}
else
{
    strMsg = "BTPowerUp failed...";
}
}

```

### 2.1.2 Bluetooth Off

In C++, KernerlloControl is used to turn off BT as in BT initialization. In .NET, BTDeInit must run before BTPowerDown to turn off BT. The operation is performed in the opposite way as done in initialization.

#### C++

```

void BluetoothOff
{

```

```

    unsigned char ucSel;
    ucSel = 0;
    KernelIoControl(IOCTL_HAL_BT_POWER_SEL, &ucSel, 1, NULL, 0, NULL);
}

```

## C#

```

void BtOff()
{
    bTCore.BTDeInit();
    bTCore.BTPowerDown();
    strMsg = "Bluetooth not initialized";
}

```

### 2.1.3 Device Search

DeviceSearch, as its name indicates, searches available devices around M3 GREEN unit. In order to search all available devices, it refers to GapInquireAndRetrieveDevice function.

## C++

```

void DeviceSearch()
{
    BT_DEVICE devices[15] = {0};
    UINT8 retCount = 0;
    INT I = 0;
    CString strMsg;

    if(genStackInterface.GapInquireAndRetrieveDevices(BT_GIAC,6,devices,10,&retCount))
    {
        CString cstrDevAddress;
        INT iListIndex = 0;

        if(retCount)
        {
            for(i=0;i<retCount;i++)
            {
                cstrDevAddress.Format(TEXT("%.2X:%.2X:%.2X:%.2X:%.2X:%.2X"),\
                    devices[i].bdAddress[0],devices[i].bdAddress[1],devices[i].bdAddress[2],\
                    devices[i].bdAddress[3],devices[i].bdAddress[4],devices[i].bdAddress[5]);

                DUN_SERVICE_HEAD *deviceNode = new DUN_SERVICE_HEAD;

                memset(&deviceNode->btService,0,sizeof(BT_SERVICE));
                memcpy(deviceNode->btDevice.bdAddress,devices[i].bdAddress,6);
                memcpy(deviceNode->btDevice.clockOffset,devices[i].clockOffset,2);
                deviceNode->btDevice.majorDeviceClass = devices[i].majorDeviceClass;
                deviceNode->btDevice.minorDeviceClass = devices[i].minorDeviceClass;
                deviceNode->btDevice.pageScanMode = devices[i].pageScanMode;
                deviceNode->btDevice.pageScanPeriodMode = devices[i].pageScanPeriodMode;
                deviceNode->btDevice.pageScanRepetitionMode = devices[i].pageScanRepetitionMode;
                deviceNode->btDevice.serviceClass = devices[i].serviceClass;
            }
            strMsg.Format(TEXT("%d Devices found"),retCount);
        }
        else
        {
            strMsg.Format(TEXT("No Device Found.));

```

```

    }
    else
    {
        strMsg.Format(TEXT("Device Search - Fail"));
    }
}

```

## C#

```

void DeviceSearch()
{
    bool retVal = false;
    string strResult = "Searching for devices....";

    devices = new BT_DEVICE[6];

    for (int i = 0; i < 6; i++)
    {
        devices[i] = new BT_DEVICE();
        devices[i].bdAddress = new byte[6];
        devices[i].clockOffset = new byte[2];
    }
    byte retCount;

    // Call the SDK Api to find the devices in the vicinity
    retVal = bTCore.GAPInquireAndRetrieveDevices(BTCore.BT_GIAC, 6, ref devices,
6, out retCount);
    if (retVal)
    {
        if (retCount > 0)
        {
            for (int i = 0; i < retCount; i++)
            {
                string[] address = new string[6];
                for (int j = 0; j < 6; j++)
                {
                    address[j] = String.Format("{0:x2}", devices[i].bdAddress[j]);
                    strResult = "Devices Found...";
                }
            }
        }
        else
            strResult = "No device found in the vicinity";
    }
    else
    {
        strResult = "Device search failed";//Device Inquiry failed
    }
}

```

### 2.1.4 Service Search

BT enabled devices normally provide one or more services that can be used through BT. In such devices, the service provided by the device must be identified. SdapSearchRemoteService function allows to search the services provided by the detected devices.

## C++

```

void OnServiceSearch()
{
    BT_DEVICE device = {0};
    PDUN_SERVICE_HEAD dunHead = NULL;
    BT_UUID uuid = {{0x11, 0x01}, 2};
    BT_SERVICE services[10] = {0};
}

```

```

UINT16 numServices = 0;
INT i =0 , j = 0;

// Before select a Bt device and setting variable device.
if(genStackInterface.SdapSearchRemoteServices(&device, &uuid, 1, services, 5,
&numServices))
{
    if(numServices)
    {
        CString strItemText = services[0].serviceName;
        dunHead->btService.identifier = services[0].identifier;
        dunHead->btService.profileDescriptor = services[0].profileDescriptor;
        dunHead->btService.protocolDescriptor = services[0].protocolDescriptor;
        strcpy((char*)dunHead->
        btService.serviceName,(char*)services[0].serviceName);
        GetDlgItem(IDC_STATUS)->SetWindowText(TEXT("Service found.));
    }
    else
    {
        GetDlgItem(IDC_STATUS)->SetWindowText(TEXT("No Service Found.));
        m_devlistBt.SetItemText(index,1,L"No Service found");
    }
    else
    {
        GetDlgItem(IDC_STATUS)->SetWindowText(TEXT("Service Search - Fail "));
    }
}

```

## C#

```

void servsearch()
{
    bool retVal = false;
    string strMsg = " ";

    strMsg = "Searching for services....";

    //Save the index of the selected device from the devices array

    BT_DEVICE device = devices[deviceIndex];
    //Fetching the selected device from the devices array
    BT_UUID newuuid = new BT_UUID();
    newuuid.length = 2;
    newuuid.uuid = new byte[16]; // UUID for SPP
    newuuid.uuid[0] = 0x11;
    newuuid.uuid[1] = 0x01;

    ushort numServices;
    services = new BT_SERVICE[2];
    for (int i = 0; i < 2; i++)
    {
        services[i] = new BT_SERVICE();
        services[i].serviceName = new byte[255];
    }

    // Call the SDK Api to find the services of the selected device
    retVal = btCore.SPPGetRemoteServices(device, newuuid, ref services, 2, out
numServices);
}

```

### 2.1.5 Connect to Service

Select a service that is searched from M3 PDA then, connect to the service. SppConnect function enables to connect M3 terminal with the service in Remote Device

## C++

```
HANDLE sppClient = NULL; // Handle for Serial port client connection

void OnSppConnect()
{
    BT_DEVICE device = {0};
    BT_SERVICE service = {0};
    UINT16 frameSize = 1500;
    PINT8 portName = (PINT8) SPP_COM_Port;

    CString strMsg;

    // After device search and its service search
    // The device is current device
    // The service is that you want to connect in selected device.
    sppClient = genStackInterface.SppConnect(&device, &service, &frameSize,

    NULL,portName);
    if(sppClient == NULL)
    {
        strMsg.Format(L"SPP_Connect - Fail");
    }
    else
    {
        strMsg.Format(L"SPP_Connect - Success");
    }
}
```

## C#

```
void sppconn()
{
    string strMsg = "Connecting....";

    BT_SERVICE selectedService = services[serviceIndex];
    BT_DEVICE selectedDevice = devices[deviceIndex];
    ushort frameSize = 1500;
    string comPort = "COM4:";
    IntPtr sppHandle = IntPtr.Zero;

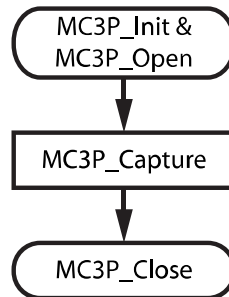
    //Creating serial Connection
    sppHandle = bTCore.SPPConnect(selectedDevice, selectedService, ref
frameSize,comPort);

    if (sppHandle != IntPtr.Zero)
    {
        strMsg = "SPPConnect success";
    }
    else
    {
        strMsg = "SPPConnect failed";
    }
}
```

## 2.2 Camera

This SDK is applicable to both 1.3 mega pixel and 2.0 mega pixel cameras.

Please refer to below flow charts for Bluetooth.



**Simple Camera flow chart**

### 2.2.1 Initialization

In Camera Init, MC3P\_Init and MC3P\_Open function are needed. First, MC3P\_Init function needs to be called. Its parameters are HWND which runs camera and Window Control Handle which outputs the preview. Then, MC3P\_Open starts the camera program by inputting the Window Handle of registered Control.

#### C++

```
CStatic m_ctrlpreview;
CM3P_Camera m_m3p_camera; // CM3P_Camera is the camera dll class.

void CameraInit()
{
    // m_ctrlpreview is a static tool on camera dlg
    m_m3p_camera.MC3P_Init(m_hWnd,m_ctrlpreview.m_hWnd))
    //return device type 0: 6300, 1: 6400, 2:6500

    if(!m_m3p_camera.MC3P_Open())
    {
        AfxMessageBox(L"Com Open Error");
        return FALSE;
    }
}
```

#### C#

```
using M3p_cam_net;
private M3p_cam_net.CamCore camctrl;
public CameraInit()
{
    camctrl = new CamCore();
    camctrl.MC3P_Init(this.Handle, pictureBox1.Handle);

    if(!camctrl.MC3P_Open())
    {
        MessageBox.Show("open error");
    }
}
```

### 2.2.2 Close Camera

If preview is running when trying to terminate the program, preview should be stopped first. Then, the camera program is cleased through Close Camera.

## C++

```
void Camera Close()  
{  
    m_m3p_camera.MC3P_PreViewStop();  
    m_m3p_camera.MC3P_Close();  
}
```

## C#

```
void CameraClose()  
{  
    camctrl.MC3P_Close();  
}
```

### 2.2.3 Capture Image

Capture Image captures the Still Shot of Preview screen.

## C++

```
void CaptureImage()  
{  
    m_m3p_camera.MC3P_Capture();  
}
```

## C#

```
void CaptureImage()  
{  
    camctrl.MC3P_Capture();  
}
```

### 2.2.4 Flash On / Off

Flash On/Off function is helping camera captures.

## C++

```
void CameraFlashOn(BOOL bOn)  
{  
    if(bOn == TRUE)  
    {  
        m_m3p_camera.MC3P_FlashON();  
    }  
    else  
    {  
        m_m3p_camera.MC3P_FlashOFF();  
    }  
}
```

## C#

```
void CameraFlashOn(BOOL bOn)  
{  
    if(bOn == TRUE)  
    {  
        camctrl.MC3P_FlashON();  
    }  
    else  
    {  
        camctrl.MC3P_FlashOFF();  
    }  
}
```



### 2.2.5 Preview Start / Stop

Preview Start/Stop toggles the preview of the image that are input to the camera.

#### C++

```
void PreviewStart(BOOL bOn)
{
    if(bOn == TRUE)
    {
        m_m3p_camera.MC3P_PreviewStart();
    }
    else
    {
        m_m3p_camera.MC3P_PreviewStop();
        m_ctrpreview.Invalidate();
    }
}
```

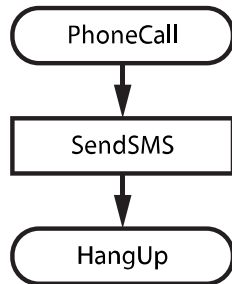
#### C#

```
void PreviewStart(BOOL bOn)
{
    if(bOn == TRUE)
    {
        camctrl.MC3P_PreviewStart();
    }
    else
    {
        camctrl.MC3P_PreviewStop();
        m_ctrpreview.Invalidate();
    }
}
```

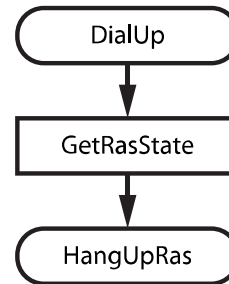
## 2.3 CDMA

To use this SDK, M3 GREEN device must have CDMA module.

Please refer to below flow charts for phone and data communication usage.



Phone flow chart



Data Communication flow chart

### 2.3.1 PhoneCall

**C++**

```
void CTabDlg1::OnBtnCall()
{
    UpdateData();

    m_szPhoneNo.TrimLeft();
    m_szPhoneNo.TrimRight();

    if(m_szPhoneNo.IsEmpty())
    {
        return;
    }

    CString szMsg;
    int nRet = g_pCdmaCmd->PhoneCall(m_szPhoneNo);
    if(nRet != ERROR_NONE)
    {
        szMsg.Format(_T("Error Occured(Error Code : %d)"), nRet);
        ::PostMessage(m_hwndParent, WM_LOG, (LPARAM)szMsg.GetBuffer(0), NULL);
    }
}
```

### 2.3.2 SendSMS

**C++**

```
void CTabDlg1::OnBtnSendSms()
{
    INT    nRet = 0;
    CString szMsg;
    // TODO: Add your control notification handler code here
    UpdateData();
    if(m_szPhoneNo.IsEmpty())
    {
        szMsg = _T("Input phone number!!");
        ::PostMessage(m_hwndParent, WM_LOG, (LPARAM)szMsg.GetBuffer(0), NULL);
        return;
    }
}
```

```

if(m_szSms.IsEmpty())
{
    szMsg = _T("Input SMS content!!");
    ::PostMessage(m_hwndParent, WM_LOG, (WPARAM)szMsg.GetBuffer(0), NULL);
    return;
}

if(m_szCallBack.IsEmpty())
{
    szMsg = _T("Input SMS callback number!!");
    ::PostMessage(m_hwndParent, WM_LOG, (WPARAM)szMsg.GetBuffer(0), NULL);
    return;
}

nRet = g_pCdmaCmd->SendSMS(m_szPhoneNo.GetBuffer(0),
m_szSms.GetBuffer(0),
m_szCallBack.GetBuffer(0),
SMS_PRIORITY_NORMAL);
if(nRet != ERROR_NONE)
{
    szMsg.Format(_T("Error Occured(Error Code : %d)"), nRet);
    ::PostMessage(m_hwndParent, WM_LOG, (WPARAM)szMsg.GetBuffer(0), NULL);
}
}

```

### 2.3.3 HangUp

**C++**

```

void CTabDlg1::OnBtnCallend()
{
    CString szMsg;

    int nRet = g_pCdmaCmd->Hangup();
    if(nRet != ERROR_NONE)
    {
        szMsg.Format(_T("Error Occured(Error Code : %d)"), nRet);
        ::PostMessage(m_hwndParent, WM_LOG, (WPARAM)szMsg.GetBuffer(0), NULL);
    }
}

```

### 2.3.4 DialUp

**C++**

```

void CTabDlg3::OnBtnDialUp()
{
    DWORD dwDial = g_pCdmaCmd->DialUp(this->GetSafeHwnd(), _T("1501"),
_T("sktelecom"), _T(""), _T("Internet"));
    CString szTemp;

    if(dwDial != 0)
    {
        szTemp.Format(L"DialUp Err- %d", dwDial);
        ::PostMessage(m_hwndParent, WM_LOG, (WPARAM)szTemp.GetBuffer(0), NULL);
    }
    else
    {
        szTemp.Format(L"Ras Connecting");
        ::PostMessage(m_hwndParent, WM_LOG, (WPARAM)szTemp.GetBuffer(0), NULL);
    }
}

```

### 2.3.5 GetRasState

**C++**

```
void CTabDlg3::OnBtnRasState()
{
    DWORD dwRet = 0;
    CString szTemp;

    dwRet = g_pCdmaCmd->GetRasState();
    if(dwRet == RASCS_Disconnected)
    {
        szTemp.Format(_T("Ras Disconnected"));
        ::PostMessage(m_hwndParent, WM_LOG, (WPARAM)szTemp.GetBuffer(0), NULL);
    }
    else if(dwRet == RASCS_Connected)
    {
        szTemp.Format(_T("Ras Connected"));
        ::PostMessage(m_hwndParent, WM_LOG, (WPARAM)szTemp.GetBuffer(0), NULL);
    }
    else
    {
        szTemp.Format(L"Ras Status - %d",dwRet);
        ::PostMessage(m_hwndParent, WM_LOG, (WPARAM)szTemp.GetBuffer(0), NULL);
    }
}
```

### 2.3.6 HangUpRas

**C++**

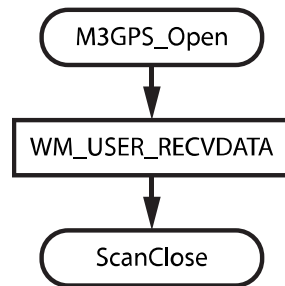
```
void CTabDlg3::OnBtnRasHangup()
{
    BOOL bDial = g_pCdmaCmd->HangUpRas();
    CString szTemp;

    if(bDial == TRUE)
    {
        szTemp.Format(L"Disconnected");
        ::PostMessage(m_hwndParent, WM_LOG, (WPARAM)szTemp.GetBuffer(0), NULL);
    }
}
```

## 2.4 GPS

In M3 GREEN, GPS can only be used via vehicle cradle connection.

Please refer to below flow diagram for GPS



**GPS flow chart**

### 2.4.1 Open GPS

GPS uses serial communication. To open GPS, input the COM port number, Baud Rate for serial communication and Windows HWND for getting GPS data. M3 GREEN uses an external GPS and the COM port is COM8.

#### C++

```
void OpenGps(TCHAR* tzCom, int nBaudRate)
{
    M3GPS_Open(m_hWnd, tzCom, nBaudRate);
}
```

#### C#

```
Class_Gps_Parse cGps;
cGps = new Class_Gps_Parse();

void OpenGps(String strCom, int nBaudRate)
{
    if (!cGps.Gps_Open(MsgWin.Hwnd, strCom, nBaudRate))
    {
        MessageBox.Show("Open Fail");
    }
}
```

### 2.4.2 Close GPS

GPS can be closed through simply closing the opened Serial Port.

#### C++

```
void CloseGps()
{
    M3GPS_Close();
}
```

#### C#

```
void CloseGps()
{
    cGps.Gps_Close();
}
```

```
}
```

### 2.4.3 Receive Message to GPS Module

When GPS Module downloads data from satellite, the data received when opening the GPS is sent to HWND. User can get information through getting WM\_USER\_RECVDATA message and transferring parameter to structure.

#### C++

```
#define WM_USER_RECVDATA      (WM_USER+10000)

// Receive WM_USER_RECVDATA
long OnRecGpsData(WPARAM wParam, LPARAM lParam)
{
    GPSParseInfo *pInf = (GPSParseInfo*)wParam;

    // GPS Information
    ParseGPSMsg(pInf);
    //

    return 0;
}
```

#### C#

```
long OnRecvGpsData(IntPtr wParam, IntPtr lParam)
{
    lass_Gps_Parse.GPS_PARSE_INFO info = new Class_Gps_Parse.GPS_PARSE_INFO();

    try
    {
        info = (Class_Gps_Parse.GPS_PARSE_INFO)Marshal.PtrToStructure(wParam,
            typeof(Class_Gps_Parse.GPS_PARSE_INFO));

        GpsInfoParse(info);
    }
    catch(Exception e)
    {
        MessageBox.Show(e.Message);
        cGps.Gps_Close();
        this.Close();
        return 0;
    }
    return 0;;
}
```

## 2.5 RFID

In M3 GREEN, the supported tag types are:

ISO 15693  
etc...

### 2.5.1 Initialization

In M3 GREEN, RFID uses Ceyon Module. For initializing RFID, supply power to module through KernelIoControl. Then, open COM8 through CAP20\_OpenCommPort. Note that COM8 is used for RFID. Also please register Callback function in CAP20\_RegisterAddRcvData to get data from RFID Module. This example registers through HandleRcvData function. Then, every response from module such as Tag Read/Write is received through this HandleRcvData function.

#### C++

```
#define IOCTL_HAL_RFID_PWR_SEL CTL_CODE(FILE_DEVICE_HAL, 2118 , METHOD_BUFFERED, FILE_ANY_ACCESS)

// Power Supply
void OpenRfid()
{
    unsigned char ucSel = 1;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);

    m_hComHandle = CAP20_OpenCommPort((unsigned short*)LPCTSTR(L"COM8:")) != INVALID_HANDLE_VALUE);
    {
        // Register Callback Func to proceed Frame data from reader
        // Can get the data from RFID through this registered function
        CAP20_RegisterAddRcvData(HandleRcvData);

        //BaudRate Setting
        CAP20_ConfigCommPort(CBR_115200);

        //Single Read Mode Setting
        CAP20_SetConfig(0x01, 0x0b, 0x86);

        //Registry Save
        CAP20_SendCmd(0x01, 0x05);
    }
}
```

#### C#

```
private const uint FILE_DEVICE_HAL = 0x00000101;
private const uint METHOD_BUFFERED = 0;
private const uint FILE_ANY_ACCESS = 0;
private IntPtr INVALID_HANDLE_VALUE = new IntPtr(-1);

// RFID Type(0:maintain current status on sleep (Telefunken)), 1: Always Low status on sleep (HHE)), 2: Always high status on sleep()), 0xFF: Type Read)
public uint IOCTL_HAL_RFID_TYPE_SEL = CTL_CODE(FILE_DEVICE_HAL, 2117, METHOD_BUFFERED, FILE_ANY_ACCESS);

// RFID Power Control(0:Low, 1:High, 0x2:Status)
public uint IOCTL_HAL_RFID_PWR_SEL = CTL_CODE(FILE_DEVICE_HAL, 2118, METHOD_BUFFERED, FILE_ANY_ACCESS);

// RFID Reset Control(0:Low, 1:High, 0x2:Status)
public uint IOCTL_HAL_RFID_RST_SEL = CTL_CODE(FILE_DEVICE_HAL, 2119,
```

```

METHOD_BUFFERED, FILE_ANY_ACCESS);

public static uint CTL_CODE(uint DeviceType, uint Function, uint Method, uint
Access)
{
    return ((DeviceType << 16) | (Access << 14) | (Function << 2) | Method);
}

void OpenRfid()
{
    string strMsg;
    uint unSel = 1, cout = 0, bytesReturn = 0;

    m_ComHandle = INVALID_HANDLE_VALUE;
    m_bContinuousMode = false;

    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, ref unSel, 1, out cout, 0, ref
bytesReturn);

    m_ComHandle = CAP20_mobile.CAP20_OpenCommPort("COM8:",
CAP20_mobile.CBR_9600);

    if (m_ComHandle != INVALID_HANDLE_VALUE)
    {
        CAP20_mobile.FuncAddRcvData fp = new
CAP20_mobile.FuncAddRcvData(HandleRcvData);

        // Register Callback Func to proceed Frame data from reader
        // Can get the data from RFID through this registered function
        CAP20_mobile.CAP20_RegisterAddRcvData(fp);

        CAP20_mobile.CAP20_ConfigCommPort(CAP20_mobile.CBR_115200);

        //Single Read Mode Setting
        CAP20_mobile.CAP20_SetConfig(0x01, 0x0b, 0x86);

        //Registry Save
        CAP20_mobile.CAP20_SendCmd(0x01, 0x05);

        strMsg = "Open success" ;
    }
    else
    {
        strMsg = "Open fail";
    }
}

```

### 2.5.2 Close RFID

RFID can be closed through closing COM port and cut the power to RFID through KernelloControl.

#### C++

```

#define IOCTL_HAL_RFID_PWR_SEL CTL_CODE(FILE_DEVICE_HAL, 2118 , METHOD_BUFFERED,
FILE_ANY_ACCESS)

void CloseRfid()
{
    // Close port
    CAP20_CloseCommPort();
    // not supply power
    unsigned char ucSel = 0;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);
}

```



## C#

```
void CloseRfid()
{
    CAP20_mobile.CAP20_CloseCommPort();
    m_ComHandle = INVALID_HANDLE_VALUE;

    uint unSel = 0, cout = 0, bytesReturn = 0;

    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, ref unSel, 1, out cout, 0, ref
bytesReturn);
}
```

### 2.5.3 Read Tag

Read Tag starts by calling CAP20\_Unselect\_Read function. Once this function is called, RFID Module is ready for reading Tag. In addition, when the tag is read, module calls the HandleRcvData function registered when it was opened and sends data to user. User can use data inputted through pucDataBuf by getting this HandleRcvData function.

## C++

```
void ReadTag()
{
    CAP20_Unselect_Read(0x01, 0, 4)
}

// callback function part. Register in Open function
int HandleRcvData(HANDLE hHandle, UCHAR *pucDataBuf)
{
    // Get data from DLL in pucDataBuf
    return 1;
}
```

## C#

```
void ReadTag()
{
    CAP20_mobile.CAP20_Unselect_Read(0x01, 0, 4);
}

// delegate function part. Register in Open function
public int HandleRcvData(IntPtr hHandle, IntPtr pucDataBuf)
{
    // Get data from DLL in pucDataBuf
    return 1;
}
```

### 2.5.4 Write Tag

Write Tag starts by calling CAP20\_Unselect\_Write function. Once this function is called, RFID module is waiting for tag that will be used as a parameter of the function. If the tag is found, this function inputs data and shows the result through HandleRcvData.

## C++

```
void WriteTag(int nWriteDataLen, BYTE * btWriteData)
{
    CAP20_Unselect_Write(0x01, nWriteDataLen, btWriteData);
}
```

## C#

```
void WriteTag(int nWriteDataLen, Byte [] btWriteData)
```

```
{
    CAP20_mobile.CAP20_Unselect_Write(0x01, (Byte)nWriteDataLen, btWriteData);
}
```

## 2.5.5 Read Mode Change (Continuous or Normal)

There are two read modes for RFID. One is reading consecutively, the other is reading once when it is needed and closing it. To change between these two modes, use CAP20\_SetConfig function after reset RFID power through KernelIoControl.

### C++

```
// RFID Power Control(0:Low, 1:High, 0x2:Status)
#define IOCTL_HAL_RFID_PWR_SEL CTL_CODE(FILE_DEVICE_HAL, 2118 , METHOD_BUFFERED,
FILE_ANY_ACCESS)

void ModeChange()
{
    unsigned char ucSel = 0;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);

    Sleep(500);
    ucSel = 1;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);

    Sleep(1000);
    if(m_bContinuousMode)
    {
        CAP20_SetConfig(0x01, 0x0b, 0x86); // Normal 모드
        Sleep(50);
    }
    else
    {
        CAP20_SetConfig(0x01, 0x1b, 0x04); // Continuous mode
        Sleep(50);

        CAP20_SetConfig(0x01, 0x0b, 0x8c);
        Sleep(50);
    }
}
```

### C#

```
void ModeChange()
{
    uint unSel = 0, cout = 0, bytesReturn = 0;

    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, ref unSel, 1, out cout, 0, ref
bytesReturn);
    Thread.Sleep(500);

    unSel = 1;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, ref unSel, 1, out cout, 0, ref
bytesReturn);
    Thread.Sleep(1000);

    if(m_bContinuousMode == true)
    {
        // Normal mode
        label_Mode.Text = "Read mode: Normal";
        CAP20_mobile.CAP20_SetConfig(0x01, 0x0b, 0x86);
        Thread.Sleep(50);
    }
}
```

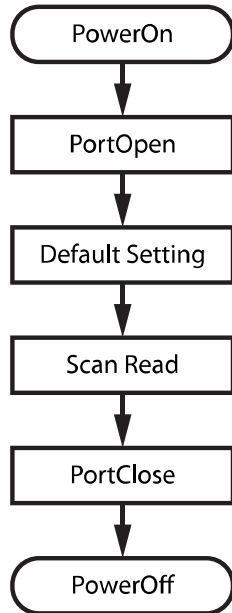
```
else
{
    // continuous
    label_Mode.Text = "Read mode: Continuous";
    CAP20_mobile.CAP20_SetConfig(0x01, 0x1b, 0x04);
    Thread.Sleep(50);

    CAP20_mobile.CAP20_SetConfig(0x01, 0x0b, 0x8c);
    Thread.Sleep(50);
}
}
```

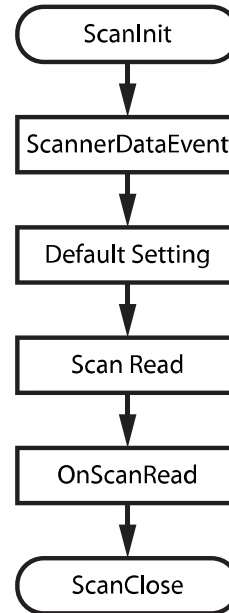
## 2.6 Scanner 1D

This SDK is applicable to 1D scanner modules.

Please refer to below flow charts for C++ and C# usage.



1D Scanner flow chart for C++



1D Scanner flow chart for C#

### 2.6.1 Initialization

Required header file for power control: pkfuncs.h

Scanner initialization must be performed in the order scanner power on then open scanner port.  
Comport Number : 6 / Baudrate : 115200

#### C++

```
[Header File]
#include "KScanBar.h"

CKScan m_KScan;
[CPP File]

#include <pkfuncs.h>
// The Scanner Power On
BOOL bRet = FALSE;
bRet = KernelIoControl(IOCTL_HAL_SCAN_POWER_EN, NULL, NULL, NULL, NULL, NULL);

if(bRet == FALSE)
{
    ::MessageBox(NULL, L"Error : Scanner Power On Failed", NULL, MB_TOPMOST);
    return FALSE;
}
Sleep(200);
// The Scanner Port Open
Int nPort = 6;
bRet = m_KScan.Open(nPort, FALSE, CBR_115200, FALSE, NULL);
if(bRet == FALSE)
```

```

{
    ::MessageBox(NULL, L"Error : Scanner Open Failed", NULL, MB_TOPMOST);
    Return FALSE;;
}

```

In C#, unlike in C++, ScanCtrl.ScanInit function does all necessary steps at once. ScanCtrl.ScanInit() power on the scanner then opens the com port.

Additionally, an event must be created to receive scan data message.  
ScanCtrl.ScannerDataEvent += new ScannerDataDelegate(OnScanRead);

Receiving the data can be done in OnScanRead() function.

## C#

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices; // DllImport
using System.Threading; // Sleep
using MCSSLibNet;

namespace M3ScanTest_Net
{
    public partial class M3Scanner : Form
    {
        private MCSSLibNet.ScannerControl ScanCtrl;
        public int m_bResult;

        public M3Scanner()
        {
            InitializeComponent();
            ScanCtrl = new ScannerControl();
            ScanCtrl.ScannerDataEvent += new ScannerDataDelegate(OnScanRead);
        }

        private void M3Scanner_Load(object sender, EventArgs e)
        {
            // Scanner Initializtion
            m_nResult = ScanCtrl.ScanInit();
            if(m_nResult != 0)
            {
                MessageBox.Show("Error : Scanner Init Failed");
            }
        }
    }
}

```

### 2.6.2 Default Settings

KSCANREAD structure that has scanner options must be initialized.

Initialize TimeOut, MinLen and SecurityLevel.

Receiving type of ScanData: Initialize to Callback type

Initialize options for each barcodes.

## C++

```

void CMainSheet::DefaultSetting()

```

```

{
    // KSCANREAD kRead initialization
    memset(&kRead, 0, sizeof(kRead));
    kRead.nSize = sizeof(kRead);

    // KSCANREADEX2 kReadEx2 initialization
    memset(&kReadEx2, 0, sizeof(kReadEx2));
    strcat(kReadEx2.Signature, "KSCANEX2");

    kRead.nTimeInSeconds    = 10;    // time out
    kRead.nMinLength = 2;    // minimum data length
    kRead.nSecurity         = 1      // security level
    kReadEx2.XmitAIMID      = DISABLE; // Transmit AIMID

    // KScanReadCallBack
    kRead.pUserData         = this;
    kRead.fnCallBack = KScanReadCallBack;
    kRead.dwFlags |= KSCAN_FLAG_WIDESCANANGLE;

    //UPCA
    kReadEx2.UpcA.Enable      = ENABLE;
    kReadEx2.UpcA.Format      = AS_UPCA;
    kReadEx2.UpcA.XmitNumber  = XMIT_NUMBER;
    kReadEx2.UpcA.XmitCheckDigit = XMIT_CHECK_DIGIT;
    kReadEx2.UpcA.Supp        = NO_Supp;

    //UPCE
    kReadEx2.UpcE.Enable      = ENABLE;
    kReadEx2.UpcE.Format      = AS_UPCE;
    kReadEx2.UpcE.XmitNumber  = XMIT_NUMBER;
    kReadEx2.UpcE.XmitCheckDigit = XMIT_CHECK_DIGIT;
    kReadEx2.UpcE.Supp        = NO_Supp;    // Not Supported

    //EAN13
    kReadEx2.Ean13.Enable      = ENABLE;
    kReadEx2.Ean13.Format      = AS_BOOKLAND;    // Including AS_EAN13;
    kReadEx2.Ean13.XmitNumber  = NO_XMIT_NUMBER;
    kReadEx2.Ean13.XmitCheckDigit = XMIT_CHECK_DIGIT;
    kReadEx2.Ean13.Supp        = NO_Supp;

    //EAN8
    kReadEx2.Ean8.Enable      = ENABLE;
    kReadEx2.Ean8.Format      = AS_EAN8;
    kReadEx2.Ean8.XmitNumber  = NO_XMIT_NUMBER;
    kReadEx2.Ean8.XmitCheckDigit = XMIT_CHECK_DIGIT;
    kReadEx2.Ean8.Supp        = NO_Supp;    // Not Supported

    //Code39
    kReadEx2.Code39.Enable      = ENABLE;
    kReadEx2.Code39.MinLength  = 4;
    kReadEx2.Code39.MaxLength  = 30;
    kReadEx2.Code39.SetAscii   = STD_ASCII;
    kReadEx2.Code39.CDV        = DISABLE;
    kReadEx2.Code39.XmitCheckDigit = NO_XMIT_CHECK_DIGIT;
    kReadEx2.Code39.AsCode32    = ENABLE;
    kReadEx2.Code39.AsPZN       = ENABLE;

    //Code128
    kReadEx2.Code128.Enable      = ENABLE;
    kReadEx2.Code128.AsUCCEAN128 = ENABLE;
    kReadEx2.Code128.FNC1_ASCII = FNC1_Ascii; //NULL: No FNC1 conversion
    kReadEx2.Code128.MinLength  = 4;
    kReadEx2.Code128.MaxLength  = 30;

```

```

//Code93
kReadEx2.Code93.Enable      = ENABLE;
kReadEx2.Code93.MinLength   = 4;
kReadEx2.Code93.MaxLength   = 30;

//Code35
kReadEx2.Code35.Enable      = ENABLE;

//Code11
kReadEx2.Code11.Enable      = ENABLE;
kReadEx2.Code11.MinLength   = 4;
kReadEx2.Code11.MaxLength   = 30;
kReadEx2.Code11.CheckDigit  = DIGIT1;
kReadEx2.Code11.XmitCheckDigit = NO_XMIT_CHECK_DIGIT;

//Interleaved 2of5
kReadEx2.Code25.Enable      = ENABLE;
kReadEx2.Code25.MinLength   = 4;
kReadEx2.Code25.MaxLength   = 30;
kReadEx2.Code25.CDV         = DISABLE;
kReadEx2.Code25.XmitCheckDigit = NO_XMIT_CHECK_DIGIT;
kReadEx2.Code25.KindofDecode = (CODE25KIND_INTER |
CODE25KIND_ITF14|CODE25KIND_MATRIX | CODE25KIND_INDUSTY | CODE25KIND_DLOGIC |
CODE25KIND_IATA);

//Codabar
kReadEx2.Codabar.Enable      = ENABLE;
kReadEx2.Codabar.XmitStartStop = NO_XMIT;
kReadEx2.Codabar.MinLength   = 4;
kReadEx2.Codabar.MaxLength   = 30;

//MSI
kReadEx2.Msi.Enable          = ENABLE;
kReadEx2.Msi.CDV             = ENABLE;
kReadEx2.Msi.XmitCheckDigit  = NO_XMIT_CHECK_DIGIT;
kReadEx2.Msi.MinLength       = 4;
kReadEx2.Msi.MaxLength       = 30;

//Plessey
kReadEx2.Plessey.Enable      = ENABLE;
kReadEx2.Plessey.CDV         = ENABLE;
kReadEx2.Plessey.XmitCheckDigit = NO_XMIT_CHECK_DIGIT;
kReadEx2.Plessey.MinLength   = 4;
kReadEx2.Plessey.MaxLength   = 30;

//GS1
kReadEx2.Gs1.Enable          = ENABLE;

//GS1 Limited
kReadEx2.Gs1Limited.Enable = ENABLE;

//GS1 Expanded
kReadEx2.Gs1Expanded.Enable  = ENABLE;

// Telepen
kReadEx2.Telepen.Enable      = ENABLE;
kReadEx2.Telepen.OldStyle    = DISABLE;

kRead.pReadEx=&kReadEx2;
}

```

In C#, unlike in C++, ScanCtrl.Default\_Setting function does all necessary initialization and restores to its

default settings.

- Enable all barcodes.
- TimeOut: 10 / SecurityLevel: 1 / AsyncMode.

## C#

```
ScanCtrl.Default_Setting();
```

### 2.6.3 ScanRead and GetDecodeData

This function reads and get decoded data.

m\_KScan.Read() function is called to Trigger On scanner.  
KScanReadCallBack() function acquires the actual scan data.

## C++

```
int KSCANAPI KScanReadCallBack(LPVOID pRead);
...
void CMainSheet::M3_ScanRead()
{
    CString      Str;
    BOOL         bRet;
    m_bReading = TRUE;

    if (m_bReading)
    {
        // Reading already in progress, now cancel it.
        bRet = m_KScan.ReadCancel();
    }

    m_bReading = TRUE;
    bRet = m_KScan.Read(&kRead);
    if (!bRet) {
        ::MessageBox(NULL, L"Error: ScanRead Failed", NULL, MB_TOPMOST);
        m_bReading = FALSE;
    }
}

int KSCANAPI KScanReadCallBack(LPVOID pRead)
{
    int          Status;
    int          Type;

    CMainSheet* lpCls = (CMainSheet*)((PKSCANREAD)pRead)->pUserData;
    Status = ((PKSCANREAD)pRead)->out_Status;

    switch(Status) {
    case KSCAN_RET_TIMEOUT:           // Timed out.
        Status = 0;
        break;
    case KSCAN_RET_USER_CANCEL:       // User called stop
        Status = 0;
        break;

    case KSCAN_RET_NORMAL:           // Barcode was read
    case KSCAN_RET_TYPE_UNKNOWN:
        Status = 0;
        if(((PKSCANREAD)pRead)->out_Type == -1)
            break;
        if(((PKSCANREAD)pRead)->out_Status == KSCAN_RET_NORMAL)
        {
            if(!lpCls->m_bReading)return 0;
            Type = ((PKSCANREAD)pRead)->out_Type;
            info.gstrCodeTypeName = _T("");
        }
    }
}
```



```

        lpCls->SetBarCodeString(Type);
        info.gstrBarCodeValue = _T("");
        info.gstrBarCodeValue = ((PKSCANREAD)pRead)->out_Barcode;
        ::PostMessage(lpCls->m_hWnd, WM_DATA_READ, 0, 0);

        lpCls->LoadResourceSound();
    }
    lpCls->m_bReading = FALSE;
    break;

    case KSCAN_RET_BAR_NOTFOUND: // Not yet found - Continue.
    case KSCAN_RET_NORMAL_SWEEP: // barcode was read, and security criteria was
not met yet
        Status = 1;
        break;

    default: // Other error.
        Status = 0; // just continue reading until barcode was read with
security criteria met
        break;
    }
    return Status;
}

```

## C#

```

private MCSSLibNet.ScannerControl ScanCtrl;
...
public M3Scanner()
{
    InitializeComponent();
    ScanCtrl = new ScannerControl();
    ...
    ScanCtrl.ScannerDataEvent += new ScannerDataDelegate(OnScanRead);
}
public void ScanRead()
{
    if (m_bReading == true)
    {
        ScanCtrl.ScanReadCancel();
        m_bReading = false;
        return;
    }

    m_bReading = false;
    ScanCtrl.ScanRead();
}

public void OnScanRead(object sender, ScannerDataArgs e)
{
    if (LV_SCANDATA.Items.Count > 6)
        LV_SCANDATA.Items.Clear();

    if (e.ScanData != "")
    {
        ListViewItem ScanData = new ListViewItem();
        ScanData.Text = e.ScanType;
        ScanData.SubItems.Add(e.ScanData);
        LV_SCANDATA.Items.Add(ScanData);

        PlaySound(@"\windows\Alarm1.wav", 0, (int)(SND.SND_ASYNC |
SND.SND_FILENAME));
    }
    m_bReading = false;
}

```

```
}
```

#### 2.6.4 Close Scanner

Terminating the scanner is done in the order closing scanner port the power off the scanner.

##### C++

```
BOOL bRet = FALSE;
Int i = 0;
for(i=0;i<3;i++)
{
    // Scanner Port Close
    bRet = m_KScan.Close();
    if(bRet)
        break;
    Sleep(100);
}
If(bRet == FALSE)
    ::MessageBox(NULL, L"Error: Scanner Close Failed", NULL, MB_TOPMOST);

// Scanner Power Off
for(i=0;i < 3 ; i++)
{
    bRet =KernelIoControl(IOCTL_HAL_SCAN_POWER_DIS, NULL, NULL, NULL, NULL,
NULL);
    if(bRet)break;
    Sleep(100);
}
If(bRet == FALSE)
{
    ::MessageBox(NULL, L"Error: Scanner Power Off Failed", NULL, MB_TOPMOST);
}
```

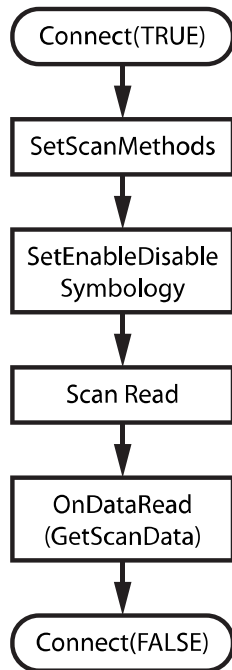
##### C#

```
private void M3Scanner_Closing(object sender, CancelEventArgs e)
{
    ScanCtrl.ScanClose();
}
```

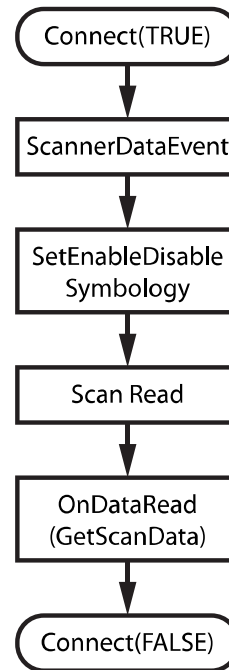
## 2.7 Scanner 2D (Imager)

This SDK is applicable to 2D scanner (imager) modules.

Please refer to below flow charts for C++ and C# usage.



**2D Scanner flow chart for C++**



**2D Scanner flow chart for C#**

### 2.7.1 Initialization

**NOTE:** Imager driver and Camera driver are sharing the same CPU interface. Consequently, 2D scanner and Camera cannot be used at the same time.

#### C++

```
if (Connect(TRUE) != TRUE)
{
    MessageBox(L"Connect Failed");
}
```

#### C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using M3MobileImagerNet;
using System.Runtime.InteropServices;
using System.Threading;
using System.Reflection;
using System.IO;
```

```
namespace Scan3Net
```

```

{
    public partial class Form1 : Form
    {
        private Scanner m_scan;
        private ScannerControl ScanCtrl;

        public Form1()
        {
            InitializeComponent();

            m_scan = new Scanner();
            ScanCtrl = new ScannerControl();

            ScanCtrl.ScannerDataEvent += new ScannerDataDelegate(OnScanData);
            if (!m_scan.Connect(true))
                MessageBox.Show("Connect failed");
        }
    };
}

```

### 2.7.2 Default Settings

Registering an event and Symbology settings.

#### C++

```

SetScanMethods(NULL, m_hWnd, NULL);
SetEnableDisableSymbology(ID_ALL, TRUE);

```

#### C#

```

m_scan.SetEnableDisableSymbology(SYMID.ID_ALL, true);

```

### 2.7.3 ScanRead and GetDecodeData

This function reads and get decoded data.

ScanRead() function is called to Trigger On scanner.  
OnDataRead() function acquires the actual scan data

#### C++

```

BEGIN_MESSAGE_MAP(CMainSheet, CPropertySheet)
//{{AFX_MSG_MAP(CMainSheet)
ON_MESSAGE(WM_SCAN_DATA, OnDataRead)
ON_WM_DESTROY()
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

BOOL CMainSheet::ScanRead()
{
    return ::ScanRead(m_nTimeOut, NULL);
}

void CMainSheet::OnDataRead()
{
    ScanLed(TRUE);

    DECODE_MSG decodeInfo;
    EventType_t eventType;
    GetScanResult(&eventType, &decodeInfo);

    Check_Barcode(decodeInfo.chCodeID, decodeInfo.chSymLetter,
decodeInfo.chSymModifier, decodeInfo.pchMessage);
}

```

```

        Sleep(150);
        ScanLed(FALSE);
    }

```

Create an event for OnScanData(), which acquires scan data, when the program starts.

Barcode type and data is received through ScannerDataArgs which is a parameter of OnScanData().

## C#

```

public Form1()
{
    InitializeComponent();

    m_scan = new Scanner();
    ScanCtrl = new ScannerControl();

    ScanCtrl.ScannerDataEvent += new ScannerDataDelegate(OnScanData);
}

private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.F22)
    {
        if (m_bKeyFlag == false)
        {
            m_bKeyFlag = true;
            Scanner.DECODE_MSG msg = new Scanner.DECODE_MSG();
            int nTimeout = int.Parse(TimoutTextBox.Text);
            m_scan.ScanRead(nTimeout, ref msg);
        }
    }
}

private void Form1_KeyUp(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.F22)
    {
        if (m_bKeyFlag == true)
        {
            if (m_bSyncMode == false)
                m_scan.CancelIO();

            m_bKeyFlag = false;
        }
    }
}

private void OnScanData(object sender, ScannerDataArgs e)
{
    m_scan.GetScanResult(ref m_scan.m_event, ref m_scan.m_msg);
    ListViewItem BarcodeItem = new ListViewItem();
    switch (m_scan.m_msg.chSymLetter)
    {
        case '<':
            BarcodeItem.Text = "Code 32";
            break;
        case 'l':
            BarcodeItem.Text = "Code 49";

```

```

        break;
    case 'M':
        BarcodeItem.Text = "Code 4CB";
        break;
    case 'w':
        BarcodeItem.Text = "DATA Matrix";
        break;
    case 'j':
        if (m_scan.m_msg.chSymModifier == '4')
            BarcodeItem.Text = "ISBT 128";
        else
            BarcodeItem.Text = "Code 128";
        break;
    case 'J':
        BarcodeItem.Text = "Japan POST";
        break;
    case 'K':
        BarcodeItem.Text = "KIX POST";
        break;
    case 'm':
        BarcodeItem.Text = "MATRIX 2of5";
        break;
    case 'x':
        BarcodeItem.Text = "MaxiCode";
        break;
    case 'g':
        BarcodeItem.Text = "MSI";
        break;
    case 'R':
        BarcodeItem.Text = "Micro PDF417";
        break;
    case 'L':
        BarcodeItem.Text = "Planet Code";
        break;
    case 'n':
        BarcodeItem.Text = "Plessey Code";
        break;
    case 'W':
        BarcodeItem.Text = "PosiCode";
        break;
    case 'P':
        BarcodeItem.Text = "Postnet";
        break;
    case 's':
        BarcodeItem.Text = "QR Code";
        break;
    case 'f':
        if (m_scan.m_msg.chCodeID == 'R')
            BarcodeItem.Text = "Straight 2of5 IATA";
        else
            BarcodeItem.Text = "Straight 2of5 Industria";
        break;
    case 'T':
        BarcodeItem.Text = "TCIF Linked Code39";
        break;
    case 't':
        BarcodeItem.Text = "Telepen";
        break;
    case '=':
        BarcodeItem.Text = "Trioptic Code";
        break;
    case 'A':
        BarcodeItem.Text = "Austria POST";
        break;

```

```
case 'z':
    BarcodeItem.Text = "Aztec Code";
    break;
case 'Z':
    BarcodeItem.Text = "Aztec Mesa";
    break;
case 'B':
    BarcodeItem.Text = "British POST";
    break;
case 'C':
    BarcodeItem.Text = "Canada POST";
    break;
case 'Q':
    BarcodeItem.Text = "China POST";
    break;
case 'q':
    BarcodeItem.Text = "Codablock F";
    break;
case 'a':
    BarcodeItem.Text = "Codabar";
    break;
case 'h':
    BarcodeItem.Text = "Code11";
    break;
case 'o':
    BarcodeItem.Text = "Code 16K";
    break;
case 'b':
    BarcodeItem.Text = "Code39";
    break;
case 'D':
    BarcodeItem.Text = "EAN8";
    break;
case 'd':
    BarcodeItem.Text = "EAN13";
    break;
case 'e':
    BarcodeItem.Text = "Interleaved 2of5";
    break;
case '?':
    BarcodeItem.Text = "KOREA POST";
    break;
case 'r':
    BarcodeItem.Text = "PDF417";
    break;
case 'c':
    BarcodeItem.Text = "UPC-A";
    break;
case 'E':
    if (m_scan.m_msg.chCodeID == 'E')
        BarcodeItem.Text = "UPC-E";
    else
        BarcodeItem.Text = "UPC-E1";
    break;
case 'i':
    BarcodeItem.Text = "Code93";
    break;
case 'y':
    BarcodeItem.Text = "EAN/UCC Composite or Reduced Space Symbolology";
    break;
case 'I':
    BarcodeItem.Text = "GS1-128";
    break;
case 'O':
```

```

        BarcodeItem.Text = "OCR";
        break;
    case 'N':
        BarcodeItem.Text = "UPU 4 State ID Tag";
        break;
    default:
        BarcodeItem.Text = "ETC";
        break;
    }
}
}

```

#### 2.7.4 Close Scanner

Terminates Imager and unload the imager driver.

##### C++

```
Connect(FALSE);
```

##### C#

```

private void Form1_Closing(object sender, CancelEventArgs e)
{
    m_scan.Connect(false);
}

```



## 3.0 Samples

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

### 3.1 Bluetooth

Type	Demo	Source	Version	Date	Tool
C++	M3P_CameraTest.exe M3P_Camera.dll	M3P_CameraTest.exe	1.6.3	2010-11-25	eMbedded Visual C++ 4.0
C#.Net	M3P_Cam_Net_Test.exe M3p_cam_net.dll M3P_Camera.dll	M3P_Cam_Net_Test.exe	1.0.1	2010-02-01	eMbedded Visual C++ 4.0

### 3.2 Camera

Type	Demo	Source	Version	Date	Tool
C++	M3P_CameraTest.exe M3P_Camera.dll	M3P_CameraTest.exe	1.6.3	2010-11-25	eMbedded Visual C++ 4.0
C#.Net	M3P_Cam_Net_Test.exe M3p_cam_net.dll M3P_Camera.dll	M3P_Cam_Net_Test.exe	1.0.1	2010-02-01	eMbedded Visual C++ 4.0

### 3.3 CDMA

Type	Demo	Source	Version	Date	Tool
C++	ExtlCmdTest.exe ExtlCDMACmd.dll MCDBEng.dll	ExtlCmdTest.exe	1.0.0	2009-11-07	eMbedded Visual C++ 4.0
C#.Net	ExtlCDMACmdTest.exe ExtlCDMACmd.dll MCDBEng.dll	ExtlCmdTest.exe	1.0.0	2009-11-07	Visual Studio 2005

### 3.4 GPS

Type	Demo	Source	Version	Date	Tool
C++	GpsParseDemo.exe M3GpsParse.dll	GpsParseDemo.exe	1.0.3	2010-11-05	Visual Studio 2005
C#.Net	GpsParseDemoNet.exe M3GpsParse.dll	GpsParseDemoNet.exe	1.0.2	2010-06-07	Visual Studio 2005

### 3.5 RFID

Type	Demo	Source	Version	Date	Tool
C++	TestStataion.exe CAP20_Mobile.dll	TestStataion.exe	1.0.2	2010-09-30	eMbedded Visual C++ 4.0
C#.Net	RFID_Demo_net.exe CAP20_Mobile.dll	RFID_Demo_net.exe	1.0.0	2009-10-26	Visual Studio 2005

### 3.6 Scanner (Software Decoder)

Type	Demo	Source	Version	Date	Tool
C++	M3ScanTest.exe ScanEmul.exe KScanBar.dll	M3ScanTest.exe	3.0.0	2010-06-21	eMbedded Visual C++ 4.0
C#.Net	M3ScanTest_Net.exe MCSSLib.dll MCSSLibNet.dll	M3ScanTest_Net.exe	3.1.0	2010-10-27	Visual Studio 2005
VB.Net	M3ScanTest_VB.exe MCSSLib.dll MCSSLibNet.dll	M3ScanTest_VB.exe	2.1.0	2010-10-27	Visual Studio 2005

### 3.7 Scanner (Hardware Decoder with MC-6400S)

Type	Demo	Source	Version	Date	Tool
C++	M3PlusScanTest.exe M3Plus_ScanEmul.exe KScanBar.dll	M3PlusScanTest.exe	2.2.0	2010-11-24	eMbedded Visual C++ 4.0
C#.Net	M3PlusCE5NetTest.exe M3CPSSLib.dll M3PSSLibNetCla.dll	M3PlusCE5NetTest.exe	1.0.0	2009-09-22	Visual Studio 2005
VB.Net	M3PlusScanTest_VB.exe M3CPSSLib.dll M3PSSLibNetCla.dll	M3PlusScanTest_VB.exe	1.0.0	2009-09-22	Visual Studio 2005

### 3.8 Imager

Type	Demo	Source	Version	Date	Tool
C++	M3ScanTest.exe ScanEmul.exe TestCam.exe M3MobileImager.dll	M3ScanTest.exe	2.1.0	2010-04-22	eMbedded Visual C++ 4.0
C#.Net	M3ScanTest_Net.exe TestCam_Net.exe M3MobileImager.dll M3MobileImagerNet.dll	TestCam.exe	2.0.0	2010-03-08	eMbedded Visual C++ 4.0

## 4.0 References (Function Lists)

This chapter provides references of the modules included in M3 GREEN. APIs are described using C language. Applications are created using Visual C++ 6.0 and they are compatible with Visual C++ 7.0 / 7.1 / 8.0 or higher.

For accessing C / C++ style API in Visual Basic 6.0, we supply M3 Mobile SDK 1.2.0 Type Library for Visual Basic 6.0. Users can add this type library to their projects using browse button in the References dialog (drop down the Project menu and select the References item).

## 4.1 Bluetooth

This section provides description of the functions and DLLs which are used to manage the Bluetooth module.

### Required Products

#### For C++

Required header:

```
albtcore.h
albterrorcodes.h
albtradio.h
albttypes.h
```

Required lib:

```
albtcore.lib
albtradio.lib
```

Required DLL:

Does not require DLL in Bluetooth

#### For C#

Required DLL:

```
albtcoreNet.dll
```

### Supported Product

M3 GREEN with AveLinkBT

## 4.1.1 Reference and Function List for C++

### Definitions

#### Constant Values

```
#define BT_DEVICE_Name "Atinav BlueCE-II" //Device name to set

// M3 Builtin Bluetooth HCI port
#define HCI_UART_Port "COM7:" // Port on which bluetooth card is listening
#define HCI_BAUDRATE 921600 // Baud rate of the bluetooth card

// Emulated COM port by avelink Bluetooth SDK
#define SPP_COM_Port "COM3:"
```

### Structure

#### DUN\_SERVICE\_HEAD

```
typedef struct _DUN_SERVICE_HEAD
{
    BT_DEVICE btDevice;
    BT_SERVICE btService;
} DUN_SERVICE_HEAD, *PDUN_SERVICE_HEAD;
```

#### BT\_DEVICE

```
typedef struct _BT_DEVICE /* BT Device structure */
{
    BD_ADDRESS bdAddress; //Address of the devcie
    UINT8 pageScanRepetitionMode; //Page scan repetetion mode
    UINT8 pageScanPeriodMode; //Page scan period mode
    UINT8 pageScanMode; //Page scan mode
    UINT16 serviceClass; //Service class field
    UINT8 majorDeviceClass; //Device class(Major) feild
    UINT8 minorDeviceClass; //Device class(Minor) feild
    UINT8 clockOffset[2]; //Clock offset
} BT_DEVICE, *PBT_DEVICE;
```

The **BT\_DEVICE** structure defines the characteristics of a Bluetooth device.

### Members

#### *bdAddress*

Specifies address of the Bluetooth device.

#### *pageScanRepetitionMode*

Specifies page scan repetition mode of the Bluetooth device. This member can be one of the following values.

Value	Meaning
PSRM_R0	Page scan repetition mode R0
PSRM_R1	Page scan repetition mode R1
PSRM_R2	Page scan repetition mode R2

#### *pageScanPeriodMode*

Specifies page scan period mode of the Bluetooth device. This member can be one of the following values.

Value	Meaning
PSRM_P0	P0
PSRM_P1	P1
PSRM_P2	P2

#### *pageScanRepetitionMode*

Specifies page scan mode of the Bluetooth device. This member can be one of the following values.

Value	Meaning
PSM_MANDATORY	Mandatory page scan mode
PSM_OPTIONAL_I	Optional page scan mode I
PSM_OPTIONAL_II	Optional page scan mode II
PSM_OPTIONAL_III	Optional page scan mode III

#### *serviceClass*

Specifies service class of the Bluetooth device. This member can be one of the following values.

Value	Meaning
BT_SC_UNSPECIFIED	Unspecified service class
BT_SC_LIMITED_DISCOVERABLE_MODE	Limited discoverable mode service class
BT_SC_NETWORKING	Networking service class
BT_SC_RENDERING	Rendering service class
BT_SC_CAPTURING	Capturing service class
BT_SC_OBJECT_TRANSFER	Object Transfer service class
BT_SC_AUDIO	Audio service class
BT_SC_TELEPHONY	Telephony service class
BT_SC_INFORMATION	Information service class

#### *majorDeviceClass*

Specifies the minor device class of the Bluetooth device. This member can be one of the following values depending on the major device class.

Value	Meaning
BT_MAJOR_MISC	Miscellaneous
BT_MAJOR_COMPUTER	Computer (Desktop, Notebook, ...)
BT_MAJOR_PHONE	Phone (Cellular, Codeless, ...)
BT_MAJOR_LAP	LAN/Network access point.
BT_MAJOR_AUDIO	Audio/video
BT_MAJOR_PERIPHERAL	Peripheral (Mouse, ...)

#### *minorDeviceClass*

Specifies the minor device class of the Bluetooth device. This member can be one of the following values depending on the major device class.

1. Major device class: BT\_MAJOR\_COMPUTER

Value	Meaning
BT_MINOR_UNCLASSIFIED	Uncategorized.
BT_MINOR_DESKTOP	Desktop workstation.
BT_MINOR_SERVER	Server-class computer
BT_MINOR_LAPTOP	Laptop
BT_MINOR_HANDHELD	Handheld PC/PDA
BT_MINOR_PALM	Wearable computer

2. Major device class: BT\_MAJOR\_PHONE

Value	Meaning
BT_MINOR_UNCLASSIFIED	Uncategorized.
BT_MINOR_CELLULAR	Cellular
BT_MINOR_CORDLESS	Cordless
BT_MINOR_SMARTPHONE	Smart phone
BT_MINOR_WIREDMODEM	Wired modem

3. Major device class: BT\_MAJOR\_LAP

Value	Meaning
<b>BT_MINOR_UNCLASSIFIED</b>	Uncategorized.
<b>BT_MINOR_FULLY_AVAILABLE</b>	Service fully available
<b>BT_MINOR_17</b>	Service 17% utilized
<b>BT_MINOR_33</b>	Service 33% utilized
<b>BT_MINOR_50</b>	Service 50% utilized
<b>BT_MINOR_67</b>	Service 67% utilized
<b>BT_MINOR_99</b>	Service 99% utilized
<b>BT_MINOR_NO_SERVICE</b>	Service not service

#### 4. Major device class: BT\_MAJOR\_AUDIO

Value	Meaning
<b>BT_MINOR_UNCLASSIFIED</b>	Uncategorized.
<b>BT_MINOR_HS_PROFILE</b>	Device conforms to the Headset profile

#### *clockOffset*

Specifies the clock offset of the Bluetooth device

### BT\_SERVICE

```
typedef struct _BT_SERVICE /* BT Service structure */
{
    UINT8    protocolDescriptor;
    //Protocol descriptor specifying the protocol
    UINT8    profileDescriptor;
    //Profile descriptor specifying the profile over which the service is
    //registered
    UINT16    identifier;           //Protocol specific identifier[OUT]
    UINT8    serviceName[255];    //Service name, if a name was registered
} BT_SERVICE, *PBT_SERVICE;
```

The **BT\_SERVICE** structure defines service related information.

#### Members

##### *protocolDescriptor*

Specifies the underlying protocol. Can be one of the following.

Value	Meaning
<b>L2CAP</b>	Specifies L2CAP protocol
<b>RFCOMM</b>	Specifies RFCOMM protocol

##### *profileDescriptor*

Specifies the underlying protocol. Can be one of the following.

Value	Meaning
<b>SPP</b>	Specifies Serial Port Profile
<b>OPP</b>	Specifies Object Push Profile
<b>FTP</b>	Specifies File Transfer Profile

##### *identifier*

Protocol specific identifier. Will be PSM when protocol descriptor is L2CAP and server channel when protocol descriptor is RFCOMM.

##### *serviceName*

Specifies the name of the service, if a name was registered.

### BT\_CONNECTION

```
typedef struct _BT_CONNECTION /* Information on an active connection */
{
```

```

BT_DEVICE      btDevice;      //Device information
BT_SERVICE     btService;     //Service information
DEVICE_NAME    name;          //Device name
UINT8          initiator;     //Initiator of the connection
PROFILE_CONTEXT profileContext;
// profile specific additional information if there is any
} BT_CONNECTION, *PBT_CONNECTION;

```

## BT\_REGISTERSERVICE\_INFO

```

typedef struct _BT_REGISTERSERVICE_INFO /*Structure which returns Service and
Record Handles(used in Profiles for Service Registration)*/
{
    HANDLE      serviceHandle;    // handle to the Service record created.
    UINT32      recordHandle;
    // handle to the Service record added in the SDP Database.
} BT_REGISTERSERVICE_INFO, *PBT_REGISTERSERVICE_INFO;

```

## BT\_SECURITY\_ATTRIBUTE

```

typedef struct _BT_SECURITY_ATTRIBUTE
{
    UINT8 authenticate: 1;        // enable/disable authentication
    UINT8 authorize: 1;          // enable/disable authorization
    UINT8 encrypt: 1;            // enable/disable encryption
    UINT8 incomingConnection: 1; // incoming/outgoing connection
    UINT8 allowConnectionless: 1; // enable/disable connectionless security
    UINT8 broadcast: 1;          // broadcast allowed or not
    UINT8 reserved: 2;           // reserved
} BT_SECURITY_ATTRIBUTE, *PBT_SECURITY_ATTRIBUTE;

```

The **BT\_SECURITY\_ATTRIBUTE** structure defines the security parameters to be used with service level security.

### Members

#### *authenticate*

Specifies whether to enable/disable authentication. TRUE to enable, FALSE to disable.

#### *authorize*

Specifies whether to enable/disable authorization. TRUE to enable, FALSE to disable.

#### *encrypt*

Specifies whether to enable/disable encryption. TRUE to enable, FALSE to disable.

#### *incoming Connection*

Specifies whether security is to be forced on incoming or outgoing connection. TRUE for incoming, FALSE for outgoing.

#### *allowConnectionless*

Specifies whether to allow connection less security. TRUE to allow, FALSE to not allow.

#### *broadcast*

Specifies whether broadcast is allowed or not.

## BT\_UUID

```

typedef struct _BT_UUID /* SDP UUID */
{
    UINT8      uuid[16]; //UUID bytes
    UINT8      length;    //length of UUID
} BT_UUID, *PBT_UUID;

```



The **BT\_UUID** structure defines the Universally Unique Identifier (UUID) used in SDAP transactions.

### Members

*uuid*

Specifies the value of the UUID

*length*

The number of bytes used to specify the UUID, length of 2,4 or 16 are valid.

#### 4.1.1.1 BT\_GetErrorCode

The BT\_GetErrorCode function returns the most recent error-code.

---

```
UINT32 BT_GetErrorCode();
```

---

##### Parameters

None

##### Return Values

Returns 32 bit code for the most recent error that have occurred.

#### 4.1.1.2 GAP\_InquireAndRetrieveDevices

The GAP\_InquireAndRetrieveDevices function performs synchronous, non-periodic device discovery.

---

```
BOOL GAP_InquireAndRetrieveDevices(  
    UINT32 accessCode,  
    UINT8 duration,  
    BT_DEVICE devices[],  
    UINT8 size,  
    PUINT8 retCount  
);
```

---

##### Parameters

*accessCode*

[in] Specifies the type of inquiry to be performed. This parameter can be one of the following values

Value	Meaning
BT_GIAC	General inquiry. Retrieve all the devices
BT_LIAC	Limited inquiry. Retrieve only devices that are set to scan for inquiry messages with the Limited Inquiry Access Code
0x9E8B01 to 0x9E8B32	Custom inquiry access codes. One of the values in the range can be specified.

*duration*

[in] Specifies the maximum amount of time before the inquiry is halted. This value is specified in units of 1.28seconds and can have values in between 0x01 (1.28 sec) and 0x30 (61.44 sec).

*devices*

[out] Pointer to an array of BT\_DEVICE objects that receives the information of discovered devices. The array should be able to accommodate 'size' number of devices.

*size*

[in] Specifies the maximum number of devices to be retrieved before the inquiry is halted.

*retCount*

[out] Pointer to an UINT8 variable that receives value for the number of devices returned.

##### Return Values

Returns TRUE on success, FALSE otherwise. In case of error, call BT\_GetErrorCode to get extended error information.

##### Remarks

**GAP\_InquireAndRetrieveDevices** puts the local Bluetooth device in inquiry mode for the specified duration. When put in inquiry mode, the local Bluetooth device searches for devices in the vicinity that are in discoverable mode using the access code supplied. Devices that responded to the discovery procedure will be returned back to the caller through the output parameter.

Depending on the link manager implementation, some cards might report multiple entries of the same device during device discovery. aveLinkBT SDK does not explicitly filter the multiple entries.

**GAP\_InquireAndRetrieveDevices** will fail if the device is already in inquiry mode, periodic or non-periodic.

---

C++

```
BOOLEAN retVal;
BT_DEVICE devices[100];
UINT8 retCount = 0;

// Perform device search

retVal = GAP_InquireAndRetrieveDevices(BT_GIAC, 6, devices, 100, &retCount);

if(retVal == FALSE)
{
    // Call error handler
    return FALSE;
} // Device search completed successfully
```

---

#### 4.1.1.3 SDAP\_CloseConnection

The SDAP\_CloseConnection closes a previously established L2CAP channel.

---

```
BOOL SDAP_CloseConnection(
    HANDLE handle
);
```

---

##### **Parameters**

*handle*

[in] handle to a previously established L2CAP channel.

##### **Return Values**

Returns TRUE on success, FALSE on failure. In case of error, call BT\_GetErrorCode to get extended error information.

---

C++

```
BOOLEAN retVal;

//closing the channel
//The 'handle' in SDAP_CloseConnection is the handle obtained from
//SDAP_CreateConnection.

retVal = SDAP_CloseConnection (handle);

if(retVal == FALSE)
{
    // Call error handler
    return FALSE;
}
```

---

#### 4.1.1.4 SDAP\_GetConnection

The SDAP\_GetConnection function establishes an L2CAP channel with the remote device. All the future SDP transactions will go through this channel.

---

```
HANDLE SDAP_GetConnection (
    PBT_DEVICE btDevice
);
```

---

##### **Parameters**

*btDevice*

[in] Pointer to an instance of BT\_DEVICE that contains information required for identifying the remote device.

## Return Values

Returns unique handle to distinguish the connection on success, NULL on failure. In case of error, call BT\_GetErrorCode to get extended error information.

---

C++

```
BT_DEVICE devices;
HANDLE handle = NULL;

//Creating an L2CAP channel, the future SDP transactions
//will be through this channel
//The structure instance("devices") should contain
//the bdaddress of the device with which the
//user wants to connect

handle = SDAP_GetConnection(&devices);
if(!handle)
{
    // Call error handler
}
```

---

### 4.1.1.5 SPP\_Close

The SPP\_Close function closes an SPP connection.

---

```
BOOL SPP_Close(
    HANDLE handle
);
```

---

## Parameters

*handle*

[in] The unique handle corresponding to the SPP connection which is to be closed.

## Return Values

Returns TRUE on success, FALSE otherwise. In case of error, call BT\_GetErrorCode to get extended error information.

## Remarks

After issuing SPP\_Close, the port bound to this SPP Connection will no longer be usable because the underlying SPP connection is closed.

SPP\_Close is implicitly invoked by the Virtual Port Driver when a legacy application closes a virtual port bound to the underlying SPP connection.

### 4.1.1.6 SPP\_Connect

The SPP\_Connect function establishes a serial port connection to a peer device.

---

```
HANDLE SPP_Connect (
    PBT_DEVICE device,
    PBT_SERVICE service,
    PINT16 frameSize,
    PBT_SECURITY_ATTRIBUTE secAttrib,
    PINT8 portName
);
```

---

## Parameters

*device*

[in] Pointer to the BT\_DEVICE instance that contains specifications of the remote device.

*service*

[in] Pointer to an instance of BT\_SERVICE corresponding to the service on which the channel should be established.

#### *frameSize*

[in,out] Maximum number of bytes transferred per frame. If 0, a default size of 127 bytes will be used. On return, the pointer will receive the actual frame size that was negotiated for.

#### *secAttrib*

[in] Pointer to an instance of BT\_SECURITY\_ATTRIBUTE if the user would like to use on this connection when initiated as server or client. Can be null if no security is preferred.

#### *portName*

[in] Pointer to a null terminated string that specifies the name of the virtual COM port for which the SPP connection is made.

### **Return Values**

Returns a unique handle on success, NULL otherwise. In case of error call BT\_GetErrorCode to get extended error information.

### **Remarks**

If the frame size parameter is given as 0, a default value of 127 will be used for negotiation. All other SPP functions use the connection handle returned by this function. The user should explicitly close the connection after all the transactions. User can either obtain the BT\_Service parameters Using Service Discovery functions and fill the BT\_Service or by using SPP\_GetRemoteServices function.

---

C++

```
HANDLE sppHandle = NULL;
BT_DEVICE device = {0};
BT_SERVICE service = {0};
UINT16 frameSize = 1024;
INT8 portName[] = "COM4:";

sppHandle = SPP_Connect(
    &device,      // device info of the server
    &service,     // service info
    &frameSize,   // frame size to negotiate
    NULL,        // security attribute
    portName     // communications port to bind with
);
if(!sppHandle)
{
    printf("Creating Connection failed %d \n", BT_GetErrorCode());
}
```

---

## 4.1.2 Reference and Function List for C#

### Structure

#### BT\_SERVICE

```
struct BT_SERVICE
{
    byte    protocolDescriptor; // Protocol descriptor specifying the protocol
    byte    profileDescriptor;  // Profile descriptor specifying the profile
    ushort   identifier;        // Protocol specific identifier
    byte[]   serviceName;       // Service name,if a name was registered
}
```

The **BT\_SERVICE** structure defines service related information.

### Members

#### *protocolDescriptor*

Specifies the underlying protocol. Can be one of the following.

Value	Meaning
<b>L2CAP</b>	Specifies L2CAP protocol
<b>RFCOMM</b>	Specifies RFCOMM protocol

#### *profileDescriptor*

Specifies the underlying protocol. Can be one of the following.

Value	Meaning
<b>SPP</b>	Specifies Serial Port Profile
<b>OPP</b>	Specifies Object Push Profile
<b>FTP</b>	Specifies File Transfer Profile

#### *identifier*

Protocol specific identifier. Will be PSM when protocol descriptor is L2CAP and server channel when protocol descriptor is RFCOMM.

#### *serviceName*

Specifies the name of the service, if a name was registered.

#### BT\_DEVICE

```
struct BT_DEVICE
{
    byte[]    bdAddress;    // Bluetooth device address
    byte      pageScanRepetitionMode;
                // page scan repetition mode of the device
    byte      pageScanPeriodMode; // page scan period mode of the device
    byte      pageScanMode;       // page scan mode of the device
    byte      serviceClass;       // service class of the device
    byte      majorDeviceClass;   // major device class of the device
    byte      minorDeviceClass;   // minor device class of the device
    byte      clockOffset[2];     // clock offset of the device
}
```

The **BT\_DEVICE** structure defines the characteristics of a Bluetooth device..

### Members

#### *bdAddress*

Specifies address of the Bluetooth device.

#### *pageScanRepetitionMode*

Specifies page scan repetition mode of the Bluetooth device. This member can be one of the following values.

Value	Meaning
PSRM_R0	Page scan repetition mode R0
PSRM_R1	Page scan repetition mode R1
PSRM_R2	Page scan repetition mode R2

#### *pageScanPeriodMode*

Specifies page scan period mode of the Bluetooth device. This member can be one of the following values.

Value	Meaning
PSRM_P0	P0
PSRM_P1	P1
PSRM_P2	P2

#### *pageScanRepetitionMode*

Specifies page scan mode of the Bluetooth device. This member can be one of the following values.

Value	Meaning
PSM_MANDATORY	Mandatory page scan mode
PSM_OPTIONAL_I	Optional page scan mode I
PSM_OPTIONAL_II	Optional page scan mode II
PSM_OPTIONAL_III	Optional page scan mode III

#### *serviceClass*

Specifies service class of the Bluetooth device. This member can be one of the following values.

Value	Meaning
BT_SC_UNSPECIFIED	Unspecified service class
BT_SC_LIMITED_DISCOVERABLE_MODE	Limited discoverable mode service class
BT_SC_NETWORKING	Networking service class
BT_SC_RENDERING	Rendering service class
BT_SC_CAPTURING	Capturing service class
BT_SC_OBJECT_TRANSFER	Object Transfer service class
BT_SC_AUDIO	Audio service class
BT_SC_TELEPHONY	Telephony service class
BT_SC_INFORMATION	Information service class

#### *majorDeviceClass*

Specifies the minor device class of the Bluetooth device. This member can be one of the following values depending on the major device class.

Value	Meaning
BT_MAJOR_MISC	Miscellaneous
BT_MAJOR_COMPUTER	Computer (Desktop, Notebook, ...)
BT_MAJOR_PHONE	Phone (Cellular, Codeless, ...)
BT_MAJOR_LAP	LAN/Network access point.
BT_MAJOR_AUDIO	Audio/video
BT_MAJOR_PERIPHERAL	Peripheral (Mouse, ...)

#### *minorDeviceClass*

Specifies the minor device class of the Bluetooth device. This member can be one of the following values depending on the major device class.

1. Major device class: BT\_MAJOR\_COMPUTER

Value	Meaning
BT_MINOR_UNCLASSIFIED	Uncategorized.
BT_MINOR_DESKTOP	Desktop workstation.

<b>BT_MINOR_SERVER</b>	Server-class computer
<b>BT_MINOR_LAPTOP</b>	Laptop
<b>BT_MINOR_HANDHELD</b>	Handheld PC/PDA
<b>BT_MINOR_PALM</b>	Wearable computer

2. Major device class: BT\_MAJOR\_PHONE

Value	Meaning
<b>BT_MINOR_UNCLASSIFIED</b>	Uncategorized.
<b>BT_MINOR_CELLULAR</b>	Cellular
<b>BT_MINOR_CORDLESS</b>	Cordless
<b>BT_MINOR_SMARTPHONE</b>	Smart phone
<b>BT_MINOR_WIREDMODEM</b>	Wired modem

3. Major device class: BT\_MAJOR\_LAP

Value	Meaning
<b>BT_MINOR_UNCLASSIFIED</b>	Uncategorized.
<b>BT_MINOR_FULLY_AVAILABLE</b>	Service fully available
<b>BT_MINOR_17</b>	Service 17% utilized
<b>BT_MINOR_33</b>	Service 33% utilized
<b>BT_MINOR_50</b>	Service 50% utilized
<b>BT_MINOR_67</b>	Service 67% utilized
<b>BT_MINOR_99</b>	Service 99% utilized
<b>BT_MINOR_NO_SERVICE</b>	Service not service

4. Major device class: BT\_MAJOR\_AUDIO

Value	Meaning
<b>BT_MINOR_UNCLASSIFIED</b>	Uncategorized.
<b>BT_MINOR_HS_PROFILE</b>	Device conforms to the Headset profile

*clockOffset*

Specifies the clock offset of the Bluetooth device

## **BT\_UUID**

```
struct _BT_UUID
{
    byte    uuid[16]; // UUID value
    byte    length;   // number of bytes used to specify the UUID
}
```

The **BT\_UUID** structure defines the Universally Unique Identifier (UUID) used in SDAP transactions.

## **Members**

*uuid*

Specifies the value of the UUID

*length*

The number of bytes used to specify the UUID, length of 2,4 or 16 are valid.



#### 4.1.2.1 BTGetErrorCode

The BT\_GetErrorCode function returns the most recent error-code.

---

**UINT BTGetErrorCode();**

---

##### Parameters

None

##### Return Values

Returns the 32 bit code for the most recent error that have occurred.

#### 4.1.2.2 BTInit

The BTInit function initializes the Bluetooth host stack as well as the Bluetooth hardware module.

---

**BOOL BTInit();**

---

##### Parameters

None

##### Return Values

Returns TRUE on success, FALSE on failure. In case of error, call BTGetErrorCode to get extended error information.

##### Remarks

**BTInit** allows the application developer to start Bluetooth by initializing the Bluetooth host stack and hardware. Transport layer and its parameters, if any, should be set prior to invoking this method. Initialization will be done, by default, in non-discoverable, non-connectable mode. Application developer can alter these settings later, using the functions GAPSetDiscoverableMode and GAPSetConnectableMode.

This function will fail when the transport layer and/or its parameters, if any, set using transport layer functions are invalid with respect to the underlying Bluetooth hardware.

In a multiple application scenario, the applications other than the one which invoked BTInit can use the initialized stack by calling BTOpen .

---

##### C#

```
Using AveBluetooth;

bool retVal;
BTCore btCore = new BTCore();

// Start Bluetooth
retVal = btCore.BTInit();
if(retVal == false)
{
    // Call error handler
}
```

---

#### 4.1.2.3 BTDeInit

The BTDeInit function resets the Bluetooth host stack and the Bluetooth hardware module.

---

**BOOL BTDeInit();**

---

##### Parameters

None

##### Return Values

Returns TRUE on success, FALSE on failure. In case of error, call BTGetErrorCode to get extended error

information.

#### Remarks

**BTDeInit** allows the application developer to stop Bluetooth by resetting the Bluetooth hardware and the host stack. It also releases all the resources allocated. The other applications running over bluetooth, will be informed via reset indication. This function will fail if Bluetooth is not already started.

---

C#

```
Using AveBluetooth;

bool retVal;
BTCore btCore = new BTCore();

// Stop Bluetooth

retVal = btCore.BTDeInit();
if(retVal == false)
{
    // Call error handler
}
```

---

#### 4.1.2.4 BTOpen

The BTOpen function opens the already initialized Bluetooth stack.

---

**BOOL BTOpen();**

---

#### Parameters

None

#### Return Values

Returns TRUE on success, FALSE on failure. In case of error, call BTGetErrorCode to get extended error information.

#### Remarks

**BTOpen** function allows multiple applications to use the already initialized stack.

---

C#

```
Using AveBluetooth;

bool retVal;
BTCore btCore = new BTCore();

// Open stack

retVal = btCore.BTIsInitialized();

if(retVal == true)
{
    retVal = btCore.BTOpen();
    if(retVal == false)
    {
        // Call error handler
    }
}
```

---

#### 4.1.2.5 BTClose

The BTClose function closes the Bluetooth stack.

---

**BOOL BTClose();**

---

## Parameters

None

## Return Values

Returns TRUE on success, FALSE on failure. In case of error, call BTGetErrorCode to get extended error information.

## Remarks

**BTClose** function is used to close the Bluetooth stack which was acquired using BTOpen. In a multiple application environment, calling BTClose, by any application won't affect other applications accessing the stack. This function will fail if the stack is not opened.

---

C#

```
Using AveBluetooth;

bool retVal;
BTCore btCore = new BTCore();

// close stack

retVal = btCore.BTClose();
if(retVal == false)
{
    // Call error handler
}
```

---

### 4.1.2.6 GAPInquireAndRetrieveDevices

The GAP\_InquireAndRetrieveDevices function performs synchronous, non-periodic device discovery.

---

```
bool GAPInquireAndRetrieveDevices(
    uint          accessCode,
    byte          duration,
    ref BT_DEVICE[] devices,
    byte          size,
    out byte      retCount
);
```

---

## Parameters

*accessCode*

[in] Specifies the type of inquiry to be performed. This parameter can be one of the following values

Value	Meaning
<b>BT_GIAC</b>	General inquiry. Retrieve all the devices
<b>BT_LIAC</b>	Limited inquiry. Retrieve only devices that are set to scan for inquiry messages with the Limited Inquiry Access Code
<b>0x9E8B01 to 0x9E8B32</b>	Custom inquiry access codes. One of the values in the range can be specified.

*duration*

[in] Specifies the maximum amount of time before the inquiry is halted. This value is specified in units of 1.28seconds and can have values in between 0x01 (1.28 sec) and 0x30 (61.44 sec).

*devices*

[out] Pointer to an array of BT\_DEVICE objects that receives the information of discovered devices. The array should be able to accommodate 'size' number of devices.

*size*

[in] Specifies the maximum number of devices to be retrieved before the inquiry is halted.

*retCount*

[out] Pointer to an UINT8 variable that receives value for the number of devices returned.

### Return Values

Returns TRUE on success, FALSE otherwise. In case of error, call BTGetErrorCode to get extended error information.

### Remarks

**GAP\_InquireAndRetrieveDevices** puts the local Bluetooth device in inquiry mode for the specified duration. When put in inquiry mode, the local Bluetooth device searches for devices in the vicinity that are in discoverable mode using the access code supplied. Devices that responded to the discovery procedure will be returned back to the caller through the output parameter.

Depending on the link manager implementation, some cards might report multiple entries of the same device during device discovery. aveLinkBT SDK does not explicitly filter the multiple entries.

**GAP\_InquireAndRetrieveDevices** will fail if the device is already in inquiry mode, periodic or non-periodic.

---

C#

```
Using AveBluetooth;

bool retVal;
BTCore btCore = new BTCore();
BT_DEVICE devices[10];
byte retCount = 0;
// Perform device search

retVal = btCore.GAPInquireAndRetrieveDevices(BTCore.BT_GIAC, 6, ref devices, 10,
out retCount);

if(retVal == false)
{
    // Call error handler
}
// Device search completed successfully
```

---

#### 4.1.2.7 SPPGetRemoteServices

The SPPGetRemoteServices function gets a list of services registered over SPP on a specified remote device.

---

```
bool SPPGetRemoteServices (
    BT_DEVICE      device,
    BT_UUID        uuid,
    ref BT_SERVICE[] services,
    ushort         count,
    out ushort     numServices
);
```

---

### Parameters

*device*

[in] An instance of BT\_DEVICE that contains information on the remote device.

*uuid*

[in] An instance of BT\_UUID structure specifying a specific UUID in case the user requires information about services that has this particular UUID in their service record. Can be NULL.

*services*

[out] Referencer to an array of BT\_SERVICE objects that receives the service specific information.

*count*

[in] The maximum number of services that is to be retrieved.

*numServices*

[out] Reference to an ushort variable that receives the actual number of services retrieved.

### Return Values

Returns TRUE on success, FALSE otherwise. In case of error, call BTGetErrorCode to get extended error information.

### Remarks

This function retrieves all the services registered over SPP from the remote device, which has the given uuid in their service record. If no UUID is specified i.e., if it is NULL, all the SPP services in the remote device will be retrieved.

---

C#

```
BT_DEVICE device; // device info received from the device search
BT_SERVICE[] services = new BT_SERVICE [10];
ushort count = 10;
ushort numServices = 0;

BT_UUID uuid = {0x11, 0x11}, 2} ;
if(!btCore.SPPGetRemoteServices(device, uuid, ref services, count, out
numServices))
{
    // SPP_GetRemoteServices failed.
}
```

---

#### 4.1.2.8 SPPConnect

The SPPConnect function establishes a serial port connection to a peer device.

---

```
IntPtr SPPConnect (
    BT_DEVICE          device,
    BT_SERVICE         service,
    ref ushort         frameSize,
    BT_SECURITY_ATTRIBUTE secAttrib,
    string             portName
);
```

---

### Parameters

*device*

[in] BT\_DEVICE instance that contains specifications of the remote device.

*service*

[in] BT\_SERVICE corresponding to the service on which the channel should be established.

*frameSize*

[in, out] Maximum number of bytes transferred per frame. If 0, a default size of 127 bytes will be used. On return, the pointer will receive the actual frame size that was negotiated for.

*secAttrib*

[in] instance of BT\_SECURITY\_ATTRIBUTE if the user would like to use on this connection when initiated as server or client. Can be null if no security is preferred.

*portName*

[in] A null terminated string that specifies the name of the virtual COM port for which the SPP connection is made.

### Return Values

Returns TRUE on success, FALSE otherwise. In case of error, call BTGetErrorCode to get extended error information.

### Remarks

If the frame size parameter is given as 0, a default value of 127 will be used for negotiation. All other SPP functions use the connection handle returned by this function. The user should explicitly close the connection

after all the transactions.

---

C#

```
IntPtr sppHandle = NULL;
BT_DEVICE device = {0};
BT_SERVICE service = {0};
ushort frameSize = 1024;
string portName = "COM4:";

sppHandle = btCore.SPPConnect
(
    device,          // device info of the server
    service,         // service info
    ref frameSize,   // frame size to negotiate
    NULL,            // security attribute
    portName         // communications port to bind with
);
if(!sppHandle)
{
    //Creating Connection failed %d \n", BT_GetErrorCode());
}
```

---

#### 4.1.2.9 SPPClose

The SPPClose function closes an SPP connection.

---

```
BOOL SPPClose(
    IntPtr handle
);
```

---

##### Parameters

*handle*

[in] The unique handle corresponding to the SPP connection which is to be closed.

##### Return Values

Returns TRUE on success, FALSE on failure. In case of error, call BTGetErrorCode to get extended error information.

##### Remarks

After issuing **SPPClose**, the port bound to this SPP Connection will no longer be usable because the underlying SPP connection is closed.

**SPPClose** is implicitly invoked by the Virtual Port Driver when a legacy application closes a virtual port bound to the underlying SPP connection

.

#### 4.1.3 Error Codes for Bluetooth

Error Code	Description
BT_NO_ERROR	No error has occurred. Value : 0x0000000
BT_INVALID_PARAMETER	A parameter that was passed to the API was invalid. Value : 0x30F0001
BT_INSUFFICIENT_RESOURCES	Insufficient system resources to complete the API. Value : 0x30F0002
BT_INVALID_HANDLE	The handle that was passed to the API was invalid. Value : 0x30F0003
BT_NOT_INITIALIZED	The bluetooth stack and hardware module has not been initialised. Value : 0x30F0004
BT_REQUEST_CANCELED	The request has been canceled. Value : 0x30F0005
BT_USER_DE_INITIALIZED	The user initiated deinitialisation. Value : 0x10F0006
BT_NOT_OPENED	The bluetooth stack was not Opened for use. Value : 0x30F0007
BT_TRIAL_PERIOD_EXPIRED	Trial period expired. Value : 0x30F0008
BT_ALREADY_INITIALIZED	Stack was already initialized. Value : 0x20F0009
BT_VALID_CERTIFICATE_FILE_NOT_FOUND	Either no certificate file or invalid certificate file. Value : 0x20F0009
BT_TRANSPORT_ALREADY_INITIALIZED	The transport layer already initialized. Value : 0x2010001
BT_TRANSPORT_NOT_INITIALIZED	Transport layer not initialized. Value : 0x3010007
BT_TRANSPORT_H2_DEVICE_NOT_READY	Attempt to retrieve device information failed. Value : 0x3110001
BT_TRANSPORT_H2_COMPONENT_MISSING	Component required for the usb transport missing. Value : 0x3110002
BT_TRANSPORT_H2_READ_FAILURE	Attempt to read from the device failed. Value : 0x4110003
BT_TRANSPORT_H2_INVALID_PACKET_ERROR	Invalid bluetooth packet. Value : 0x4110004
BT_TRANSPORT_H2_WRITE_FAILED	Attempt to write to the device failed. Value : 0x4110005
BT_TRANSPORT_H4_BAD_PORT	Port with the specified name does not exist. Value : 0x3100001
BT_TRANSPORT_H4_PORT_IN_USE	Specified port is currently in use. Value : 0x3100002
BT_TRANSPORT_H4_UNSUPPORTED_PARAMETERS	Specified parameters not supported by the transport layer. Value : 0x3100003
BT_TRANSPORT_H4_PARAMETERS_ERROR	Invalid communication parameter (Port parameters). Value : 0x3100005
BT_TRANSPORT_H4_READ_FAILURE	Attempt to read from the device failed. Value : 0x4100006
BT_TRANSPORT_H4_INVALID_PACKET_ERROR	Invalid bluetooth packet. Value : 0x4100007
BT_TRANSPORT_H4_WRITE_FAILED	Attempt to write to the device failed. Value : 0x4100008
BT_TRANSPORT_BCSP_BAD_PORT	Port with the specified name does not exist. Value : 0x3130001
BT_TRANSPORT_BCSP_PORT_IN_USE	Specified port is currently in use.

	Value : 0x3130002
<b>BT_TRANSPORT_BCSP_UNSUPPORTED_PARAMETERS</b>	Specified parameters not supported by the transport layer. Value : 0x3130003
<b>BT_TRANSPORT_BCSP_LINK_SETUP_FAILED</b>	BCSP link set up between host and host controller failed. Value : 0x3130005
<b>BT_TRANSPORT_BCSP_READ_FAILURE</b>	Attempt to read from the device failed. Value : 0x4130006
<b>BT_TRANSPORT_BCSP_INVALID_PACKET_ERROR</b>	Invalid bluetooth packet. Value : 0x4130007
<b>BT_TRANSPORT_BCSP_WRITE_FAILED</b>	Attempt to write to the device failed. Value : 0x4130008
<b>BT_HCI_UNKNOWN_COMMAND</b>	Unknown hci command Value : 0x3020001
<b>BT_HCI_NO_EXISTING_CONNECTION</b>	Host has issued a command which requires an existing connection and there is currently no connection corresponding to the specified device Address. Value : 0x3020002
<b>BT_HCI_HARDWARE_FAILURE</b>	Command cannot be executed because of a hardware failure. Value : 0x3020003
<b>BT_HCI_PAGE_TIMEOUT</b>	Page timeout occurred during connection attempt. Value : 0x3020004
<b>BT_HCI_AUTHENTICATION_FAILURE</b>	Pairing or authentication failed due to incorrect PIN code or link key. Value : 0x3020005
<b>BT_HCI_KEY_MISSING</b>	PIN code(s) missing. Value : 0x3020006
<b>BT_HCI_MEMORY_FULL</b>	Host Controller does not have memory capacity to store additional parameters. Value : 0x3020007
<b>BT_HCI_CONNECTION_TIMEOUT</b>	Connection timeout. Value : 0x3020008
<b>BT_HCI_MAX_NO_OF_CONNECTIONS</b>	Maximum number of connections established. Value : 0x3020009
<b>BT_HCI_MAX_NO_OF_SCO_CONNECTIONS</b>	Maximum number of SCO connections established. Value : 0x302000A
<b>BT_HCI_ACL_CONNECTION_ALREADY_EXISTS</b>	An ACL connection to the device already exists. Value : 0x302000B
<b>BT_HCI_COMMAND_DISALLOWED</b>	Command disallowed. Value : 0x302000C
<b>BT_HCI_HOST_REJECTED_LIMITED_RESOURCES</b>	Host rejected the incoming connection request due to limited resources. Value : 0x302000D
<b>BT_HCI_HOST_REJECTED_SECURITY_REASONS</b>	Host rejected the incoming connection request due to security reasons. Value : 0x302000E
<b>BT_HCI_HOST_REJECTED_PERSONAL_DEVICE</b>	Host Rejected the incoming connection request due to remote device is only a personal device. Value : 0x302000F
<b>BT_HCI_HOST_TIMEOUT</b>	Host timeout occurred before responding to an incoming connection request. Value : 0x3020010
<b>BT_HCI_UNSUPPORTED_FEATURE_OR_PARAMETER</b>	One or more specified parameter(s) not supported by the hardware. Value : 0x3020011
<b>BT_HCI_INVALID_COMMAND_PARAMETER</b>	One or more specified parameter values not conform to the bluetooth specification. Value : 0x3020012



<b>BT_HCI_OTHER_END_TERMINATED_USER_ENDED_CONNECTION</b>	Connection terminated by user at the other end. Value : 0x3020013
<b>BT_HCI_OTHER_END_TERMINATED_LOW_RESOURCES</b>	Connection terminated by the other end due to low resources. Value : 0x3020014
<b>BT_HCI_OTHER_END_TERMINATED_POWER_OFF</b>	Other End Terminated Connection: About to Power Off. Value : 0x3020015
<b>BT_HCI_CONNECTION_TERMINATED_LOCAL_HOST</b>	Connection terminated by the local host. Value : 0x3020016
<b>BT_HCI_REPEATED_ATTEMPTS</b>	Device not allowed authentication\pairing because too little time has elapsed since an unsuccessful authentication\pairing attempt. Value : 0x3020017
<b>BT_HCI_PAIRING_NOT_ALLOWED</b>	Device not allowed pairing due to some reason. Value : 0x3020018
<b>BT_HCI_UNKNOWN_LMP_PDU</b>	Unknown LMP PDU. Value : 0x3020019
<b>BT_HCI_UNSUPPORTED_REMOTE_FEATURE</b>	Remote device does not support the feature associated with the issued command. Value : 0x302001A
<b>BT_HCI_SCO_OFFSET_REJECTED</b>	SCO offset rejected. Value : 0x302001B
<b>BT_HCI_SCO_INTERVAL_REJECTED</b>	SCO interval rejected. Value : 0x302001C
<b>BT_HCI_SCO_AIR_MODE_REJECTED</b>	SCO air mode rejected. Value : 0x302001D
<b>BT_HCI_INVALID_LMP_PARAMETERS</b>	Invalid LMP parameters. Value : 0x302001E
<b>BT_HCI_UNSPECIFIED_ERROR</b>	This error code is used when no other error code is appropriate. Value : 0x302001F
<b>BT_HCI_UNSUPPORTED_LMP_PARAMETER</b>	Unsupported LMP parameters. Value : 0x3020020
<b>BT_HCI_ROLE_CHANGE_NOT_ALLOWED</b>	Role change not allowed. Value : 0x3020021
<b>BT_HCI_LMP_RESPONSE_TIMEOUT</b>	LMP response timeout. Value : 0x3020022
<b>BT_HCI_LMP_ERROR_TRANSACTION_COLLISION</b>	LMP error transaction collision Value : 0x3020023
<b>BT_HCI_LMP_PDU_NOT_ALLOWED</b>	LMP PDU not allowed. Value : 0x3020024
<b>BT_HCI_ENCRYPTION_MODE_NOT_ACCEPTABLE</b>	Couldnt negotiate for which encryption mode to use. Value : 0x3020025
<b>BT_HCI_UNIT_KEY_USED</b>	Link key for the specified connection cannot be changed since it is a UNIT key. Value : 0x3020026
<b>BT_HCI_QOS_NOT_SUPPORTED</b>	Requested quality of service is not supported. Value : 0x3020027
<b>BT_HCI_INSTANT_PASSED</b>	The instant at which the specified operation shall be done is in the past. Value : 0x3020028
<b>BT_HCI_PAIRING_WITH_UNIT_KEY_NOT_SUPPORTED</b>	Pairing with unit key not supported. Value : 0x3020029
<b>BT_HCI_NOT_INITIALIZED</b>	HCI not initialized. Value : 0x3020054
<b>BT_HCI_INQUIRY_IN_PROGRESS</b>	Device inquiry is in progress. Value : 0x3020055
<b>BT_HCI_PERIODIC_INQUIRY_IN_PROGRESS</b>	Periodic device inquiry is in progress. Value : 0x3020056
<b>BT_HCI_WRITE_FAILED</b>	Attempt to write to the device failed. Value : 0x4020058

<b>BT_HCI_COMMAND_NOT_ALLOWED</b>	The specified command is not allowed. Value : 0x3020059
<b>BT_HCI_TIMEOUT</b>	Timeout occurred while waiting for the hci command response. Value : 0x302005A
<b>BT_HCI_NO_ACL_BUFFER</b>	No ACL buffer free in host controller. Value : 0x302005B
<b>BT_HCI_NO_SCO_BUFFER</b>	No SCO buffer free in host controller. Value : 0x302005C
<b>BT_HCI_REMOVE_PENDING</b>	HCI connection object being removed. Value : 0x302005D
<b>BT_SCO_NOT_INITIALIZED</b>	SCO layer not initialized. Value : 0x30E0001
<b>BT_SCO_REMOVE_PENDING</b>	SCO list being removed. Value : 0x30E0002
<b>BT_SCO_NO_EXISTING_ACL_CONNECTION</b>	No ACL connection exists with the remote device with which SCO connection is to be made. Value : 0x30E0003
<b>BT_SCO_NO_EXISTING_SCO_CONNECTION</b>	No SCO connection exists with the remote device. Value : 0x30E0004
<b>BT_SCO_MAX_NO_OF_SCO_CONNECTIONS</b>	Maximum number of SCO connections(three) already made. Value : 0x30E0005
<b>BT_SCO_TIMEOUT</b>	Request timed out. Value : 0x30E0006
<b>BT_L2CAP_TIMEOUT_CONNECT_REQ</b>	The wait for connect response timed out. Value : 0x3040001
<b>BT_L2CAP_TIMEOUT_CONFIG_REQ</b>	The wait for config response timed out. Value : 0x3040002
<b>BT_L2CAP_TIMEOUT_DISC_REQ</b>	The wait for disconnect response timed out. Value : 0x3040003
<b>BT_L2CAP_TIMEOUT_ECHO_REQ</b>	The wait for echo response timed out. Value : 0x3040004
<b>BT_L2CAP_TIMEOUT_INFO_REQ</b>	The wait for info response timed out. Value : 0x3040005
<b>BT_L2CAP_TIMEOUT_CONNECT_CFM_FROM_HCI</b>	The wait for ACL connect confirm from HCI timed out. Value : 0x3040006
<b>BT_L2CAP_TIMEOUT_DISCONNECT_RSP_FROM_HCI</b>	The wait for ACL disconnect response from HCI timed out. Value : 0x3040007
<b>BT_L2CAP_INVALID_STATE</b>	The current L2CAP state is invalid. Value : 0x3040008
<b>BT_L2CAP_ACL_CLOSE_PENDING</b>	HCI connection being closed. Value : 0x3040009
<b>BT_L2CAP_ACL_CONNECTION_PENDING</b>	Currently processing another hci_connectReq to the same device. Value : 0x304000A
<b>BT_L2CAP_INVALID_GROUP</b>	No such L2CAP group present. Value : 0x304000D
<b>BT_L2CAP_INVALID_GROUP_MEMBER</b>	Device not a member of the group. Value : 0x304000E
<b>BT_L2CAP_REMOVE_PENDING</b>	L2CAP connection object being removed. Value : 0x3040010
<b>BT_L2CAP_CONNECTION_ALREADY_EXIST</b>	L2CAP connection already exists. Value : 0x3040011
<b>BT_L2CAP_UNKNOWN_OPTIONS</b>	Channel configuration failed. Value : 0x3040014
<b>BT_L2CAP_CONNECTION_REFUSED</b>	L2CAP connection refused by remote machine. Value : 0x3040016
<b>BT_SDP_REQUEST_TIMEOUT</b>	Service search/service attribute/service search attribute requests timed out.

	Value : 0x3060001
<b>BT_SDP_UNSUPPORTED_SDP_VERSION</b>	SDP version not supported. Value : 0x3060004
<b>BT_SDP_REMOVE_PENDING</b>	SDP connection object being removed. Value : 0x3060005
<b>BT_SDP_INSUFFICIENT_RESOURCE</b>	Insufficient Resources to satisfy Request. Value : 0x3060006
<b>BT_SDP_ATTRIBUTE_ALREADY_EXISTS</b>	Occurs when user tries to add an existing attribute. Value : 0x3060007
<b>BT_SDP_L2CAP_CONNECTION_FAILED</b>	Failed to establish L2CAP connection. Value : 0x3060008
<b>BT_SDP_L2CAP_IN_CLOSING_STATE</b>	Occurs when L2CAP is in closing state. Value : 0x3060009
<b>BT_SDP_SERVICE_ALREADY_EXISTS</b>	Service already exists. Value : 0x306000E
<b>BT_SDP_RESULTSET_OVERFLOW</b>	Occurs when we try to access the element after cursor position (cursor count). Value : 0x3060014
<b>BT_SDP_RESULTSET_UNDERFLOW</b>	Occurs when we try to access the element before the starting position of the list. Value : 0x3060015
<b>BT_SDP_INVALID_CURSOR_POSITION</b>	Occurs when the cursor position is not valid (ie before the start of the list). Value : 0x3060016
<b>BT_RFCOMM_TIMEOUT_TEST_REQ</b>	Test request timed out. Value : 0x3080001
<b>BT_RFCOMM_TIMEOUT_SABM_REQ</b>	SABM request timed out. Value : 0x3080002
<b>BT_RFCOMM_TIMEOUT_PN_REQ</b>	PN request timed out. Value : 0x3080003
<b>BT_RFCOMM_TIMEOUT_MSC_REQ</b>	MSC request timed out. Value : 0x3080004
<b>BT_RFCOMM_TIMEOUT_FCON_REQ</b>	FCON request timed out. Value : 0x3080005
<b>BT_RFCOMM_TIMEOUT_FCOFF_REQ</b>	FCOFF request timed out. Value : 0x3080006
<b>BT_RFCOMM_TIMEOUT_RPN_REQ</b>	RPN request timed out. Value : 0x3080007
<b>BT_RFCOMM_TIMEOUT_RLS_REQ</b>	RLS request timed out. Value : 0x3080008
<b>BT_RFCOMM_BAUDRATE_NOT_SUPPORTED</b>	Baud rate not supported. Value : 0x3080009
<b>BT_RFCOMM_DATABITS_NOT_SUPPORTED</b>	Data bit not supported. Value : 0x308000A
<b>BT_RFCOMM_STOPBIT_NOT_SUPPORTED</b>	Stop bit not supported. Value : 0x308000B
<b>BT_RFCOMM_PARITYBIT_NOT_SUPPORTED</b>	Parity bit not supported. Value : 0x308000C
<b>BT_RFCOMM_PARITYTYPE_NOT_SUPPORTED</b>	Parity type not supported. Value : 0x308000D
<b>BT_RFCOMM_FLOWCONTROL_NOT_SUPPORTED</b>	Flowcontrol not supported. Value : 0x308000E
<b>BT_RFCOMM_XONCHAR_NOT_SUPPORTED</b>	xon character not supported. Value : 0x308000F
<b>BT_RFCOMM_XOFFCHAR_NOT_SUPPORTED</b>	xoff character not supported. Value : 0x3080010
<b>BT_RFCOMM_NO_UPPERLAYER_REGISTERED</b>	No upper layer registered. Value : 0x3080011
<b>BT_RFCOMM_SESSION_NOT_CONNECTED</b>	Session not in connected state. Value : 0x3080012
<b>BT_RFCOMM_CONNECTION_ALREADY_EXISTS</b>	The connection already exists to this channel. Value : 0x3080013

<b>BT_RFCOMM_DLCI_NOT_CONNECTED</b>	DLCI not in connected state. Value : 0x3080014
<b>BT_RFCOMM_TEST_DATA_MISMATCH</b>	Received test data mismatch. Value : 0x3080015
<b>BT_RFCOMM_NO_SESSION</b>	Session does not exist. Value : 0x3080016
<b>BT_RFCOMM_FCS_MISMATCH</b>	FCS mismatch occurred. Value : 0x3080017
<b>BT_RFCOMM_MAX_NO_OF_SESSIONS</b>	Already reached maximum number of allowable sessions. Value : 0x3080018
<b>BT_RFCOMM_REMOVE_PENDING</b>	An object referenced is going to be removed. Value : 0x3080019
<b>BT_RFCOMM_SERVER_ALREADY_WAITING</b>	A server is already waiting in the given server channel. Value : 0x3080020
<b>BT_SPP_PORT_ACCESS_DENIED</b>	The specified port is already bounded with an SPP connection or in the process of binding. Value : 0x3090001
<b>BT_SPP_PORT_NOT_FOUND</b>	The specified port may not be installed in the system or its installation may not be proper. Value : 0x3090002
<b>BT_SPP_SERVICE_IN_USE</b>	The specified service cannot be deleted as it is in use. Value : 0x3090003
<b>BT_SECURITY_ACCESS_DENIED</b>	Request rejected due to security failure. Value : 0x30C0001
<b>BT_SECURITY_THREAD_CREATION_FAILED</b>	Creating thread failed. Value : 0x30C0002
<b>BT_SECURITY_REGISTRY_OPERATION_FAILED</b>	Registry operation failed. Value : 0x30C0003
<b>BT_SECURITY_CALLBACK_ALREADY_REGISTERED</b>	Tried to set callback multiple times. Value : 0x30C0004
<b>BT_GOEP_CONNECTION_TERMINATED</b>	The transport connection has been terminated. Value : 0x3200001
<b>BT_GOEP_REQUEST_TIMEDOUT</b>	The request has been timed out. Value : 0x3200002
<b>BT_GOEP_OPERATION_IN_PROGRESS</b>	An operation is already in progress. Value : 0x3200003
<b>BT_GOEP_AUTHENTICATION_FAILED</b>	Server could not authenticate the client. Value : 0x3200004
<b>BT_GOEP_INVALID_RESPONSE</b>	Response from the server is invalid. Value : 0x3200005
<b>BT_GOEP_UNAUTHORIZED_SERVER</b>	Client could not authenticate the server. Value : 0x3200006
<b>BT_GOEP_OPERATION_ABORTED</b>	The operation is aborted. Value : 0x3200007
<b>BT_GOEP_INVALID_HEADER_ID</b>	The header is invalid. Value : 0x3200008
<b>BT_GOEP_INVALID_HANDLE</b>	Given handle is invalid. Handle can be sessionhandle, operation handle or headersethandle. Value : 0x3200009
<b>BT_GOEP_SERVICE_NOT_AVAILABLE</b>	The specified service is not available at the server. Value : 0x3200010
<b>BT_GOEP_ATTRIBUTE_NOT_FOUND</b>	The specified attribute is not available in the SDDb of the server. Value : 0x3200011
<b>BT_GOEP_WORKING_DIRECTORY_REQUIRED</b>	The working directory should be set prior to the function call. Value : 0x3200012
<b>BT_GOEP_FILE_NOT_FOUND</b>	The specified file not found in the given directory.

	Value : 0x3200013
<b>BT_GOEP_INVALID_PARAMETER</b>	A parameter that was passed to the API was invalid. Value : 0x3200014
<b>BT_GOEP_OBEXCONNECTION_FAILED</b>	GOEP connection failed to establish. Value : 0x3200015
<b>BT_GOEP_INVALID_TYPE</b>	Object is of an invalid type. Value : 0x3200016
<b>BT_GOEP_OPERATION_DISALLOWED</b>	Operation is disallowed. Value : 0x3200017
<b>BT_GOEP_NO_OPERATION_IN_PROGRESS</b>	Push or pull operation is not in progress. Value : 0x3200018
<b>BT_OPP_BUSINESSCARD_PUSH_FAILED</b>	Pushing business card failed. Value : 0x3210001
<b>BT_OPP_BUSINESSCARD_PULL_FAILED</b>	Pulling business card failed. Value : 0x3210002
<b>BT_DUN_ALREADY_CONNECTED</b>	The DUN connection is already established. Value : 0x3260001
<b>BT_LAP_ALREADY_CONNECTED</b>	The LAP connection is already established. Value : 0x3270001
<b>BT_HP_NOT_INITIALIZED</b>	HP has not been initialized Value : 0x3280001
<b>BT_HP_MAX_NO_OF_SESSIONS</b>	Number of possible sessions Exceeds Maximum Limit Value : 0x3280002
<b>BT_HIDP_ALREADY_INITIALIZED</b>	The HIDP component is already initialized Value : 0x3310001
<b>BT_HIDP_NOT_INITIALIZED</b>	The HIDP component is not initialized at all Value : 0x3310002
<b>BT_HIDP_REQUEST_PENDING</b>	An HIDp request is already pending Value : 0x3310003
<b>BT_HIDP_COMMAND_TIMEOUT</b>	HIDP command timed out Value : 0x3310004
<b>BT_HIDP_DEVICE_NOT_READY</b>	HID device is not ready for responding Value : 0x3310005
<b>BT_HIDP_INVALID_REPORT_ID</b>	An invalid report Id for the HID device Value : 0x3310006
<b>BT_HIDP_UNSUPPORTED_REQUEST</b>	The request is not supported by HID Value : 0x3310007
<b>BT_HIDP_INVALID_PARAMETER</b>	An invalid HIDP parameter Value : 0x3310008
<b>BT_HIDP_ERR_UNKNOWN</b>	An HIDP unknown error occurred Value : 0x3310009
<b>BT_HIDP_ERR_FATAL</b>	A fatal error in HIDP component Value : 331000A
<b>BT_HIDP_CONNECT_CANCELLED</b>	HIDP connection process cancelled Value : 331000B
<b>BT_AVDTP_L2CAP_REGISTRATION_FAILED</b>	Failed to register the L2cap Callbacks Value : 0x3320001
<b>BT_AVDTP_STREAM_CONTEXT_EMPTY</b>	The stream Context is Empty Value : 0x3320002
<b>BT_AVDTP_INVALID_SEID</b>	The given SEID is invalid Value : 0x3320003
<b>BT_AVDTP_INVALID_STREAM_STATE</b>	The present State of the Stream is Invalid Value : 0x3320004
<b>BT_AVDTP_NO_CHANNEL</b>	Code Text :No L2cap channel exists Value : 0x3320005
<b>BT_AVDTP_REQUEST_TIMEOUT</b>	Code Text :AVDTP request has timed out Value : 0x3320006
<b>BT_GAVDP_LOCAL_SEID_MISMATCH</b>	A mismatch has occurred between the Seids maintained locally. Value : 0x3330001
<b>BT_GAVDP_MEDIA_CODEC_NOT_SUPPORTED</b>	Media Codec is not supported

	Value : 0x3330002
<b>BT_GAVDP_CAPABILITY_REGISTRATION_VIOLATION</b>	An attempt to register more than one capability of a particular category Value : 0x3330003
<b>BT_GAVDP_ACCEPT_TIMEOUT</b>	GAVDP_Accept request timed out. Value : 0x3330004
<b>BT_GAVDP_SIGNALING_CHANNEL_ALREADY_EXISTS</b>	Presently supporting only connection between 2 devices Value : 0x3330005
<b>BT_GAVDP_REQUEST_TIMEOUT</b>	Code Text :GAVDP request has timed out Value : 0x3330006
<b>BT_A2DP_NOT_INITIALIZED</b>	A2D profile not initialized properly Value : 0x3340001
<b>BT_A2DP_SEPTYPE_NOT_SET</b>	A2DP -Local SEP Type not set Value : 0x3340002
<b>BT_A2DP_INVALID_SEPTYPE</b>	A2DP -Local SEP Type is invalid Value : 0x3340003
<b>BT_A2DP_REQUEST_TIMEOUT</b>	A2DP request has timed out Value : 0x33400

## 4.2 Camera

This section provides description of the functions and DLLs which are used to manage the camera module.

### Required Products

#### For C++

Required header:

M3P\_Camera.h

Required lib:

M3P\_Camera.lib

Required DLL:

M3P\_Camera.dll

#### For C#

Required DLL:

M3P\_Camera.dll  
M3p\_cam\_net.dll

### Supported Product

M3 GREEN with camera option

## 4.2.1 Reference and Function List for C++

### Definitions

#### 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

### Structure

#### CAM\_OPTION

```
typedef struct
{
    INT    nInFormat;        // IN Stream Format
    INT    nResolution;      // Resolution
    INT    nPrevRGBDepth;    // IN Stream RGB Depth
    INT    nSaveFormat;      // SaveFormat
    INT    nJpegQuality;     // Jpeg Quality
    INT    nImgWidth;        // Image Width
    INT    nImgHeight;       // Image Height
    INT    nNamePrefix;      // Image File Name Prefix

    INT    nImgLux;          // Image Lux
    INT    nImgExposure;     // Image exposure
    INT    nImgEffect;       // Image Effect
    INT    nImgbalance;      // Image Balance

    INT    nImgRotatesave;   //Image Rotate
    INT    nImgRotateview;  //Image Rotate
} CAM_OPTION, *LPCAM_OPTION;
```

### Parameters

#### Image Lux

```
#define OPTION_IMAGE_NOMAL_LUX    0
#define OPTION_IMAGE_NIGHT_LUX    1
```

#### Image Exposure

```
#define OPTION_IMAGE_EXPOSURE_ZERO    0
#define OPTION_IMAGE_EXPOSURE_ONE     1
#define OPTION_IMAGE_EXPOSURE_TWO     2
#define OPTION_IMAGE_EXPOSURE_THREE   3
#define OPTION_IMAGE_EXPOSURE_FOUR    4
#define OPTION_IMAGE_EXPOSURE_FIVE    5
```

#### 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_UVBLUE_EFFECT   5
#define OPTION_IMAGE_UVGREEN_EFFECT  6
```



Image Balance		
#define	OPTION_IMAGE_BALANCE_AUTO	0
#define	OPTION_IMAGE_BALANCE_SUNNY	1
#define	OPTION_IMAGE_BALANCE_DAY	2
#define	OPTION_IMAGE_BALANCE_CWF	3
#define	OPTION_IMGAE_BALANCE_INCANDESENCE	4

Image Resolution		
#define	OPTION_RESOLUTION_1600X1200	4
#define	OPTION_RESOLUTION_1280X1024	0
#define	OPTION_RESOLUTION_640X480	1
#define	OPTION_RESOLUTION_320X240	2

Preview RGB Depth		
#define	OPTION_PREV_RGB_DEPTH_16BIT	0
#define	OPTION_PREV_RGB_DEPTH_24BIT	1

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

Image Rotate		
#define	IMAGE_ROTATE_0	0
#define	IMAGE_ROTATE_90	1
#define	IMAGE_ROTATE_180	2
#define	IMAGE_ROTATE_270	3

#### 4.2.1.1 MC3P\_Init

This function performs initialization.

---

```
BOOL MC3P_Init (IntPtr hWnd, IntPtr p_hWnd);
```

---

##### Parameters

*hWnd*

Main window handle.

*p\_hWnd*

Windows handle to show image.

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

MC3P\_Init is a basic initialization step to run Camera. Please register Window Handle focused on running camera at the first parameter. At second parameter, register Window Handle that will output image. If initialization is succeeded, preview screen will be seen on registered control.

---

C++

```
CStatic m_ctrlpreview;
CM3P_Camera m_m3p_camera; // CM3P_Camera is the camera dll class.

void CameraInit()
{
    // m_ctrlpreview is a static tool on camera dlg
    m_m3p_camera.MC3P_Init(m_hWnd,m_ctrlpreview.m_hWnd)
    //return device type  0: 6300, 1: 6400, 2:6500

    if(!m_m3p_camera.MC3P_Open())
    {
        AfxMessageBox(L"Com Open Error");
        return FALSE;
    }
}
```

#### 4.2.1.2 MC3P\_Open

This function opens a COM port to a camera module.

---

```
BOOL MC3P_Open ();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

MC3P\_Opne function opens a camera port and initializes module at the same time.

---

C++

```
CStatic m_ctrlpreview;
CM3P_Camera m_m3p_camera; // CM3P_Camera is the camera dll class.

void CameraInit()
{
    // m_ctrlpreview is a static tool on camera dlg
```

```

m_m3p_camera.MC3P_Init(m_hWnd,m_ctrlpreview.m_hWnd)
//return device type  0: 6300, 1: 6400, 2:6500

if(!m_m3p_camera.MC3P_Open())
{
    AfxMessageBox(L"Com Open Error");
    return FALSE;
}
}

```

#### 4.2.1.3 MC3P\_Close

This function closes a COM port to a camera module.

---

```

BOOL MC3P_Close ( );

```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

MC3P\_Close function is called when the user which to close the camera.

---

C++

```

void CloseCam()
{
    MC3P_Close();
}

```

---

#### 4.2.1.4 MC3P\_PreviewStart

This function starts preview.

---

```

BOOL MC3P_PreviewStart( );

```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

MC3P\_PreviewStart function output the image from camera module through Window Handle registered in MC3P\_Init function.

---

C++

```

void PreviewStart(BOOL bOn)
{
    if(bOn == TRUE)
    {
        m_m3p_camera.MC3P_PreViewStart();
    }
    else
    {
        m_m3p_camera.MC3P_PreViewStop();
        m_ctrlpreview.Invalidate();
    }
}

```

---

```
}
```

#### 4.2.1.5 MC3P\_PreviewStop

This function stops preview.

---

```
BOOL MC3P_PreviewStop();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

MC3P\_PreviewStop stops preview image.

---

C++

```
void PreviewStart(BOOL bOn)
{
    if(bOn == TRUE)
    {
        m_m3p_camera.MC3P_PreViewStart();
    }
    else
    {
        m_m3p_camera.MC3P_PreViewStop();
        m_ctrpreview.Invalidate();
    }
}
```

#### 4.2.1.6 MC3P\_Capture

MC3P\_Capture outputs the preview image as JPG or BMP.

---

```
BOOL MC3P_Capture();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

The image captured is affected by the saved options.

---

C++

```
void CaptureImage()
{
    m_m3p_camera.MC3P_Capture();
}
```

#### 4.2.1.7 MC3P\_ReceiveEvent

This function Receives event.

---

```
BOOL MC3P_ReceiveEvent();
```

---

## Parameters

None

## Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

### 4.2.1.8 MC3P\_Set\_CAMERA\_OPTION

This function sets option values.

---

```
BOOL MC3P_Set_CAMERA_OPTION(ref CAM_OPTION option, ref string savefolder);
```

---

## Parameters

*option*

Option value

*savefolder*

Pointer to buffer to receive save folder name

## Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

## Remarks

LPCAM\_OPTION is a structure that contains option values of camera. Once the configurations of the structure are saved, it is used as the input parameter of this function. The captured or preview image is affected by the saved options.

---

C++

```
void SaveCamOption()
{
    INT    nIdx;
    UpdateData();

    //Image Format
    nIdx = m_cbImgFormat.GetCurSel();
    if( nIdx == 0)
        m_pCamOption->nInFormat =OPTION_XLLP_CAMERA_IMAGE_FORMAT_YCBCR422_PACKED;
    else if ( nIdx == 1)
        m_pCamOption->nInFormat = OPTION_XLLP_CAMERA_IMAGE_FORMAT_RGB565;

    //Resolution
    nIdx = m_cbResolution.GetCurSel();
    if (nIdx == 0)
    {
        m_pCamOption->nResolution = OPTION_RESOLUTION_1600X1200;
        m_pCamOption->nImgWidth = 1600;
        m_pCamOption->nImgHeight = 1200;
    }
    else if (nIdx == 1)
    {
        m_pCamOption->nResolution = OPTION_RESOLUTION_1280X1024;
        m_pCamOption->nImgWidth = 1280;
        m_pCamOption->nImgHeight = 1024;
    }
    else if (nIdx == 2)
    {
        m_pCamOption->nResolution = OPTION_RESOLUTION_640X480;
        m_pCamOption->nImgWidth = 640;
        m_pCamOption->nImgHeight = 480;
    }
}
```

```

else if (nIdx == 3)
{
    m_pCamOption->nResolution = OPTION_RESOLUTION_320X240;
    m_pCamOption->nImgWidth = 320;
    m_pCamOption->nImgHeight = 240;
}

//Display RGB Depth
if (m_b16Bit == TRUE)
    m_pCamOption->nPrevRGBDepth = OPTION_PREV_RGB_DEPTH_16BIT;
else if (m_b24Bit == TRUE)
    m_pCamOption->nPrevRGBDepth = OPTION_PREV_RGB_DEPTH_24BIT;

//Save Format
nIdx = m_cbSaveFormat.GetCurSel();
if (nIdx == 0) m_pCamOption->nSaveFormat = OPTION_SAVE_FORMAT_BMP;
else if (nIdx == 1) m_pCamOption->nSaveFormat = OPTION_SAVE_FORMAT_JPG;

//Jpeg Quality
m_pCamOption->nJpegQuality = m_edJpegQuality;

//Image Effect
nIdx = m_cbEffect.GetCurSel();
if(nIdx == 0) m_pCamOption->nImgEffect = OPTION_IMAGE_NOMAL_EFFECT;
else if(nIdx == 1) m_pCamOption->nImgEffect = OPTION_IMAGE_SEPIA_EFFECT;
else if(nIdx == 2)
    m_pCamOption->nImgEffect = OPTION_IMAGE_BLACKWHITE_EFFECT;
else if(nIdx == 3) m_pCamOption->nImgEffect = OPTION_IMAGE_NEGATIVE_EFFECT;
else if(nIdx == 4) m_pCamOption->nImgEffect = OPTION_IMAGE_UVRED_EFFECT;
else if(nIdx == 5) m_pCamOption->nImgEffect = OPTION_IMAGE_UVBLUE_EFFECT;
else if(nIdx == 6)
    m_pCamOption->nImgEffect = OPTION_IMAGE_UVGREEN_EFFECT;

nIdx = m_cbPrefix.GetCurSel();
if(nIdx == 0) m_pCamOption->nNamePrefix = OPTION_IMAGE_FILENAME_PREF_DATE;
else if(nIdx == 1)
    m_pCamOption->nNamePrefix = OPTION_IMAGE_FILENAME_PREF_SERIAL;

nIdx = m_cbRotatesave.GetCurSel();
if(nIdx == 0) { m_pCamOption->nImgRotatesave = IMAGE_ROTATE_0; }
else if(nIdx == 1){ m_pCamOption->nImgRotatesave = IMAGE_ROTATE_90; }
else if(nIdx == 2){ m_pCamOption->nImgRotatesave = IMAGE_ROTATE_180; }
else if(nIdx == 3){ m_pCamOption->nImgRotatesave = IMAGE_ROTATE_270; }

nIdx = m_cbRotateview.GetCurSel();
if(nIdx == 0) { m_pCamOption->nImgRotateview = IMAGE_ROTATE_0; }
else if(nIdx == 1){ m_pCamOption->nImgRotateview = IMAGE_ROTATE_90; }
else if(nIdx == 2){ m_pCamOption->nImgRotateview = IMAGE_ROTATE_180; }
else if(nIdx == 3){ m_pCamOption->nImgRotateview = IMAGE_ROTATE_270; }
m_m3p_camera.MC3P_Set_CAMERA_OPTION(m_pCamOption,
m_edSaveFolder.GetBuffer(0));
}

```

#### 4.2.1.9 MC3P\_Get\_CAMERA\_OPTION

This function sets option value

---

```

BOOL int MC3P_Get_CAMERA_OPTION(LPCAM_OPTION option, TCHAR *savefolder);

```

---

##### Parameters

*option*

Option value [out]

*\*savefolder*

Pointer to buffer to receive save folder name

### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

### Remarks

LPCAM\_OPTION is a structure that contains option values of camera. Option values set up in current module can be brought through this structure.

---

C++

```
void GetOption()  
{  
    m_m3p_camera.MC3P_Get_CAMERA_OPTION(m_pCamOption,  
m_edSaveFolder.GetBuffer(0))  
}
```

#### 4.2.1.10 MC3P\_FlashOFF

This function turns off the flash.

---

```
int MC3P_FlashOFF();
```

---

### Parameters

None

### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

### Remarks

When a flash is required to take a better quality image, this function can be used especially in dark environment.

---

C++

```
void CameraFlashOn(BOOL bOn)  
{  
    if(bOn == TRUE)  
    {  
        m_m3p_camera.MC3P_FlashON();  
    }  
    else  
    {  
        m_m3p_camera.MC3P_FlashOFF();  
    }  
}
```

#### 4.2.1.11 MC3P\_FlashON

This function turns on the flash.

---

```
int MC3P_FlashON();
```

---

### Parameters

None

### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

### Remarks

When a flash is required to take a better quality image, this function can be used especially in dark environment.

---

C++

```
void CameraFlashOn(BOOL bOn)
{
    if(bOn == TRUE)
    {
        m_m3p_camera.MC3P_FlashON();
    }
    else
    {
        m_m3p_camera.MC3P_FlashOFF();
    }
}
```

#### 4.2.1.12 MC3P\_Bright

This function sets up the brightness of camera lense.

---

```
void MC3P_Bright(int nbright);
```

---

### Parameters

*nbright*

Value ranges from 0 to 5

### Return Values

None

### Remarks

MC3P\_Bright adjusts the brightness of camera lense. It can only be applied to 2M camera.

---

C++

```
int m_nBright = 0;

void OnClickBrightUp()
{
    MC3P_Bright(++m_nBright);
}
```

#### 4.2.1.13 MC3P\_RowData

This function gets the preview raw image data

---

```
void MC3P_RowData(byte* data);
```

---

### Parameters

*data*

Raw data

### Return Values

None

### Remarks

MC3P\_RowData gets the raw data seen by preview screen. It is available in later version than M3 GREEN OS Version 230.

---

C++

```
void CM3P_CameraTestDlg::OnButRowdata()
{
```



```

byte    *m_data;
DWORD   dsize;
RECT    rect;
HDC     hPreviewDC = NULL;
HBITMAP hBitmap = NULL;
HBITMAP hOldBitmap = NULL;
HDC     hMemoryDC = NULL;

dsize = m_m3p_camera.MC3P_GetBuffSize();
m_data = new byte[dsize];
memset(m_data, 0, dsize);
m_m3p_camera.MC3P_RowData(m_data);

hPreviewDC = ::GetDC(m_ctrltestview.m_hWnd);
hMemoryDC = CreateCompatibleDC(hPreviewDC);
::GetClientRect(m_ctrltestview.m_hWnd, &rect);
hBitmap = CreateBitmap ( 320, 240, 1, 24, (char*)m_data);
hOldBitmap = (HBITMAP)SelectObject (hMemoryDC, hBitmap);

StretchBlt(hPreviewDC, 0, 0, rect.right-rect.left, rect.bottom-rect.top,
hMemoryDC, 0, 0, 320, 240, SRCCOPY);

SelectObject(hMemoryDC, hOldBitmap);
DeleteObject(hBitmap);
DeleteDC (hMemoryDC);
::ReleaseDC (m_ctrltestview.m_hWnd, hPreviewDC);
delete [] m_data;
}

```

#### 4.2.1.14 MC3P\_GetBuffSize

This function gets size of the preview raw image data.

---

```
DWORD MC3P_GetBuffSize();
```

---

#### Parameters

None

#### Return Values

Size of the preview raw image data.

#### Remarks

It is available in later version than M3 GREEN OS Version 230.

---

C++

```

void CM3P_CameraTestDlg::OnButRowdata()
{
    byte    *m_data;
    DWORD   dsize;
    RECT    rect;
    HDC     hPreviewDC = NULL;
    HBITMAP hBitmap = NULL;
    HBITMAP hOldBitmap = NULL;
    HDC     hMemoryDC = NULL;

    dsize = m_m3p_camera.MC3P_GetBuffSize();
    m_data = new byte[dsize];
    memset(m_data, 0, dsize);
    m_m3p_camera.MC3P_RowData(m_data);

    hPreviewDC = ::GetDC(m_ctrltestview.m_hWnd);
    hMemoryDC = CreateCompatibleDC(hPreviewDC);

```

```
        ::GetClientRect(m_ctrltestview.m_hWnd, &rect);
        hBitmap = CreateBitmap ( 320,240, 1, 24, (char*)m_data);
        hOldBitmap = (HBITMAP)SelectObject (hMemoryDC, hBitmap);

        StretchBlt(hPreviewDC, 0, 0, rect.right-rect.left, rect.bottom-rect.top,
hMemoryDC, 0, 0, 320, 240, SRCCOPY);

        SelectObject(hMemoryDC,hOldBitmap);
        DeleteObject(hBitmap);
        DeleteDC (hMemoryDC);
        ::ReleaseDC (m_ctrltestview.m_hWnd, hPreviewDC);
        delete [] m_data;
    }
```

## 4.2.2 Reference and Function List for C#

### Definitions

#### 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

### Structure

#### CAM\_OPTION

```
public struct CAM_OPTION
{
    public int nImgbalance;
    public int nImgEffect;
    public int nImgExposure;
    public int nImgHeight;
    public int nImgLux;
    public int nImgRotatesave;
    public int nImgRotateview;
    public int nImgWidth;
    public int nInFormat;
    public int nJpegQuality;
    public int nNamePrefix;
    public int nPrevRGBDepth;
    public int nResolution;
    public int nSaveFormat;
}
```

#### 4.2.2.1 MC3P\_Init

This function performs initialization.

---

```
int MC3P_Init (HWND hWnd, HWND p_hWnd);
```

---

##### Parameters

*hWnd*

Main window handle.

*p\_hWnd*

Windows handle to show image.

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

MC3P\_Init is a basic initialization step to run Camera. Please register Window Handle focused on running camera at the first parameter. At second parameter, register Window Handle that will output image. If initialization is succeeded, preview screen will be seen on registered control.

---

C#. NET

```
using M3p_cam_net;

private M3p_cam_net.CamCore camctrl;

public CameraInit()
{
    camctrl = new CamCore();
    camctrl.MC3P_Init(this.Handle, pictureBox1.Handle);

    if(!camctrl.MC3P_Open())
    {
        MessageBox.Show("open error");
    }
}
```

---

#### 4.2.2.2 MC3P\_Open

This function opens a COM port to a camera module.

---

```
int MC3P_Open ();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

MC3P\_Opne function opens a camera port and initializes module at the same time.

---

C#

```
using M3p_cam_net;

private M3p_cam_net.CamCore camctrl;

public CameraInit()
{
    camctrl = new CamCore();
```

---

```

camctrl.MC3P_Init(this.Handle, pictureBox1.Handle);

if(!camctrl.MC3P_Open())
{
    MessageBox.Show("open error");
}
}

```

#### 4.2.2.3 MC3P\_Close

This function closes a COM port to a camera module.

---

```
int MC3P_Open ();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).  
If the function fails, the return value is zero (False).

##### Remarks

MC3P\_Close function is called when the user which to close the camera.

---

C#

```

void CameraClose()
{
    camctrl.MC3P_Close();
}

```

---

#### 4.2.2.4 MC3P\_PreviewStart

This function starts preview.

---

```
int MC3P_PreviewStart();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).  
If the function fails, the return value is zero (False).

##### Remarks

MC3P\_PreviewStart function output the image from camera module through Window Handle registered in MC3P\_Init function.

---

C#

```

void PreviewStart(BOOL bOn)
{
    if(bOn == TRUE)
    {
        camctrl.MC3P_PreviewStart();
    }
    else
    {
        camctrl.MC3P_PreviewStop();
        m_ctrpreview.Invalidate();
    }
}

```

---

#### 4.2.2.5 MC3P\_PreviewStop

This function stops preview.

---

```
int MC3P_PreviewStop();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).  
If the function fails, the return value is zero (False).

##### Remarks

MC3P\_PreviewStop stops preview image.

---

C#

```
void PreviewStart(BOOL bOn)
{
    if(bOn == TRUE)
    {
        camctrl.MC3P_PreViewStart();
    }
    else
    {
        camctrl.MC3P_PreViewStop();
        m_ctrpreview.Invalidate();
    }
}
```

---

#### 4.2.2.6 MC3P\_Capture

MC3P\_Capture outputs the preview image as JPG or BMP.

---

```
int MC3P_Capture();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).  
If the function fails, the return value is zero (False).

##### Remarks

The image captured is affected by the saved options.

---

C#

```
void CaptureImage()
{
    camctrl.MC3P_Capture();
}
```

---

#### 4.2.2.7 MC3P\_ReceiveEvent

This function Receives event.

---

```
int MC3P_ReceiveEvent();
```

---

##### Parameters

None

### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

#### 4.2.2.8 MC3P\_Set\_CAMERA\_OPTION

This function sets option values.

---

```
int MC3P_Set_CAMERA_OPTION(LPCAM_OPTION option, TCHAR *savefolder);
```

---

### Parameters

*option*

Option value

*savefolder*

Pointer to buffer to receive save folder name

### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

### Remarks

LPCAM\_OPTION is a structure that contains option values of camera. Once the configurations of the structure are saved, it is used as the input parameter of this function. The captured or preview image is affected by the saved options.

---

C#

```
void SetOptionClick()  
{  
    if (comboBox_Format.SelectedIndex == 1)  
        m_camoption.nInFormat = 5;  
    else  
        m_camoption.nInFormat = 15;  
    m_camoption.nResolution = comboBox_Resolution.SelectedIndex;  
    if (true == m_b2MCAm)  
    {  
        if (m_camoption.nResolution == 0)  
        {  
            m_camoption.nImgWidth = 1600;  
            m_camoption.nImgHeight = 1200;  
        }  
        if (m_camoption.nResolution == 1)  
        {  
            m_camoption.nImgWidth = 1280;  
            m_camoption.nImgHeight = 1024;  
        }  
        else if (m_camoption.nResolution == 2)  
        {  
            m_camoption.nImgWidth = 640;  
            m_camoption.nImgHeight = 480;  
        }  
        else if (m_camoption.nResolution == 3)  
        {  
            m_camoption.nImgWidth = 320;  
            m_camoption.nImgHeight = 240;  
        }  
    }  
    else  
    {  
        if (m_camoption.nResolution == 0)
```

---

```

    {
        m_camoption.nImgWidth = 1280;
        m_camoption.nImgHeight = 1024;
    }
    if (m_camoption.nResolution == 1)
    {
        m_camoption.nImgWidth = 640;
        m_camoption.nImgHeight = 480;
    }
    else if (m_camoption.nResolution == 2)
    {
        m_camoption.nImgWidth = 320;
        m_camoption.nImgHeight = 240;
    }
}
m_camoption.nSaveFormat = comboBox_saveformat.SelectedIndex;
m_camoption.nNamePrefix = comboBox_picturename.SelectedIndex;
m_camoption.nJpegQuality = (int)numericUpDown_jpec.Value;
m_camoption.nImgRotatesave = 1;
m_camoption.nImgRotateview = 1;
m_szfoldername = textBox_savefolder.Text;
m_pcamcore.MC3P_Set_CAMERA_OPTION(ref m_camoption, ref m_szfoldername);
}

```

#### 4.2.2.9 MC3P\_Get\_CAMERA\_OPTION

This function sets option value

---

```
int MC3P_Get_CAMERA_OPTION(LPCAM_OPTION option, TCHAR *savefolder);
```

---

##### Parameters

*option*

Option value [out]

*\*savefolder*

Pointer to buffer to receive save folder name

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

LPCAM\_OPTION is a structure that contains option values of camera. Option values set up in current module can be brought through this structure.

---

C#

```

void GetOption()
{
    CAM_OPTION pcam_option;
    string szfolder;

    camctrl.MC3P_Get_CAMERA_OPTION(out pcam_option,out szfolder);
}

```

#### 4.2.2.10 MC3P\_FlashOFF

This function turns off the flash.

---

```
int MC3P_FlashOFF();
```

---

##### Parameters



None

#### Return Values

If the function succeeds, the return value is nonzero (True).  
If the function fails, the return value is zero (False).

#### Remarks

When a flash is required to take a better quality image, this function can be used especially in dark environment.

---

C#

```
void CameraFlashOn(BOOL bOn)
{
    if(bOn == TRUE)
    {
        m_m3p_camera.MC3P_FlashON();
    }
    else
    {
        m_m3p_camera.MC3P_FlashOFF();
    }
}
```

---

#### 4.2.2.11 MC3P\_FlashON

This function turns on the flash.

---

int MC3P\_FlashON();

---

#### Parameters

None

#### Return Values

If the function succeeds, the return value is nonzero (True).  
If the function fails, the return value is zero (False).

#### Remarks

When a flash is required to take a better quality image, this function can be used especially in dark environment.

---

C#

```
void CameraFlashOn(BOOL bOn)
{
    if(bOn == TRUE){
        camctrl.MC3P_FlashON();
    }
    else{
        camctrl.MC3P_FlashOFF();
    }
}
```

---

#### 4.2.2.12 MC3P\_Bright

This function sets up the brightness of camera lense.

---

void MC3P\_Bright(int nbright);

---

#### Parameters

*nbright*

Value ranges from 0 to 5

### Return Values

None

### Remarks

MC3P\_Bright adjusts the brightness of camera lense. It can only be applied to 2M camera.

---

C++

```
int m_nBright = 0;

void OnClickBrightUp()
{
    camctrl.MC3P_Bright(++m_nBright);
}
```

---

#### 4.2.1.13 MC3P\_RowData

This function gets the preview raw image data

---

```
void MC3P_RowData(byte* data);
```

---

### Parameters

*data*

Raw data

### Return Values

None

### Remarks

MC3P\_RowData gets the raw data seen by preview screen. It is available in later version than M3 GREEN OS Version 230.

---

C#

```
void CM3P_CameraTestDlg::OnButRowdata()
{
}

}
```

---

#### 4.2.2.14 MC3P\_GetBuffSize

This function gets size of the preview raw image data.

---

```
DWORD MC3P_GetBuffSize();
```

---

### Parameters

None

### Return Values

Size of the preview image raw data.

### Remarks

It is available in later version than M3 GREEN OS Version 230.

---

C#

```
void CM3P_CameraTestDlg::OnButRowdata()
{
}

}
```

---

## 4.3 CDMA

This section provides description of the functions and DLLs which are used to manage the CDMA module.

### Required Products

#### For C++

Required header:

```
DbSqlite.h  
ExtlCmd.h  
global_type.h  
MemoryMap.h  
ReadData.h
```

Required lib:

```
CommMain.lib  
ExtlCDMACmd.lib
```

Required DLL:

```
ExtlCDMACmd.dll  
MCDBEng.dll
```

### Supported Product

M3 GREEN with CDMA option

### 4.3.1 Reference and Function List for C++

#### Definitions

##### 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

#### Structure

##### CDMA

```
typedef struct
{
    CEOID    ceoID;
    short    index;
    short    FirstNum;
    TCHAR    szName[50];
    TCHAR    szMobileTel[20];
    TCHAR    szHomeTel[20];
    TCHAR    szOfficeTel[20];
    short    Group;
}PBITEM, *LPPBITEM;

typedef struct SMSDATA_st
{
    TCHAR    DateTime[18];
    TCHAR    PhoneNum[40];
    TCHAR    Name[30];
    TCHAR    Memo[100];
    TCHAR    Mflag;
} SMSDATA_ST;
```

#### 4.3.1.1 SetUseLib

A function that allows the use of library.

---

```
void SetUseLib();
```

---

##### Parameters

None

##### Return Value

None

##### Remarks

About the library and used **UnSetUseLib** function will be registered on or off.  
Must call before using the library.

#### 4.3.1.2 UnSetUseLib

The library to disable the function

---

```
void UnSetUseLib();
```

---

##### Parameters

None

##### Return Value

None

##### Remarks

About the library and used **SetUseLib** function will be registered on or off.  
After using the library you should call

#### 4.3.1.3 CallInitialize

Library handle of the application to decide whether to register and receive a function of **Noti**.

---

```
void CallInitialize(HWND hNotiTaker, BOOL bGetCDMANoti);
```

---

##### Parameters

*hNotiTaker*

The handle of the application.

*bGetCDMANoti*

CDMA is able to accept the **Noti**.

##### Return Value

None

##### Remarks

In the library used with function over **CallUnInitialize** application will decide whether to register and receive **Noti**.  
Must call before using the library

#### 4.3.1.4 CallUnInitialize

n the library of the application to register a function to disable.

---

```
void CallUnInitialize();
```

---

##### Parameters

None

**Return Value**

None

**Remarks**

About the library and used CallInitialize function to disable the application is registered.  
After using the library you should call

**4.3.1.5 CallMakePhoneCallW**

To connect a voice call function

---

```
void CallMakePhoneCallW(WCHAR* szPhoneNum);
```

---

**Parameters**

*szPhoneNum*

Number of voice calls to connect.

**Return Value**

None

**Remarks**

None

**4.3.1.6 CallMakePhoneCallC**

To connect a voice call function

---

```
void CallMakePhoneCallC(char* szPhoneNum);
```

---

**Parameters**

*szPhoneNum*

Number of voice calls to connect.

**Return Value**

None

**Remarks**

None

**4.3.1.6 CallHangup**

Disconnect the voice call function.

---

```
void CallHangup();
```

---

**Parameters**

None

**Return Value**

None

**Remarks**

None

**4.3.1.7 CallSendSMSU**

To send the SMS function.

---

```
To send the SMS function..
```

---

**Parameters**

*szPhoneNo*

Number to send a message.

*szSMSm*

Send a message.

*szCallBack*

Number of messages sent.

*nPriority*

Priority.

**Return Value**

Error Code

**Remarks**

None

**4.3.1.8 CallSendSMSW**

To send the SMS function.

---

```
UINT CallSendSMSW(WCHAR *szPhoneNo, WCHAR *szSMSm, WCHAR *szCallBack, INT  
nPriority);
```

---

**Parameters**

*szPhoneNo*

Number to send a message.

*szSMSm*

To transmit a message.

*szCallBack*

Number of messages sent.

*nPriority*

Priority.

**Return Value**

Error Code

**Remarks**

None

**4.3.1.9 CallReadSMS**

SMS read function.

---

```
INT CallReadSMS(int nNotiCode, char* szDatetime, char* szCallback, char* szTI, char* szData);
```

---

**Parameters**

*nNotiCode*

Notification Code.

*szDateTime*

SMS the date / time.

*szCallBack*

Transmitting-side contact telephone number.

*szTI*

TI value --> SMS / Call / Other Services Category.

*szData*

SMS Data.

**Return Value**

Error Code

**Remarks**

None

#### 4.3.1.10 CallDialUp

RAS tries to access a function.

---

```
DWORD CallDialUp(HWND hwnd, TCHAR* szPhoneNum, TCHAR* szUserName, TCHAR* szPassword, TCHAR* szEntryName);
```

---

**Parameters**

*hWnd*

Receive a response to the RAS connection handle of the window.

*szPhoneNum*

Access telephone number.

*szUserName*

User Name.

*szPassword*

Password.

*szEntryName*

Entry Name.

**Return Value**

SUCCESS - 0

FAIL - Error Code

**Remarks**

None

#### 4.3.1.11 CallHandleRasEvent

Text information about the Ras Event a function to convert.

---

```
DWORD CallHandleRasEvent(WPARAM wParam, LPARAM lParam, TCHAR* NotiString);
```

---

**Parameters**

*wParam*

Ras Event of wParam.

*lParam*

Ras Event of lParam.

**Return Value**

Ras State

**Remarks**

None



#### 4.3.1.12 CallGetRasState

The function gets the status of RAS access.

---

```
DWORD CallGetRasState();
```

---

##### Parameters

None

##### Return Value

Error Code

##### Remarks

Ras Connect state Code

#### 4.3.1.13 CallHangUpRas

Disable RAS access to the function.

---

```
BOOL CallHangUpRas();
```

---

##### Parameters

None

##### Return Value

Error Code

##### Remarks

None

#### 4.3.1.14 CallRegisterNotiReceiver

CDMA receive from the Noti window should register or revocation.

---

```
INT CallRegisterNotiReceiver(HWND hWnd, BOOL bRegister);
```

---

##### Parameters

*hWnd*

Register or to revocation the handle of the window

*bRegister*

Registration / Revocation whether

##### Return Value

Error Code

##### Remarks

None

#### 4.3.1.15 CallGetNotiData

Receive the application registered in Noti, CDMA from the Noti a message is passed to the window to run this function, the type of Noti / Related Parameter / description gets.

---

```
INT CallGetNotiData(int nSlotNo, char* szParam, PDWORD pParamLen, char* szDesc, PDWORD pszDescLen);
```

---

##### Parameters

*nSlotNo*

MemoryMap Slop Number.

*szParam*

Notification of the Parameter. (Null can be).

*pParamLen*

Parameter Buffer Size.

*szDesc*

A description of the Notification. (Null can be).

*pszDescLen*

szDesc Buffer Size.

#### **Return Value**

CDMA Notification Code

#### **Remarks**

None

#### **4.3.1.15 CallGetNotiDataW**

Receive the application registered in Noti, CDMA from the Noti a message is passed to the window to run this function, the type of Noti / Related Parameter / description gets.

---

```
INT CallGetNotiDataW(int nSlotNo, WCHAR* szParam, PDWORD pParamLen, WCHAR* szDesc, PDWORD pszDescLen);
```

---

#### **Parameters**

*nSlotNo*

MemoryMap Slop Number.

*szParam*

Notification of the Parameter (Null can be).

*pParamLen*

Parameter buffer size.

*szDesc*

A description of the Notification (Null can be).

*pszDescLen*

szDesc buffer size

#### **Return Value**

CDMA Notification Code

#### **Remarks**

None

#### **4.3.1.16 CallGetPhone**

The function to gets the phone number

---

```
INT CallGetPhone(char *phonenum);
```

---

#### **Parameters**

*phonenum*

Phone Number.

**Return Value**

Error Code

**Remarks**

None

**4.3.1.17 CallGetPhoneW**

The function to get the phone number.

---

```
INT CallGetPhoneW(WCHAR *phonenum);
```

---

**Parameters**

*phonenum*

phone number.

**Return Value**

Error Code

**Remarks**

None t

**4.3.1.18 CallGetRSSIGrade**

Grade sensitivity antenna read into the function.

---

```
INT CallGetRSSIGrade();
```

---

**Parameters**

None

**Return Value**

SUCCESS - Sensitivity Grade (0 ~ 7)

FAIL- ERROR\_FAIL

**Remarks**

None

**4.3.1.20 CallSetCallBlock**

Blocks the voice call function.

---

```
BOOL CallSetCallBlock(BOOL bBlock);
```

---

**Parameters**

*bBlock*

Whether to block voice calls.

**Return Value**

Error Code

**Remarks**

None

**4.3.1.21 CallRequestCallBlockStatus**

The function gets the current block status of voice calls.

---

```
BOOL CallRequestCallBlockStatus(HWND hWnd);
```

---

**Parameters**

*hWnd*

The handle of the window.

**Return Value**

Whether the voice call block.

**Remarks**

None

**4.3.1.22 CallGetCDMA Serial**

The function gets the serial number of CDMA.

---

```
BOOL CallGetCDMA Serial(TCHAR* szSerial, TCHAR* szErrStr);
```

---

**Parameters**

*szSerial*

Serial Number of CDMA

*szErrStr*

Error and error information.

**Return Value**

Error Code

**Remarks**

None

**4.3.1.23 CallGetDateTime**

The function gets the current time from CDMA.

---

```
BOOL CallGetDateTime(SYSTEMTIME* pSt, TCHAR* szErrString);
```

---

**Parameters**

*pSt*

The current time in CDMA.

*szErrString*

Error and error information.

**Return Value**

Error Code

**Remarks**

None

**4.3.1.24 CallSetupPowerNoti**

PDA's power status (AC or DC), Wake up event, a function to be set.

---

```
BOOL CallSetupPowerNoti(HWND hwnd);
```

---

**Parameters**

*hWnd*

The handle of the current application.

**Return Value**

Error Code

**Remarks**

None

**4.3.1.25 CallClosePowerNoti**

PDA's power status (AC or DC), Wake up event to disable the function.

---

```
void CallClosePowerNoti();
```

---

**Parameters**

None

**Return Value**

None

**Remarks**

None

## 4.4 GPS

This section provides description of the functions and DLLs which are used to manage the GPS module.

### Required Products

#### For C++

Required header:

M3GpsParse.h

Required lib:

M3GpsParse.Lib

Required DLL:

M3GpsParse.DLL

#### For C#

Required DLL:

M3GpsParse.DLL

### Supported Product

M3 GREEN needs a vehicle cradle to use GPS

#### 4.4.1 Reference and Function List for C++

##### Definitions

##### 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

##### Structure

##### GPSDop

```
struct GPSDop{
    double dPDop;
    double dHDop;
    double dVDop;
};
```

##### GPSSatellite

```
struct GPSSatellite{
    int nID;
    int nElevation;
    int nAzimuth;
    int nSNR;
};
```

##### GPSParseInfo

```
struct GPSParseInfo{
    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;
    CString   mNMEAMsg;
};
```

##### Parameters

M3GPS\_ModuleRestart() functions use parameters defined as below.

Parameter Value Definition	Code	Meaning
GPS_COLD_START	0	Cold start command for GPS
GPS_WARM_START	1	Warm start command for GPS
GPS_HOT_START	2	Hot start command for GPS

##### Windows Message

DLL sends the below message to the user to inform the data received from the satellite. It sends the

message thru windows handle that is registered when GPS open.

Parameter	Value	Definition	Code	Meaning
WM_USER_RECVDATA			WM_USER+10000	Message sent from GPS module to the user



#### 4.4.1.1 M3GPS\_Open

The Open function opens a COM port to a GPS module and performs initialization.

---

```
BOOL M3GPS_Open(HWND hMainWnd, TCHAR* tzComPort, int nBaudRate);
```

---

##### Parameters

*hMainWnd*

Register a window handle to received data from GPS module.

*tzComPort*

Configures the GPS Com port for M3 handheld device.

*nBaudRate*

Configures the Baudrate for communication with M3 handheld device.

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

GPS receives data from the satellite. The data is sent by the DLL thru WM\_USER\_RECEIVEDATA. User can process the data after reception.

---

C++

```
BEGIN_MESSAGE_MAP(CGpsParseDemoDlg, CDialog)
ON_MESSAGE(WM_USER_RECVDATA, OnRecvGPSData)
END_MESSAGE_MAP()

void OpenGps(TCHAR* tzCom, int nBaudRate)
{
    M3GPS_Open(m_hWnd, tzCom, nBaudRate);
}

long CGpsParseDemoDlg::OnRecvGPSData(WPARAM wParam, LPARAM lParam)
{
    GPSParseInfo *pInf = (GPSParseInfo*)wParam;
    return 0;
}
```

#### 4.4.1.2 M3GPS\_Close

The Close function closes a COM port to a GPS module.

---

```
BOOL M3GPS_Close();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

GPS uses serial communication. Hence, closing will disconnect the connection with the satellite and DLL will NOT send messages.

---

C++

```
void CloseGps()
{
    M3GPS_Close();
}
```

#### 4.4.1.3 M3GPS\_ModuleRestart

The ModuleRestart function resets the GPS module.

---

```
BOOL M3GPS_ModuleRestart(int nStartType);
```

---

##### Parameters

*nStartType*

Configures the restart type: Cold, Warm or Hot start.

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

For faster re-connection after an initial connection, GPS module remembers the data such as location and time. Deleting those data will reset the module.

**Cold Start:** A condition in which the GPS receiver can arrive at a navigation solution without initial position, time, and current Ephemeris.

**Warm Start:** Start mode of the GPS receiver when current position, clock offset, and approximate GPS time are input by the user. Ephemeris data is not available.

**Hot Start:** Start mode of the GPS receiver when current position, clock offset, approximate GPS time, and current ephemeris data are all available.

---

C++

```
void ModuleRestart()  
{  
    M3GPS_ModuleRestart(GPS_COLD_START);  
}
```

---

## 4.4.2 Reference and Function List for C#

### Definitions

#### 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

### Structure

#### GPSDop

```
[StructLayout(LayoutKind.Sequential)]
public struct GPS_DOP
{
    public double dPDop;    // Position dilution of precision
    public double dHDop;    // Horizontal dilution of precision
    public double dVDop;    // Vertical dilution of precision
};
```

#### GPSSatellite

```
[StructLayout(LayoutKind.Sequential)]
public struct GPS_SATELLITE
{
    public int nID; // Satellite PRN number
    public int nElevation; // Elevation, degrees
    public int nAzimuth; // Azimuth, degrees
    public int nSNR; // Signal to noise ration in dBHz
};
```

#### GPSParseInfo

```
[DllImport("M3GpsParse.dll")]
private static extern bool M3GPS_Open(IntPtr hMainWnd, string tzComPort, int nBaudRate);

[DllImport("M3GpsParse.dll")]
private static extern bool M3GPS_Close();

[DllImport("M3GpsParse.dll")]
private static extern bool M3GPS_ModuleRestart(int nStartType);

public bool Gps_Open(IntPtr hMainWnd, string tzComPort, int nBaudRate)
{
    return M3GPS_Open(hMainWnd, tzComPort, nBaudRate);
}

public bool Gps_Close()
{
    return M3GPS_Close();
}

public bool Gps_ModuleRestart(int nStarType)
{
    return M3GPS_ModuleRestart(nStarType);
}
```

### Parameters

M3GPS\_ModuleRestart() functions use parameters defined as below.

Parameter Value Definition	Code	Meaning
GPS_COLD_START	0	Cold start command for GPS
GPS_WARM_START	1	Warm start command for GPS
GPS_HOT_START	2	Hot start command for GPS

### **Windows Message**

DLL sends the below message to the user to inform the data received from the satellite. It sends the message thru windows handle that is registered when GPS open.

Parameter Value Definition	Code	Meaning
WM_USER_RECVDATA	0x0400+10000	Message sent from GPS module to the user

#### 4.4.2.1 Class\_Gps\_Parse.GPS\_Open

The Open function opens a COM port to a GPS module and performs initialization.

---

```
Bool Class_Gps_Parse.Gps_Open(IntPtr hMainWnd, string tzComPort, int  
nBaudRate);
```

---

##### Parameters

*hMainWnd*

Register a window handle to received data from GPS module.

*tsComPort*

Configures the GPS Com port for M3 handheld device.

*nBaudRate*

Configures the Baudrate for communication with M3 handheld device.

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

GPS receives data from the satellite. The data is sent by the DLL thru WM\_USER\_RECEIVEDATA. User can process the data after reception.

---

C#

```
namespace GpsParseDemoNet{  
    public partial class Form_GPS : Form{  
        Class_Gps_Parse cGps;  
        MsgWindow MsgWin;  
        IntPtr m_hWnd;  
        private void connect_Gps(string strComport){  
            cGps = new Class_Gps_Parse();  
            MsgWin = new MsgWindow(this);  
            m_hWnd = MsgWin.Hwnd;  
            cGps.Gps_Open(MsgWin.Hwnd, strComport, 9600);  
        }  
        public long OnRecvGpsData(IntPtr wParam, IntPtr lParam){  
            Class_Gps_Parse.GPS_PARSE_INFO info  
                = new Class_Gps_Parse.GPS_PARSE_INFO();  
            info = (Class_Gps_Parse.GPS_PARSE_INFO)  
                Marshal.PtrToStructure(wParam,typeof(Class_Gps_Parse.GPS_PARSE_INFO));  
            return 0;  
        }  
    }  
    public class MsgWindow : MessageWindow{  
        private Form_GPS msgform;  
        public MsgWindow(Form_GPS form){  
            this.msgform = form;  
        }  
        protected override void WndProc(ref Message msg){  
            switch (msg.Msg)  
            {  
                case Class_Gps_Parse.WM_USER_RECVDATA:  
                    this.msgform.OnRecvGpsData(msg.WParam, msg.LParam);  
                    break;  
            }  
            base.WndProc(ref msg);  
        }  
    }  
}
```

#### 4.4.2.2 Class\_Gps\_Parse.Gps\_Close

The Close function closes a COM port to a GPS module.

---

```
Bool    Class_Gps_Parse.Gps_Open(IntPtr    hMainWnd,    string    tzComPort,    int  
nBaudRate);
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

GPS uses serial communication. Hence, closing will disconnect the connection with the satellite and DLL will NOT send messages.

---

C#

```
void CloseGps()  
{  
    cGps.Gps_Close();  
}
```

---

#### 4.4.2.3 Class\_Gps\_Parse.Gps\_ModuleRestart

The ModuleRestart function resets the GPS module.

---

```
BOOL M3GPS_ModuleRestart(int nStartType);
```

---

##### Parameters

*nStartType*

Configures the restart type: Cold, Warm or Hot start.

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

For faster re-connection after an initial connection, GPS module remembers the data such as location and time. Deleting those data will reset the module.

Cold Start: A condition in which the GPS receiver can arrive at a navigation solution without initial position, time, and current Ephemeris.

Warm Start: Start mode of the GPS receiver when current position, clock offset, and approximate GPS time are input by the user. Ephemeris data is not available.

Hot Start: Start mode of the GPS receiver when current position, clock offset, approximate GPS time, and current ephemeris data are all available.

---

C#

```
void ModuleRestart()  
{  
    cGps.Gps_ModuleRestart(Class_Gps_Parse.GPS_COLD_START);  
}
```

---

## 4.5 RFID

This section provides description of the functions and DLLs which are used to manage the RFID module.

### Required Products

#### For C++

Required header:

CAP20\_Mobile.h

Required lib:

CAP20\_Mobile.Lib

Required DLL:

CAP20\_Mobile.DLL

#### For C#

Required DLL:

CAP20\_Mobile.DLL

### Supported Product

M3 GREEN needs a RFID module. Model No.: MC-6X10S

#### 4.5.1 Reference and Function List for C++

##### Definitions

##### **Callback Function**

When RFID Module reads tag, the following function is called. User registers Callback function through CAP20\_RegisterAddRcvData function.

```
// INPUT :  
// hHandle - Port Handle assigned from OpenComPort  
// *buf - Data buffer pointer sent by the reader  
//          First byte refers to the total buffer size,  
//          Second byte is the start of the data.  
  
typedef int (FuncAddRcvData) (HANDLE hHandle, UCHAR *buf);
```



#### 4.5.1.1 CAP20\_OpenCommPort

Serial ports are open between the leader and the Host. This function will start the app in order to communicate with the leader should be run first.

---

```
HANDLE CAP20_OpenCommPort(_TCHAR *port_name, DWORD dwRate= CBR_9600);
```

---

##### Parameters

*\*Port Name*

Port number of RFID module.

*dwRate*

Set up the Baud\_Rate to connect with module.

##### Return Values

Com port Handle, failure Null.

##### Remarks

CAP20\_OpneCommPort is an initialization function to enable the RFID module. After opening it, register the Callback function that will be used when reading the tag.

---

C++

```
#define IOCTL_HAL_RFID_PWR_SEL CTL_CODE(FILE_DEVICE_HAL, 2118 ,  
METHOD_BUFFERED, FILE_ANY_ACCESS)  
// Power Supply  
void OpenRfid()  
{  
    unsigned char ucSel = 1;  
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);  
    m_hComHandle = CAP20_OpenCommPort((unsigned short*)LPCTSTR(L"COM8:")) !=  
    INVALID_HANDLE_VALUE;  
    {  
        // Register Callback Func to proceed Frame data from reader  
        // Can get the data from RFID through this registered function  
        CAP20_RegisterAddRcvData(HandleRcvData);  
  
        //BaudRate Setting  
        CAP20_ConfigCommPort(CBR_115200);  
  
        //Single Read Mode Setting  
        CAP20_SetConfig(0x01, 0x0b, 0x86);  
  
        //Registry Save  
        CAP20_SendCmd(0x01, 0x05);  
    }  
}
```

#### 4.5.1.2 CAP20\_RegisterAddRcvData

Host from a reader in the app to handle a Frame Callback function is registered. This function before the reader and communications should be run first.

Any response from the reader or the tag data is processed by this function.

---

```
void CAP20_RegisterAddRcvData(FuncAddRcvData *func);
```

---

##### Parameters

*func*

CAP20\_RegisterAddRcvData is run when the RFID module reads tag data.

##### Return Values

None

### Remarks

This function runs when COM port is opened. When CAP20\_Unselect\_Read is inputted, registered FuncAddRcvData function reads the tag.

---

```
C++
#define IOCTL_HAL_RFID_PWR_SEL CTL_CODE(FILE_DEVICE_HAL, 2118 ,
METHOD_BUFFERED, FILE_ANY_ACCESS)
// Power Supply
void OpenRfid()
{
    unsigned char ucSel = 1;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);
    m_hComHandle = CAP20_OpenCommPort((unsigned short*)LPCTSTR(L"COM8:")) !=
INVALID_HANDLE_VALUE);
    {
        // Register Callback Func to proceed Frame data from reader
        // Can get the data from RFID through this registered function
        CAP20_RegisterAddRcvData(HandleRcvData);
        CAP20_ConfigCommPort(CBR_115200);

        //BaudRate Setting
        CAP20_SetConfig(0x01, 0x0b, 0x86);

        //Single Read Mode Setting
        CAP20_SendCmd(0x01, 0x05);           //Registry Save
    }
}
```

---

#### 4.5.1.3 FuncAddRcvData

Host of the app to handle Frame Callback function in a reader.

---

```
int (FuncAddRcvData) (HANDLE hHandle, UCHAR *buf);
```

---

### Parameters

*hHandle*

From the penComPort assigned port handle.

*\*buf*

From the reader in the transmit data buffer pointer. The entire data buffer length of the first byte, second byte is the actual data is stored.

### Return Values

None

### Remarks

Register function pointer when you open RFID. This function runs from DLL when the tag is read. The tag data from DLL can be processed within the function.

---

```
C++
void ReadTag()
{
    CAP20_mobile.CAP20_Unselect_Read(0x01, 0, 4);
}

// delegate function part. Register in Open function
public int HandleRcvData(IntPtr hHandle, IntPtr pucDataBuf)
{
    // Get data from DLL in pucDataBuf
    return 1;
}
```

---

#### 4.5.1.4 CAP20\_CloseCommPort

App is called at the end to release all the resources are allocated.

---

```
int CAP20_CloseCommPort();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

CAP20\_CloseCommPort closes RFID program. It would not be done if the Port is opened through CAP20\_OpenCommPort.

---

C++

```
#define IOCTL_HAL_RFID_PWR_SEL CTL_CODE(FILE_DEVICE_HAL, 2118 ,  
METHOD_BUFFERED, FILE_ANY_ACCESS)  
void CloseRfid()  
{  
    CAP20_CloseCommPort();    // Close port  
    unsigned char ucSel = 0;  // not supply power  
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);  
}
```

#### 4.5.1.5 CAP20\_SetConfig

Reader of the configuration register to set a value to a particular spreading.

---

```
BOOL CAP20_SetConfig(UCHAR ucReaderID, UCHAR ucRegAddr, UCHAR ucVal);
```

---

##### Parameters

*ucReaderID*

Reader device number

*ucRegAddr*

Configuration register address

*ucVal*

Register for setting 1byte value

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

CAP20\_SetConfig sets up the way of reading tags. There are two modes - Continuous mode and Normal mode. You have to set up which mode will be used when you open it at first time.

---

C++

```
// RFID Power Control(0:Low, 1:High, 0x2:Status)  
#define IOCTL_HAL_RFID_PWR_SEL CTL_CODE(FILE_DEVICE_HAL, 2118 , METHOD_BUFFERED,  
FILE_ANY_ACCESS)  
void ModeChange()  
{  
    unsigned char ucSel = 0;  
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);  
    Sleep(500);  
    ucSel = 1;  
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);  
}
```

```

Sleep(1000);
if(m_bContinuousMode)
{
    CAP20_SetConfig(0x01, 0x0b, 0x86); // Normal 모드
    Sleep(50);
}
else
{
    CAP20_SetConfig(0x01, 0x1b, 0x04); // Continuous mode
    Sleep(50);
    CAP20_SetConfig(0x01, 0x0b, 0x8c);
    Sleep(50);
}
}

```

#### 4.5.1.6 CAP20\_SendCmd

Reader Get\_Proto\_Info, Get\_FW\_Info, Save config, Set default and the transfer command is invoked

---

```

BOOL CAP20_SendCmd(UCHAR ucReaderID, UCHAR ucCmd);

```

---

##### Parameters

*ucReaderID*

Reader device number

*ucCmd*

Transfer command

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

None

---

C++

```

#define IOCTL_HAL_RFID_PWR_SEL CTL_CODE(FILE_DEVICE_HAL, 2118 ,
METHOD_BUFFERED, FILE_ANY_ACCESS)
// Power Supply
void OpenRfid()
{
    unsigned char ucSel = 1;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);
    m_hComHandle = CAP20_OpenCommPort((unsigned short*)LPCTSTR(L"COM8:")) ! =
INVALID_HANDLE_VALUE);
    {
        // Register Callback Func to proceed Frame data from reader
        // Can get the data from RFID through this registered function
        CAP20_RegisterAddRcvData(HandleRcvData);
        CAP20_ConfigCommPort(CBR_115200);

        //BaudRate Setting
        CAP20_SetConfig(0x01, 0x0b, 0x86);

        //Single Read Mode Setting
        CAP20_SendCmd(0x01, 0x05);           //Registry Save
    }
}

```

#### 4.5.1.7 CAP20\_Unselect\_Read

CAP20\_Unselect\_Read requests to read the tag data at the reader. It is only used for ISO 15693 tags.

---

```
BOOL CAP20_Unselect_Read(UCHAR ucReaderID, UCHAR ucStartBlock, UCHAR ucSize);
```

---

##### Parameters

*ucReaderID*

Reader device number

*ucStartBlock*

Start to receive the request block

*ucSize*

Size

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

When Tag Read is requested, FuncAddRcvData that is registered in CAP20\_OpenCommPort, is called from DLL. User can proceed with data from called function.

---

C++

```
void ReadTag()  
{  
    CAP20_Unselect_Read(0x01, 0, 4)  
}  
// callback function part. Register in Open function  
int HandleRcvData(HANDLE hHandle, UCHAR *pucDataBuf)  
{  
    // Get data from DLL in pucDataBuf  
    return 1;  
}
```

#### 4.5.1.8 CAP20\_Unselect\_Write

CAP20\_Unselect\_Write saves data in tag.

---

```
BOOL CAP20_Unselect_Write(UCHAR ucReaderID, UCHAR ucLen, UCHAR* pucData);
```

---

##### Parameters

*ucReaderID*

Reader device number

*ucLen*

\*pucData length

*\*pucData*

Store data

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

This function is used when user writes data in tags.

---

C++

```
void WriteTag(int nWriteDataLen, BYTE * btWriteData)  
{
```

```
CAP20_Unselect_Write(0x01, nWriteDataLen, btWriteData);
```

```
}
```

## 4.5.2 Reference and Function List for C#

### Definitions

#### Callback Function

When RFID Module reads tag, the following function is called. User registers Callback function through CAP20\_RegisterAddRcvData function.

```
//-----  
  
//Callback func proceeding leader frame of app in host proto type  
// INPUT :  
// hHandle - Port Handle assigned from OpenComPort  
// *buf - Data buffer pointer sent by the reader  
//          First byte refers to the total buffer size,  
//          Second byte is the start of the data.  
  
//-----  
  
public delegate int FuncAddRcvData(IntPtr hHandle, IntPtr buf);
```

#### 4.5.2.1 CAP20\_mobile.CAP20\_OpenCommPort

Serial ports are open between the leader and the Host. This function will start the app in order to communicate with the leader should be run first.

---

```
IntPtr CAP20_OpenCommPort(string port_name, UInt32 dwRate);
```

---

##### Parameters

*\*Port Name*

Port number of RFID module.

*dwRate*

Set up the Baud\_Rate to connect with module.

##### Return Values

Com port Handle, failure Null.

##### Remarks

CAP20\_OpneCommPort is an initialization function to enable the RFID module. After opening it, register the Callback function that will be used when reading the tag.

---

C#

```
private const uint FILE_DEVICE_HAL = 0x00000101;
private const uint METHOD_BUFFERED = 0;
private const uint FILE_ANY_ACCESS = 0;
private IntPtr INVALID_HANDLE_VALUE = new IntPtr(-1);

// RFID Type(0:maintain current status on sleep (Telefunken)), 1: Always Low
// status on sleep (HHE)), 2: Always high status on sleep()), 0xFF: Type Read)
public uint IOCTL_HAL_RFID_TYPE_SEL = CTL_CODE(FILE_DEVICE_HAL, 2117,
METHOD_BUFFERED, FILE_ANY_ACCESS);

// RFID Power Control(0:Low, 1:High, 0x2:Status)
// public uint IOCTL_HAL_RFID_PWR_SEL = CTL_CODE(FILE_DEVICE_HAL, 2118,
// METHOD_BUFFERED, FILE_ANY_ACCESS);

// RFID Reset Control(0:Low, 1:High, 0x2:Status)
// public uint IOCTL_HAL_RFID_RST_SEL = CTL_CODE(FILE_DEVICE_HAL, 2119,
// METHOD_BUFFERED, FILE_ANY_ACCESS);

public static uint CTL_CODE(uint DeviceType, uint Function, uint Method, uint
Access){
    return ((DeviceType << 16) | (Access << 14) | (Function << 2) | Method);
}
void OpenRifd()
{
    string strMsg;
    uint unSel = 1, cout = 0, bytesReturn = 0;
    m_ComHandle = INVALID_HANDLE_VALUE;
    m_bContinuousMode = false;

    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, ref unSel, 1, out cout, 0, ref
bytesReturn);
    m_ComHandle = CAP20_mobile.CAP20_OpenCommPort("COM8:",
CAP20_mobile.CBR_9600);

    if (m_ComHandle != INVALID_HANDLE_VALUE)
    {
        CAP20_mobile.FuncAddRcvData fp =
        new CAP20_mobile.FuncAddRcvData(HandleRcvData);

        // Register Callback Func to proceed Frame data from reader
```

---



```

        // Can get the data from RFID through this registered function
        CAP20_mobile.CAP20_RegisterAddRcvData(fp);
        CAP20_mobile.CAP20_ConfigCommPort(CAP20_mobile.CBR_115200);

        // Single Read Mode Setting
        CAP20_mobile.CAP20_SetConfig(0x01, 0x0b, 0x86);

        //Registry Save
        CAP20_mobile.CAP20_SendCmd(0x01, 0x05);
        strMsg = "Open success" ;
    }
    else
    {
        strMsg = "Open fail";
    }
}

```

#### 4.5.2.2 CAP20\_mobile.CAP20\_RegisterAddRcvData

Host from a reader in the app to handle a Frame Callback function is registered. This function before the reader and communications should be run first.

Any response from the reader or the tag data is processed by this function.

---

```
void CAP20_RegisterAddRcvData(FuncAddRcvData func);
```

---

##### Parameters

*\*func*

CAP20\_RegisterAddRcvData is run when the RFID module reads tag data.

##### Return Values

None

##### Remarks

This function runs when COM port is opened. When CAP20\_Unselect \_Read is inputted, registered FuncAddRcvData function reads the tag.

---

C#

```

private const uint FILE_DEVICE_HAL = 0x00000101;
private const uint METHOD_BUFFERED = 0;
private const uint FILE_ANY_ACCESS = 0;
private IntPtr INVALID_HANDLE_VALUE = new IntPtr(-1);

// RFID Type(0:maintain current status on sleep (Telefunken)), 1: Always Low
// status on sleep (HHE)), 2: Always high status on sleep()), 0xFF: Type Read)
public uint IOCTL_HAL_RFID_TYPE_SEL = CTL_CODE(FILE_DEVICE_HAL, 2117,
METHOD_BUFFERED, FILE_ANY_ACCESS);

// RFID Power Control(0:Low, 1:High, 0x2:Status)
public uint IOCTL_HAL_RFID_PWR_SEL = CTL_CODE(FILE_DEVICE_HAL, 2118,
METHOD_BUFFERED, FILE_ANY_ACCESS);

// RFID Reset Control(0:Low, 1:High, 0x2:Status)
public uint IOCTL_HAL_RFID_RST_SEL = CTL_CODE(FILE_DEVICE_HAL, 2119,
METHOD_BUFFERED, FILE_ANY_ACCESS);

public static uint CTL_CODE(uint DeviceType, uint Function, uint Method, uint
Access)
{
    return ((DeviceType << 16) | (Access << 14) | (Function << 2) | Method);
}

void OpenRifd()

```

```

{
    string strMsg;
    uint unSel = 1, cout = 0, bytesReturn = 0;
    m_ComHandle = INVALID_HANDLE_VALUE;
    m_bContinuousMode = false;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, ref unSel, 1, out cout, 0, ref
bytesReturn);
    m_ComHandle = CAP20_mobile.CAP20_OpenCommPort("COM8:",
CAP20_mobile.CBR_9600);

    if (m_ComHandle != INVALID_HANDLE_VALUE)
    {
        CAP20_mobile.FuncAddRcvData fp =
        new CAP20_mobile.FuncAddRcvData(HandleRcvData);

        // Register Callback Func to proceed Frame data from reader
        // Can get the data from RFID through this registered function
        CAP20_mobile.CAP20_RegisterAddRcvData(fp);
        CAP20_mobile.CAP20_ConfigCommPort(CAP20_mobile.CBR_115200);

        //Single Read Mode Setting
        CAP20_mobile.CAP20_SetConfig(0x01, 0x0b, 0x86);
        //Registry Save
        CAP20_mobile.CAP20_SendCmd(0x01, 0x05);
        strMsg = "Open success" ;
    }
    else
    {
        strMsg = "Open fail";
    }
}

```

#### 4.5.2.3 FuncAddRcvData

Host of the app to handle Frame Callback function in a reader.

---

```
int FuncAddRcvData(IntPtr hHandle, IntPtr buf);
```

---

##### Parameters

*hHandle*

From the penComPort assigned port handle.

*\*buf*

From the reader in the transmit data buffer pointer. The entire data buffer length of the first byte, second byte is the actual data is stored.

##### Return Values

None

##### Remarks

Register function pointer when you open RFID. This function runs from DLL when the tag is read. The tag data from DLL can be processed within the function.

---

C#

```

void ReadTag()
{
    CAP20_Unselect_Read(0x01, 0, 4)
}
// callback function part. Register in Open function
int HandleRcvData(HANDLE hHandle, UCHAR *pucDataBuf)
{
    // Get data from DLL in pucDataBuf

```

```
    return 1;
}
```

#### 4.5.2.4 CAP20\_CloseCommPort

App is called at the end to release all the resources are allocated.

---

```
int CAP20_CloseCommPort();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

CAP20\_CloseCommPort closes RFID program. It would not be done if the Port is opened through CAP20\_OpenCommPort.

---

C#

```
void CloseRfid()
{
    CAP20_mobile.CAP20_CloseCommPort();
    m_ComHandle = INVALID_HANDLE_VALUE;
    uint unSel = 0, cout = 0, bytesReturn = 0;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, ref unSel, 1, out cout, 0, ref
bytesReturn);
}
```

#### 4.5.2.5 CAP20\_mobile.CAP20\_ConfigCommPort

Serial communication speed between the Host and the reader is used to change.

---

```
int CAP20_ConfigCommPort(UINT32 dwBaudRate);
```

---

##### Parameters

*dwBaudRate*

Setting serial communication speed (default: CBR\_9600)

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

Set up the BaudRate after opening RFID.

---

C#

```
private const uint FILE_DEVICE_HAL = 0x00000101;
private const uint METHOD_BUFFERED = 0;
private const uint FILE_ANY_ACCESS = 0;
private IntPtr INVALID_HANDLE_VALUE = new IntPtr(-1);

// RFID Type(0:maintain current status on sleep (Telefunken)), 1: Always Low
// status on sleep (HHE)), 2: Always high status on sleep()), 0xFF: Type Read)
public uint IOCTL_HAL_RFID_TYPE_SEL = CTL_CODE(FILE_DEVICE_HAL, 2117,
METHOD_BUFFERED, FILE_ANY_ACCESS);
// RFID Power Control(0:Low, 1:High, 0x2:Status)
public uint IOCTL_HAL_RFID_PWR_SEL = CTL_CODE(FILE_DEVICE_HAL, 2118,
```

```

METHOD_BUFFERED, FILE_ANY_ACCESS);
// RFID Reset Control(0:Low, 1:High, 0x2:Status)
public uint IOCTL_HAL_RFID_RST_SEL = CTL_CODE(FILE_DEVICE_HAL, 2119,
METHOD_BUFFERED, FILE_ANY_ACCESS);
public static uint CTL_CODE(uint DeviceType, uint Function, uint Method, uint
Access)
{
    return ((DeviceType << 16) | (Access << 14) | (Function << 2) | Method);
}
void OpenRifd()
{
    string strMsg;
    uint unSel = 1, cout = 0, bytesReturn = 0;
    m_ComHandle = INVALID_HANDLE_VALUE;
    m_bContinuousMode = false;
    kernelIoControl(IOCTL_HAL_RFID_PWR_SEL, ref unSel, 1, out cout, 0, ref
bytesReturn);
    m_ComHandle = CAP20_mobile.CAP20_OpenCommPort("COM8:",
CAP20_mobile.CBR_9600);
    if (m_ComHandle != INVALID_HANDLE_VALUE)
    {
        CAP20_mobile.FuncAddRcvData fp =
        new CAP20_mobile.FuncAddRcvData(HandleRcvData);

        // Register Callback Func to proceed Frame data from reader
        // Can get the data from RFID through this registered function
        CAP20_mobile.CAP20_RegisterAddRcvData(fp);
        CAP20_mobile.CAP20_ConfigCommPort(CAP20_mobile.CBR_115200);

        //Single Read Mode Setting
        CAP20_mobile.CAP20_SetConfig(0x01, 0x0b, 0x86);
        //Registry Save
        CAP20_mobile.CAP20_SendCmd(0x01, 0x05);
        strMsg = "Open success" ;
    }
    else
    {
        strMsg = "Open fail";
    }
}
}

```

#### 4.5.2.6 CAP20\_mobile.CAP20\_SetConfig

Reader of the configuration register sets a value to a particular spreading.

---

```

BOOL CAP20_SetConfig(Byte cReaderID, Byte cRegAddr, Byte cVal);

```

---

##### Parameters

*ucReaderID*

Reader device number

*ucRegAddr*

Configuration register address

*ucVal*

Register for setting 1byte value

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

CAP20\_SetConfig sets up the way of reading tags. There are two modes - Continuous mode and Normal mode. You have to set up which mode will be used when you open it at first time.

---

```
C#
// RFID Power Control(0:Low, 1:High, 0x2:Status)
#define IOCTL_HAL_RFID_PWR_SEL CTL_CODE(FILE_DEVICE_HAL, 2118 ,
METHOD_BUFFERED, FILE_ANY_ACCESS)

void ModeChange()
{
    unsigned char ucSel = 0;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);

    Sleep(500);
    ucSel = 1;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, &ucSel, 1, NULL, 0, NULL);

    Sleep(1000);
    if(m_bContinuousMode)
    {
        CAP20_SetConfig(0x01, 0x0b, 0x86); // Normal 모드
        Sleep(50);
    }
    else
    {
        CAP20_SetConfig(0x01, 0x1b, 0x04); // Continuous mode
        Sleep(50);

        CAP20_SetConfig(0x01, 0x0b, 0x8c);
        Sleep(50);
    }
}
```

---

#### 4.5.2.7 CAP20\_mobile.CAP20\_SendCmd

Reader Get\_Proto\_Info, Get\_FW\_Info, Save config, Set default and the transfer command is invoked

---

```
BOOL CAP20_SendCmd(Byte cReaderID, Byte cCmd);
```

---

##### Parameters

*ucReaderID*

Reader device number

*ucCmd*

Transfer command

##### Return Values

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### Remarks

None

---

```
C#
private const uint FILE_DEVICE_HAL = 0x00000101;
private const uint METHOD_BUFFERED = 0;
private const uint FILE_ANY_ACCESS = 0;
private IntPtr INVALID_HANDLE_VALUE = new IntPtr(-1);

// RFID Type(0:maintain current status on sleep (Telefunken)), 1: Always Low
// status on sleep (HHE)), 2: Always high status on sleep()), 0xFF: Type Read)
public uint IOCTL_HAL_RFID_TYPE_SEL = CTL_CODE(FILE_DEVICE_HAL, 2117,
```

---

```

METHOD_BUFFERED, FILE_ANY_ACCESS);

// RFID Power Control(0:Low, 1:High, 0x2:Status)
public uint IOCTL_HAL_RFID_PWR_SEL = CTL_CODE(FILE_DEVICE_HAL, 2118,
METHOD_BUFFERED, FILE_ANY_ACCESS);

// RFID Reset Control(0:Low, 1:High, 0x2:Status)
public uint IOCTL_HAL_RFID_RST_SEL = CTL_CODE(FILE_DEVICE_HAL, 2119,
METHOD_BUFFERED, FILE_ANY_ACCESS);

public static uint CTL_CODE(uint DeviceType, uint Function, uint Method, uint
Access){
    return ((DeviceType << 16) | (Access << 14) | (Function << 2) | Method);
}
void OpenRifd()
{
    string strMsg;
    uint unSel = 1, cout = 0, bytesReturn = 0;
    m_ComHandle = INVALID_HANDLE_VALUE;
    m_bContinuousMode = false;
    KernelIoControl(IOCTL_HAL_RFID_PWR_SEL, ref unSel, 1, out cout, 0, ref
bytesReturn);
    m_ComHandle = CAP20_mobile.CAP20_OpenCommPort("COM8:",
CAP20_mobile.CBR_9600);

    if (m_ComHandle != INVALID_HANDLE_VALUE)
    {
        CAP20_mobile.FuncAddRcvData fp =
        new CAP20_mobile.FuncAddRcvData(HandleRcvData);

        // Register Callback Func to proceed Frame data from reader
        // Can get the data from RFID through this registered function
        CAP20_mobile.CAP20_RegisterAddRcvData(fp);
        CAP20_mobile.CAP20_ConfigCommPort(CAP20_mobile.CBR_115200);

        //Single Read Mode Setting
        CAP20_mobile.CAP20_SetConfig(0x01, 0x0b, 0x86);
        //Registry Save
        CAP20_mobile.CAP20_SendCmd(0x01, 0x05);
        strMsg = "Open success" ;
    }
    else
    {
        strMsg = "Open fail";
    }
}
}

```

#### 4.5.2.8 CAP20\_mobile.CAP20\_Unselect\_Read

CAP20\_Unselect\_Read requests to read the tag data at the reader. It is only used for ISO 15693 tags.

---

```

BOOL CAP20_Unselect_Read(UCHAR ucReaderID, UCHAR ucStartBlock, UCHAR ucSize);

```

---

##### Parameters

*ucReaderID*

Reader device number

*ucStartBlock*

Start to receive the request block

*ucSize*

Size

##### Return Values

If the function succeeds, the return value is nonzero (True).  
If the function fails, the return value is zero (False).

#### Remarks

When Tag Read is requested, FuncAddRcvData that is registered in CAP20\_OpenCommPort, is called from DLL. User can proceed with data from called function.

---

```
C#
void ReadTag()
{
    CAP20_mobile.CAP20_Unselect_Read(0x01, 0, 4);
}

// delegate function part. Register in Open function
public int HandleRcvData(IntPtr hHandle, IntPtr pucDataBuf)
{
    // Get data from DLL in pucDataBuf
    return 1;
}
```

---

#### 4.5.2.9 CAP20\_mobile.CAP20\_Unselect\_Write

CAP20\_Unselect\_Write saves data in tag.

---

```
BOOL CAP20_Unselect_Write(Byte cReaderID, Byte ucLen, Byte[] pucData);
```

---

#### Parameters

*ucReaderID*

Reader device number

*ucLen*

\*pucData length

*\*pucData*

Store data

#### Return Values

If the function succeeds, the return value is nonzero (True).  
If the function fails, the return value is zero (False).

#### Remarks

This function is used when user writes data in tags.

---

```
C#
void WriteTag(int nWriteDataLen, Byte [] btWriteData)
{
    CAP20_mobile.CAP20_Unselect_Write(0x01, (Byte)nWriteDataLen, btWriteData);
}
```

---

## 4.6 Scanner 1D

This section provides description of the functions and DLLs which are used to manage 1D scanner module.

### Required Products

#### For C++

Required header:

```
KScnaBar.h  
Option.h  
Option_defs.h
```

Required lib:

```
KScanBar.lib
```

Required DLL:

```
KScanBar.dll
```

#### For C#

Required DLL:

```
MCSSLib.dll  
MCSSLibNet.dll
```

### Supported Product

M3 GREEN with 1D scanner that uses software decoder



## 4.6.1 Reference and Function List for C++

### Definitions

#### 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

### Structure

#### 1D Scanner

```
typedef struct tagKSCANREAD {
    int                nSize;
    unsigned            nTimeInSeconds;
    unsigned long       dwFlags;
    unsigned long       dwReadType;
    int                nMinLength;
    int                nSecurity;
    KS_FN_CALLBACK      fnCallBack;
    LPVOID              pUserData;
    LPVOID              pReadEx;
    int                out_Status;
    int                out_Type;
    char                out_Barcode[2711];
} KSCANREAD;

typedef KSCANREAD *PKSCANREAD;
typedef struct tagKSCANREADEX {
    int                nI2of5MinLength;
    int                nI2of5MaxLength;
    int                nCodabarMinLength;
    int                nCodabarMaxLength;

    #if defined(QUIETZONE_CHECK_OPTION_APPEND)
    && !defined(ALWAYS_IGNORE_QUIETZONE)
        unsigned long dwIgnoreQZCheck; // Bit : Code128, EAN,
    #endif

    HWND             hwnd;
    DWORD            UserMsg;
} KSCANREADEX;

typedef KSCANREADEX* PKSCANREADEX;
typedef struct tagKSCANREADEX2 {
    char Signature[8]; // "KSCANEX2"
    HWND             hwnd;
    DWORD            UserMsg;
    DEC_UPCEAN       UpcA;
    DEC_UPCEAN       UpcE;
    DEC_UPCEAN       Ean13;
    DEC_UPCEAN       Ean8;
    DEC_CODE39        Code39;
    DEC_CODE128        Code128;
    DEC_CODE93         Code93;
    DEC_CODE35         Code35;
    DEC_CODE11         Code11; // CheckDigit : 0, Disable, 1 : 1 Digit, 2 : 2 Digit
// 추후.
    DEC_CODE25         Code25;
    DEC_CODABAR        Codabar;
```

```

DEC_MSI      Msi;
DEC_PLESSEY  Plessey;
DEC_GS1      Gs1;
DEC_GS1_LIMITED  Gs1Limited;
DEC_GS1_EXPANDED  Gs1Expanded;
DEC_TELEPEN  Telepen;
ABLE         XmitAIMID; //2009.09.30
} KSCANREADEX2;
typedef KSCANREADEX2* PKSCANREADEX2;

```

## **Parameters**

M3M\_SetParameter() functions use parameters defined as below.

```

#define KSCAN_RET_TYPE_EAN_13      0
#define KSCAN_RET_TYPE_EAN_8      1
#define KSCAN_RET_TYPE_UPCA       2
#define KSCAN_RET_TYPE_UPCE       3
#define KSCAN_RET_TYPE_CODE_39    4
#define KSCAN_RET_TYPE_ITF_14     5
#define KSCAN_RET_TYPE_CODE_128   6
#define KSCAN_RET_TYPE_CODE_I25   7
#define KSCAN_RET_TYPE_CODA_BAR   8
#define KSCAN_RET_TYPE_UCCEAN_128 9
#define KSCAN_RET_TYPE_CODE_93    10
#define KSCAN_RET_TYPE_CODE_35    11
#define KSCAN_RET_TYPE_PDF417     12
#define KSCAN_RET_TYPE_MACRO_PDF417 13
#define KSCAN_RET_TYPE_BOOKLAND   14
#define KSCAN_RET_TYPE_MSI        15
#define KSCAN_RET_TYPE_PZN        16
#define KSCAN_RET_TYPE_PLESSEY    17
#define KSCAN_RET_TYPE_MACRO_PDF417_INC 18
#define KSCAN_RET_TYPE_MACRO_PDF417_ERROR 19
#define KSCAN_RET_TYPE_CODE25_MATRIX 20
#define KSCAN_RET_TYPE_CODE25_DLOGIC 21
#define KSCAN_RET_TYPE_CODE25_INDUSTRY 22
#define KSCAN_RET_TYPE_CODE25_IATA  23
#define KSCAN_RET_TYPE_CODE25_GTIN14 24
#define KSCAN_RET_TYPE_CODE25_DPL   25
#define KSCAN_RET_TYPE_CODE25_DPI   26
#define KSCAN_RET_TYPE_CODE11       27
#define KSCAN_RET_TYPE_CODE32       28
#define KSCAN_RET_TYPE_COUPONCODE   29
#define KSCAN_RET_TYPE_CODABLOCK_A  30
#define KSCAN_RET_TYPE_CODABLOCK_F  31
#define KSCAN_RET_TYPE_GS1          32      // RSS_14
#define KSCAN_RET_TYPE_GS1_LIMITED  33      // RSS_LIMITED
#define KSCAN_RET_TYPE_GS1_EXPANDED  34      // RSS_EXPENDED
#define KSCAN_RET_TYPE_STANDARD2OF5 35
#define KSCAN_RET_TYPE_TELEPEN      36
#define KSCAN_RET_TYPE_UNKNOWN      0xFF

```

## **Enum**

```

typedef enum {
    DISABLE=0,
    ENABLE
}ABLE;

typedef enum {
    FULL_ASCII=0,
    STD_ASCII

```

```

}XMIT_ASCII;

typedef enum {
    NO_Supp=0,
    WITH_OR_WITHOUT
}SUPP;

typedef enum {
    XMIT_NUMBER=0,
    NO_XMIT_NUMBER
}XMIT_NUM_SYSTEM;

typedef enum {
    XMIT_CHECK_DIGIT=0,
    NO_XMIT_CHECK_DIGIT
}XMIT_CHECK_CHAR;

typedef enum {
    NO_XMIT=0,
    XMIT
}XMIT_STARTSTOP;

typedef enum {
    AS_UPCA=0,
    AS_EAN13,
    AS_UPCE,
    AS_EAN8,
    AS_CODE32,
    AS_UCCEAN128,
    AS_BOOKLAND
}FORMAT;

typedef enum {
    ANGLE_WIDE=0,
    ANGLE_NARROW
}ENGINE_ANGLE;

typedef enum {
    FILTER_WIDE=0,
    FILTER_NARROW
}ENGINE_FILTER;

typedef enum {
    MODULO10 = 0,
    MODULO11,
    MODULO1010,
    MODULO1110
}MOD_METHOD;

typedef enum {
    CODE25KIND_INTER    = 0x01, // 1,
    CODE25KIND_MATRIX   = 0x02, // 2,
    CODE25KIND_DLOGIC    = 0x04, // 4,
    CODE25KIND_INDUSTRY  = 0x08, // 8,
    CODE25KIND_IATA      = 0x10, // 16,
    CODE25KIND_ITF14     = 0x20, // 32,
}CODE25_KIND;

// Jiyong modified - spain
typedef enum {
    DIGIT0=0,
    DIGIT1,
    DIGIT2
}CHECK_DIGIT;

```

#### 4.6.1.1 CKScan::Open

The Open function opens a COM port to a Scan Module, and performs initialization.

---

```
BOOL Open(  
    int CommNumber = 0,  
    BOOL CommDetect = FALSE,  
    DWORD BaudRate = CBR_115200,  
    BOOL BaudDetect = FALSE,  
    DWORD ExFlags = 0  
);
```

---

##### **Parameters**

###### *CommNumber*

Specifies the serial port number to be opened. Range: 0~16. When 0 is specified automatic scan of Com port is enabled. The default is 0.

###### *CommDetect*

Specifies automatic scanning of Com port number when Scan Module was not present at CommNumber. The order of scanning is: Number, 1...16. The default is FALSE.

###### *BaudRate*

Specifies baud rate for Com port. The default is CBR\_115200.

###### *BaudDetect*

Specifies automatic scanning of baud rate when the Scan Module was not found using BaudRate specified. The order of scanning is: BaudRate, CBR\_115200, CBR\_38400, CBR\_57600. The default is FALSE.

###### *ExFlags*

Specifies optional flags for future compatibility. Must be set to zero.

##### **Return Values**

If the function succeeds, the return value is nonzero (True).

If the function fails, the return value is zero (False).

##### **Remarks**

It is necessary to pair this function with CKScan::Close for proper operation and resource management.

Depending on operating system, Com port hardware, Com port device driver and their implementation, this function may take a long time (2+ seconds) to successfully open and initialize the Scan Module. It is recommended to use a verified numbers for CommNumber and BaudRate to minimize execution time. It should also be noted that Com port number designated by operating system can be changed due to configuration changes usually caused by reinstallation and/or addition/deletion of hardware/software. It is recommended to utilize CommDetect initially, and to reuse detected Com port number for further operation in the same session.

The maximum Baud Rate defined is 115,200 bps on desktop operating systems. Depending on the platform and its operating system implementation, the maximum Baud Rate defined varies. For baud rates beyond defined maximum, there are a few techniques employed. Please refer to the platform and serial device driver reference.

---

C++

```
BOOL bRet = m_KScan.Open(6, FALSE, CBR_115200, FALSE, NULL);  
If(bRet == FALSE)  
    ::MessageBox(NULL, L"Error : Scanner Open Failed", NULL, MB_TOPMOST);
```

---

#### 4.6.1.2 CKScan::Close

The Close function closes the COM port to a Scan Module, and may put the Scan Module into sleep mode.

---

```
BOOL Open(  
    int CommNumber = 0,
```

---

---

```
    BOOL CommDetect = FALSE,  
    DWORD BaudRate = CBR_115200,  
    BOOL BaudDetect = FALSE,  
    DWORD ExFlags = 0  
);
```

---

#### **Parameters**

None

#### **Return Values**

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero.

#### **Remarks**

It is necessary to pair Close() with CKScan::Open for proper operation and resource management.

It is advised to fully utilize sleep mode capability of the Scan Module. When battery preservation is critical, users may want to implement CKScan::Sleep and CKScan::Wakeup for every operation.

---

C++

```
BOOL bRet = m_KScan.Close();  
If(bRet == FALSE)  
    ::MessageBox(NULL, L"Error : Scanner Close Failed", NULL, MB_TOPMOST);
```

---

#### **4.6.1.3 CKScan::Read**

The Read function reads a barcode from the Scan Module. This function call can be either blocking or non-blocking depending on parameters in the KSCANREAD structure.

---

```
BOOL Read(  
    PKSCANREAD    pRead  
);
```

---

#### **Parameters**

*pRead*

Specifies pointer to the KSCANREAD Structure.

#### **Return Values**

If the function succeeds, the return value is nonzero.

If the function fails or times out, the return value is zero.

#### **Remarks**

This function is valid only after a successful call to CKScan::Open.

Reading a barcode is delicate and complicated process. It is highly recommended that users read Reading a barcode with proper options section for proper understanding of these options and flags.

Please refer to included sample program for details of blocking and non-blocking operations. Non-blocking operation is a delicate process that provides applications with more control how the scan module reads barcodes. It is recommended that users take advantage of simple blocking read operation unless non-blocking operation is absolutely necessary.

If the scan engine is in sleep mode, it is necessary to issue CKScan::Wakeup first, otherwise the function call will result in fail.

---

C++

```
void CMainSheet::M3_ScanRead()  
{  
    CString Str;  
    BOOL bRet;  
    m_bReading = TRUE;  
  
    if (m_bReading)
```

---

```

{ // Reading already in progress, now cancel it.
  bRet = m_KScan.ReadCancel();

  if (bRet)
  {
    m_bReading = FALSE;
  }
  else
  {
    Str = _T("Error ReadCancel\r\n");
    Str += KScanGetLastErrorMsg();
  }
}
m_bReading = TRUE;
bRet = m_KScan.Read(&kRead);
if (!bRet) {
  Str.Format(_T("Error Read \r\n%d\r\n%s"), kRead.out_Status,
KScanGetLastErrorMsg());
  m_bReading = FALSE;
}
}

```

#### 4.6.1.4 CKScan::ReadCancel

The ReadCancel function cancels CKScan::ReadForever or CKScan::Read that was initiated as non-blocking operation. Its use is strongly discouraged.

---

```

BOOL ReadCancel ()

```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero.

If the function fails or times out, the return value is zero.

##### Remarks

This function is valid only after a successful non-blocking call to CKScan::Read. Please note that this function is not applicable when read operation was initiated as blocking read.

Please refer to included sample program for details of non-blocking operations. Non-blocking operation is a delicate process that provides applications with more control how the scan module reads barcodes. It is recommended that users take advantage of simple blocking read operation unless non-blocking operation is absolutely necessary.

The reading process makes uses of multiple threads and synchronization primitives. An orderly termination via the call back function is always preferred. The use of ReadCancel function should be avoided to terminate the read. Using the ReadCancel function multiple times to terminate the read operation may lead to unstable application.

ReadCancel tries to maintain the system integrity while terminating the threads and restoring the synchronization primitives. Only when these primitives do not respond, does it forcibly terminate them. The function thus sometimes takes up to 1 second to complete its tasks.

---

C++

```

BOOL bRet = m_KScan.ReadCancel();

```

---

#### 4.6.1.5 CKScan::ReadForever

The ReadForever function is a specialized form of the CKScan::Read function. The limitations are:

- It must always be in non-blocking mode.

- It only looks for 1D barcodes
- It does not terminate itself whether a barcode is found or not

The function terminates when the call back function returns 0 (see CallBack Function for Non-blocking Read).

---

```

BOOL ReadForever(
    PKSCANREAD  pRead
);

```

---

#### **Parameters**

*pRead*

Specifies pointer to the KSCANREAD Structure. The parameter nTimeInSeconds is ignored.

#### **Return Values**

If the function succeeds, the return value is TRUE.

If the function fails or times out, the return value is FALSE.

#### **Remarks**

All the remarks for CKScan::Read are applicable here.

ReadForever is an extremely powerful function, albeit a very fragile process that provides applications with even more control of how the scan module reads barcodes. The reading process makes uses of multiple threads and synchronization primitives. An orderly termination via the call back function is always preferred. The use of CKScan::ReadCancel function should be avoided to terminate the read. Using the CKScan::ReadCancel function multiple times to terminate the read operation may lead to unstable application.

#### **4.6.1.6 CKScan::GetCommName**

The GetCommName function returns a pointer to the name of current COM port with CKScan::Open.

---

```

LPCTSTR GetCommName( ) ;

```

---

#### **Parameters**

None

#### **Return Values**

The function returns a pointer to the name of current COM port such as "COM1:". If there is no COM port open, the return string is "None".

#### **Remarks**

None

#### **4.6.1.7 CKScan::GetLastErrorMsg**

The GetLastErrorMsg function returns a pointer to a string with description of the last error occurred in KScanBar.dll.

---

```

LPCTSTR GetLastErrorMsg( ) ;

```

---

#### **Parameters**

None

#### **Return Values**

The function returns a pointer to a string with description of the last error occurred in KScanBar.dll.

#### **Remarks**

None

#### 4.6.1.8 CKScan::GetVersionInfo

The GetVersionInfo function returns a pointer to a string with version information associated with the current port.

---

```
LPCTSTR GetVersionInfo();
```

---

##### **Parameters**

None

##### **Return Values**

The function returns a pointer to a string with version information associated with the current port. If no Scan Module is open, the return information excludes hardware version information.

##### **Remarks**

None

#### 4.6.1.9 CKScan::Sleep

The Sleep function puts the Scan Module into sleep mode.

---

```
BOOL Sleep();
```

---

##### **Parameters**

None

##### **Return Values**

If the function succeeds, the return value is nonzero.  
If the function fails, the return value is zero.

##### **Remarks**

It is advised to fully utilize sleep mode capability of the Scan Module. KSCAN\_SLEEP\_NOW puts the Scan Module into power saving sleep mode allowing only minimum current drain. It should be noted that waking up from sleep mode will incur additional time of up to 10 msec.

#### 4.6.1.10 CKScan::Wakeup

The Wakeup function wakes up the Scan Module that was previously put in sleep mode.

---

```
BOOL KScanWakeup();
```

---

##### **Parameters**

None

##### **Return Values**

If the function succeeds, the return value is nonzero.  
If the function fails, the return value is zero.

##### **Remarks**

Typical time delay for waking up from sleeping mode is measured to be less than 10 msec.

#### 4.6.1.10 CKScan::Reset

The Reset function resets the Scan Module.

---

```
BOOL Reset();
```

---

##### **Parameters**

None



**Return Values**

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero.

**Remarks**

This function is valid only after a successful call to CKScan::Open.

If the scan engine is in sleep mode, it is necessary to issue CKScan::Wakeup first, otherwise the function call will result in fail.

## 4.6.2 Reference and Function List for C#

### Definitions

#### 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

### Structure

#### 1D Scanner

```
typedef struct _MCBarCodeType
{
    // BARCODE 28
    BYTE bMC_UPCA;
    BYTE bMC_UPCA_ADDON;
    BYTE bMC_UPCE;
    BYTE bMC_EAN13;
    BYTE bMC_EAN13_ADDON;
    BYTE bMC_BOOKLAND;
    BYTE bMC_EAN8;
    BYTE bMC_CODE39;
    BYTE bMC_CODE32;
    BYTE bMC_PZN;
    BYTE bMC_CODE128;
    BYTE bMC_UCCEAN128;
    BYTE bMC_CODE93;
    BYTE bMC_CODE35;
    BYTE bMC_CODE11;
    BYTE bMC_I2OF5;
    BYTE bMC_CODE25_ITF14;
    BYTE bMC_CODE25_MATRIX;
    BYTE bMC_CODE25_DLOGIC;
    BYTE bMC_CODE25_INDUSTRY;
    BYTE bMC_CODE25_IATA;
    BYTE bMC_CODABAR;
    BYTE bMC_COUPON;
    BYTE bMC_MSI;
    BYTE bMC_PLESSEY;
    BYTE bMC_GS1;
    BYTE bMC_GS1_LIMITED;
    BYTE bMC_GS1_EXPANDED;
    BYTE bMC_TELEPEN;
}MCBarCodeType,* PMCBarCodeType;

typedef struct _MCReadOption
{
    BYTE bMC_WIDESCANGANGLE;
    BYTE bMC_RETURNCHECK;
    BYTE bMC_ERRORCHECK;
    BYTE bMC_HIGHFILTERMODE;
}MCReadOption,* PMCReadOption;

typedef struct _MCModuleOption
{
    int nMC_TimeOutSec;
    int nMC_MinLen;
    int nMC_SecurityLevel;
```

```

}MCModuleOption,* PMCModuleOption;

////////////////////////////////////
// BARCODE OPTION STRUCT
typedef struct _MCBarOption_UPCA{
    BYTE    bMC_UPCA_Enable;
    BYTE    bMC_UPCA_XNum;
    BYTE    bMC_UPCA_XCD;
    BYTE    bMC_UPCA_AS_EAN13;
    BYTE    bMC_UPCA_AddOn;
}BarOption_UPCA, *PBarOption_UPCA;

typedef struct _MCBarOption_UPCE{
    BYTE    bMC_UPCE_Enable;
    BYTE    bMC_UPCE_XNum;
    BYTE    bMC_UPCE_XCD;
    int     nMC_UPCE_Convert;
}BarOption_UPCE, *PBarOption_UPCE;

typedef struct _MCBarOption_EAN13{
    BYTE    bMC_EAN13_Enable;
    BYTE    bMC_BOOKLAND_Enable;
    BYTE    bMC_EAN13_XCD;
    BYTE    bMC_EAN13_AddOn;
}BarOption_EAN13, *PBarOption_EAN13;

typedef struct _MCBarOption_EAN8{
    BYTE    bMC_EAN8_Enable;
    BYTE    bMC_EAN8_XCD;
    BYTE    bMC_EAN8_AS_EAN13;
}BarOption_EAN8, *PBarOption_EAN8;

typedef struct _MCBarOption_CODE39{
    BYTE    bMC_CODE39_Enable;
    BYTE    bMC_CODE32_Enable;
    BYTE    bMC_PZN_Enable;
    BYTE    bMC_CODE39_CDV;
    BYTE    bMC_CODE39_XCD;
    BYTE    bMC_CODE39_FullASCII;
    int     nMC_CODE39_MinLen;
    int     nMC_CODE39_MaxLen;
}BarOption_CODE39, *PBarOption_CODE39;

typedef struct _MCBarOption_CODE128{
    BYTE    bMC_CODE128_Enable;
    BYTE    bMC_UCCEAN128_Enable;
    int     nMC_CODE128_MinLen;
    int     nMC_CODE128_MaxLen;
}BarOption_CODE128, *PBarOption_CODE128;

typedef struct _MCBarOption_CODE93{
    BYTE    bMC_CODE93_Enable;
    int     nMC_CODE93_MinLen;
    int     nMC_CODE93_MaxLen;
}BarOption_CODE93, *PBarOption_CODE93;

typedef struct _MCBarOption_CODE35{
    BYTE    bMC_CODE35_Enable;
}BarOption_CODE35, *PBarOption_CODE35;

typedef struct _MCBarOption_CODE11{
    BYTE    bMC_CODE11_Enable;
    BYTE    bMC_CODE11_XCD;

```

```

    int    nMC_CODE11_CDV;
    int    nMC_CODE11_MinLen;
    int    nMC_CODE11_MaxLen;
}BarOption_CODE11, *PBarOption_CODE11;

typedef struct _MCBarOption_I2OF5{
    BYTE    bMC_I2OF5_Enable;
    BYTE    bMC_ITF14_Enable;
    BYTE    bMC_MATRIX2OF5_Enable;
    BYTE    bMC_DLOGIG_Enable;
    BYTE    bMC_INDUSTRY_Enable;
    BYTE    bMC_IATA_Enable;
    BYTE    bMC_I2OF5_CDV;
    BYTE    bMC_I2OF5_XCD;
    int     nMC_I2OF5_MinLen;
    int     nMC_I2OF5_MaxLen;
}BarOption_I2OF5, *PBarOption_I2OF5;

typedef struct _MCBarOption_CODABAR{
    BYTE    bMC_CODABAR_Enable;
    BYTE    bMC_CODABAR_XSS;
    int     nMC_CODABAR_MinLen;
    int     nMC_CODABAR_MaxLen;
}BarOption_CODABAR, *PBarOption_CODABAR;

typedef struct _MCBarOption_MSI{
    BYTE    bMC_MSI_Enable;
    BYTE    bMC_MSI_CDV;
    BYTE    bMC_MSI_XCD;
    int     nMC_MSI_MinLen;
    int     nMC_MSI_MaxLen;
}BarOption_MSI, *PBarOption_MSI;

typedef struct _MCBarOption_PLESSEY{
    BYTE    bMC_PLESSEY_Enable;
    BYTE    bMC_PLESSEY_CDV;
    BYTE    bMC_PLESSEY_XCD;
    int     nMC_PLESSEY_MinLen;
    int     nMC_PLESSEY_MaxLen;
}BarOption_PLESSEY, *PBarOption_PLESSEY;

typedef struct _MCBarOption_GS1{
    BYTE    bMC_GS1_Enable;
    BYTE    bMC_GS1LIM_Enable;
    BYTE    bMC_GS1EXP_Enable;
}BarOption_GS1, *PBarOption_GS1;

typedef struct _MCBarOption_TELEPEN{
    BYTE    bMC_TELEPEN_Enable;
    BYTE    bMC_TELEPEN_OldStyle;
}BarOption_TELEPEN, *PBarOption_TELEPEN;;

```

#### 4.6.2.1 ScannerControl.ScanInit

The ScanInit function opens a COM port to Scan Module, and performs initialization.

---

```
int ScanInit();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is ERROR\_NONE = 0

If the function fails, the return value is POWER\_ON\_ERROR = -1

If the function fails, the return value is PORT\_OPEN\_ERROR = -3

##### Remarks

It is necessary to pair this function with ScanClose() for proper operation and resource management.

If not call ScanClose() after a successful call to ScanInit(), next call to ScanInit() will fail

Depending on operating system, Com port hardware, Com port device driver and their implementation, this function may take a long time (2+ seconds) to successfully open and initialize the Scan Module.

---

C#

```
Bool m_nResult = ScanCtrl.ScanInit();  
if(m_nResult != 0)  
{  
    MessageBox.Show("Scanner Init Failed");  
}
```

---

#### 4.6.2.2 ScannerControl.ScanClose

The ScanClose function closes the COM port to a Scan Module.

---

```
int ScanClose ();
```

---

##### Parameters

None

##### Return Values

If the function fails, the return value is POWER\_OFF\_ERROR = -2

If the function fails, the return value is PORT\_CLOSE\_ERROR = -4

If the function succeeds, the return value is ERROR\_NONE= 0

##### Remarks

It is necessary to pair Close() with ScanInit() for proper operation and resource management.

If not call this function after a successful call to ScanInit(), next call to ScanInit() will fail.

---

C#

```
ScanCtrl.ScanClose();
```

---

#### 4.6.2.3 ScannerControl.ScanRead

The ScanRead function reads a barcode from the Scan Module.

---

```
int ScanRead ();
```

---

##### Parameters

None

##### Return Values

If the function succeeds, the return value is nonzero.

If the function fails or times out, the return value is zero.

## Remarks

This function is valid only after a successful call to ScanInit()

---

C#

```
public void ScanRead()  
{  
    if (m_bReading == true)  
    {  
        ScanCtrl.ScanReadCancel();  
        m_bReading = false;  
        return;  
    }  
    m_bReading = false;  
    ScanCtrl.ScanRead();  
}
```

### 4.6.2.4 ScannerControl.ScanReadCancel

The ReadCancel function cancels ScanRead that was initiated as non-blocking operation. Its use is strongly discouraged.

---

```
int ScanReadCancel ();
```

---

## Parameters

None

## Return Values

If the function succeeds, the return value is nonzero.

If the function fails or times out, the return value is zero.

## Remarks

This function is valid only after a successful non-blocking call to ScanRead(). Please note that this function is not applicable when read operation was initiated as blocking read.

If not call to Readcancel(), Barcode reading is cancelled after timeout.

ReadCancel tries to maintain the system integrity while terminating the threads and restoring the synchronization primitives. Only when the these primitives do not respond, does it forcibly terminate them. The function thus sometimes take up to 1 second to complete its tasks.

---

C#.

```
ScanCtrl.ScanReadCancel();
```

---

### 4.6.2.5 ScannerControl.SetBarcodeType

The SetBarCodeType function Set Structure of Symbology

---

```
void SetBarCodeType(ref MCBarCodeType pBCT)
```

---

## Parameters

*MCBarCodeType*

Specifies pointer to the MCSSLibNet Structure

## Return Values

None

## Remarks

Use after successful initialization of the scanner

---

C#

```
private MCSSLibNet.MCBarCodeType M3BarCodeType;  
  
M3BarCodeType = new MCBarCodeType();
```

---

```

M3BarcodeType.bMC_UPCA          = TRUE;

M3BarcodeType.bMC_UPCE          = TRUE;
M3BarcodeType.bMC_EAN13         = TRUE;
M3BarcodeType.bMC_BOOKLAND      = TRUE;
M3BarcodeType.bMC_EAN8          = TRUE;
M3BarcodeType.bMC_CODE39        = TRUE;
M3BarcodeType.bMC_CODE32        = TRUE;
M3BarcodeType.bMC_PZN           = TRUE;
M3BarcodeType.bMC_CODE128       = TRUE;
M3BarcodeType.bMC_UCCEAN128     = TRUE;
M3BarcodeType.bMC_CODE93        = TRUE;
M3BarcodeType.bMC_CODE35        = TRUE;
M3BarcodeType.bMC_CODE11        = TRUE;
M3BarcodeType.bMC_I2OF5         = TRUE;
M3BarcodeType.bMC_MSI           = TRUE;
M3BarcodeType.bMC_PLESSEY       = TRUE;
M3BarcodeType.bMC_CODABAR       = TRUE;
M3BarcodeType.bMC_GS1           = TRUE;
M3BarcodeType.bMC_GS1_LIMITED   = TRUE;
M3BarcodeType.bMC_GS1_EXPANDED  = TRUE;

ScanCtrl.SetBarcodeType(ref M3BarcodeType);

```

#### 4.6.2.6 ScannerControl.GetBarcodeType

The GetBarcodeType function gets Structure of Symbology

---

```
void GetBarcodeType(out MBarcodeType pBCT)
```

---

##### Parameters

*MBarcodeType*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

---

```

C#
private MCSSLibNet.MBarcodeType M3BarcodeType;

M3BarcodeType = new MBarcodeType();
// Barcode

public bool m_bUpca;
public bool m_bUpce;
public bool m_bEan13;
public bool m_bBookland;
public bool m_bEan8;
public bool m_bCode39;
public bool m_bCode32;
public bool m_bPzn;
public bool m_bCode128;
public bool m_bUccean128;
public bool m_bCode93;
public bool m_bCode35;
public bool m_bCode11;
public bool m_bI2of5;
public bool m_bMsi;

```

---

```

public bool m_bPlessey;
public bool m_bCodabar;
public bool m_bGsl;
public bool m_bGslLim;
public bool m_bGslExp;

ScanCtrl.GetBarCodeType(out M3BarCodeType);

m_bUpca      = M3BarCodeType.bMC_UPCA;
m_bUpce      = M3BarCodeType.bMC_UPCE;
m_bEan13     = M3BarCodeType.bMC_EAN13;
m_bBookland  = M3BarCodeType.bMC_BOOKLAND;
m_bEan8      = M3BarCodeType.bMC_EAN8;
m_bCode39    = M3BarCodeType.bMC_CODE39;
m_bCode32    = M3BarCodeType.bMC_CODE32;
m_bPzn       = M3BarCodeType.bMC_PZN;
m_bCode128   = M3BarCodeType.bMC_CODE128;
m_bUccean128 = M3BarCodeType.bMC_UCCEAN128;
m_bCode93    = M3BarCodeType.bMC_CODE93;
m_bCode35    = M3BarCodeType.bMC_CODE35;
m_bCode11    = M3BarCodeType.bMC_CODE11;
m_bI2of5     = M3BarCodeType.bMC_I2OF5;
m_bMsi       = M3BarCodeType.bMC_MSI;
m_bPlessey   = M3BarCodeType.bMC_PLESSEY;
m_bCodabar   = M3BarCodeType.bMC_CODABAR;
m_bGsl       = M3BarCodeType.bMC_GSL;
m_bGslLim    = M3BarCodeType.bMC_GSL_LIMITED;
m_bGslExp    = M3BarCodeType.bMC_GSL_EXPANDED;

```

#### 4.6.2.7 ScannerControl.SetModuleOption

The SetModuleOption function set Structure of ModuleOption

---

```
void SetModuleOption(ref MCMModuleOption pMDO)
```

---

##### **Parameters**

*MCMModuleOption*

Specifies pointer to the MCSSLibNet Structure

##### **Return Values**

None

##### **Remarks**

Use after successful initialization of the scanner

---

C#

```

private MCSSLibNet.MCMModuleOption M3ModuleOption;

M3ModuleOption = new MCMModuleOption();
M3ModuleOption.nMC_TimeOutSec = 10;
M3ModuleOption.nMC_SecurityLevel = 1;
M3ModuleOption.nMC_MinLen = 3

ScanCtrl.SetModuleOption(ref M3ModuleOption);

```

---

#### 4.6.2.8 ScannerControl.GetModuleOption

The GetModuleOption function gets Structure of ModuleOption

---

```
void GetModuleOption(out MCMModuleOption pMDO)
```

---



**Parameters***MCMModuleOption*

Specifies pointer to the MCSSLibNet Structure

**Return Values**

None

**Remarks**

Use after successful initialization of the scanner

---

C#

```
public int m_nSecurityLevel;
public int m_nTimeOut;
public int m_nMinLen;

private MCSSLibNet.MCMModuleOption M3ModuleOption;

M3ModuleOption = new MCMModuleOption();

ScanCtrl.GetModuleOption(out M3ModuleOption);

m_nSecurityLevel = M3ModuleOption.nMC_TimeOutSec;
m_nTimeOut       = M3ModuleOption.nMC_SecurityLevel;
m_nMinLen        = M3ModuleOption.nMC_MinLen;
```

---

**4.6.2.9 ScannerControl.SetReadOption**

The SetReadOption function set Structure of ReadOption

---

void SetReadOption(ref MCMReadOption pRDO)

---

**Parameters***MCMReadOption*

Specifies pointer to the MCSSLibNet Structure

**Return Values**

None

**Remarks**

Use after successful initialization of the scanner

---

C#

```
private MCSSLibNet.MCMReadOption M3ReadOption;

M3ReadOption = new MCMReadOption();

M3ReadOption.bMC_WIDESCANANGLE = true;
M3ReadOption.bMC_HIGHFILTERMODE = false;
M3ReadOption.bMC_RETURNCHECK    = true;
M3ReadOption.bMC_ERRORCHECK     = false;

ScanCtrl.SetReadOption(ref M3ReadOption);
```

---

**4.6.2.10 ScannerControl.GetReadOption**

The SetReadOption function set Structure of ReadOption

---

void SetReadOption(ref MCMReadOption pRDO)

---

**Parameters***MCMReadOption*

Specifies pointer to the MCSSLibNet Structure

#### Return Values

None

#### Remarks

Use after successful initialization of the scanner

---

C#

```
public bool m_bERRORCHECK;  
public bool m_bHIGHFILTERMODE;  
public bool m_bRETURNCHECK;  
public bool m_bWIDESCANANGLE;  
  
private MCSSLibNet.MCReadOption M3ReadOption;  
  
M3ReadOption = new MCReadOption();  
  
ScanCtrl.GetReadOption(out M3ReadOption);  
  
m_bWIDESCANANGLE = M3ReadOption.bMC_WIDESCANANGLE;  
m_bHIGHFILTERMODE = M3ReadOption.bMC_HIGHFILTERMODE;  
m_bRETURNCHECK = M3ReadOption.bMC_RETURNCHECK;  
m_bERRORCHECK = M3ReadOption.bMC_ERRORCHECK;
```

#### 4.6.2.11 ScannerControl.GetVersionInfo

The GetVersionInfo function returns a string with version information.

---

```
string GetVersionInfo()
```

---

#### Parameters

None

#### Return Values

The function returns a string with version information associated with the current port.

#### Remarks

If no Scan Module is open, the return information excludes hardware version information.

#### 4.6.2.12 ScannerControl.RegisterReceiveForm

The RegisterRecieveForm This function registers a form as a receiver of scan message.

---

```
void RegisterRecieveForm()
```

---

#### Parameters

None

#### Return Values

None

#### Remarks

If a form is intended to process scan message, this function should be called when a form is initialized or activated.

#### 4.6.2.13 ScannerControl.UseDefaultSound

The UseDefaultSound function sound come after finishing ScanRead Success.

---

```
void UseDefaultSound(bool bSound, string sndPath)
```

---

**Parameters**

*bSound*

Determines whether Sound is used or not. True if default sound is used and false if default sound is not used.

*sndPath*

Assign the path where Wav files are located.

In the case of NULL, the default wav is played.

**Return Values**

None

**Remarks**

Default sound is \Windows\alarm4.wav

#### 4.6.2.14 ScannerControl.UseResumeMsg

The UseResumeMsg function is Default Display Scanner Initialize... message window when resuming.

---

```
void UseResumeMsg(bool bResume)
```

---

**Parameters**

*bResume*

Determines whether message is used or not. True if message is used and false if message is not used.

**Return Values**

None

**Remarks**

Default sound is \Windows\alarm4.wav

#### 4.6.2.15 ScannerControl.RegHotKey

The RegHotKey function registers Scanner Button as a hot key.

---

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

---

**Parameters**

*Id*

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

*vk*

Virtual Key value

*SyncMode*

In the case of If true? -> SyncMode

In the case of If false?-> AsyncMode

**Return Values**

If the function succeeds, the return value is nonzero.

If the function fails or times out, the return value is zero.

**Remarks**

This function can be used in CF 1.x because of it's complexity to get the event regarding KeyDown and KeyUp.

If RegHotkey is used as hotkey, it should be freed before the application ends.

#### 4.6.2.16 ScannerControl.UnRegHotKey

The UnRegHotKey function Free Scanner Button as a hot key.

---

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

---

##### Parameters

*Id*

Identifier of the hot key to be freed

##### Return Values

If the function succeeds, the return value is nonzero.

If the function fails or times out, the return value is zero.

##### Remarks

This function can be used in CF 1.x because of it's complexity to get the event regarding KeyDown and KeyUp.

If RegHotkey is used as hotkey, it should be freed before the application ends.

#### 4.6.2.17 ScannerControl.SetBarOptionUPCA

The SetBarOptionUPCA function set the option of UPC-A Barcode.

---

```
void SetBarOptionUPCA(ref MCBBarOption_UPCA pUpca);
```

---

##### Parameters

*MCBarOption\_UPCA*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_UPCA pUpca = new MCBBarOption_UPCA();

pUpca.bMC_UPCA_Enable = m_bUpca = UpcaDlg.m_bEnable;
pUpca.bMC_UPCA_XNum = UpcaDlg.m_bXNum;
pUpca.bMC_UPCA_XCD = UpcaDlg.m_bXCD;
pUpca.bMC_UPCA_AS_EAN13 = UpcaDlg.m_bUPCAasEAN13;
pUpca.bMC_UPCA_AddOn = UpcaDlg.m_bAddOn;

ScanCtrl.SetBarOptionUPCA(ref pUpca);
```

#### 4.6.2.18 ScannerControl.GetBarOptionUPCA

The GetBarOptionUPCA function gets the option of UPC-A Barcode.

---

```
void GetBarOptionUPCA(out MCBBarOption_UPCA pUpca);
```

---

##### Parameters

*MCBarOption\_UPCA*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

---

C#

```
MCTBarOption_UPCA pUpca = new MCTBarOption_UPCA();

ScanCtrl.GetBarOptionUPCA(out pUpca);

UpcaDlg.m_bEnable = pUpca.bMC_UPCA_Enable;
UpcaDlg.m_bXNum = pUpca.bMC_UPCA_XNum;
UpcaDlg.m_bXCD = pUpca.bMC_UPCA_XCD;
UpcaDlg.m_bUPCAasEAN13 = pUpca.bMC_UPCA_AS_EAN13;
UpcaDlg.m_bAddOn = pUpca.bMC_UPCA_AddOn;
```

---

#### 4.6.2.19 ScannerControl.SetBarOptionUPCE

The SetBarOptionUPCE function set the option of UPC-E Barcode.

---

```
void SetBarOptionUPCE(ref MCTBarOption_UPCE pUpce);
```

---

##### **Parameters**

*MCTBarOption\_UPCE*

Specifies pointer to the MCSSLibNet Structure

##### **Return Values**

None

##### **Remarks**

Use after successful initialization of the scanner

---

C#

```
MCTBarOption_UPCE pUpce = new MCTBarOption_UPCE();

pUpce.bMC_UPCE_Enable = m_bUpce = UpceDlg.m_bEnable;
pUpce.bMC_UPCE_XNum = UpceDlg.m_bXNum;
pUpce.bMC_UPCE_XCD = UpceDlg.m_bXCD;
pUpce.nMC_UPCE_Convert = UpceDlg.m_nConvert;

ScanCtrl.SetBarOptionUPCE(ref pUpce);
```

---

#### 4.6.2.20 ScannerControl.GetBarOptionUPCE

The GetBarOptionUPCA function gets the option of UPC-E Barcode.

---

```
void GetBarOptionUPCE(out MCTBarOption_UPCE pUpce);
```

---

##### **Parameters**

*MCTBarOption\_UPCE*

Specifies pointer to the MCSSLibNet Structure

##### **Return Values**

None

##### **Remarks**

Use after successful initialization of the scanner

---

C#

```
MCTBarOption_UPCE pUpce = new MCTBarOption_UPCE();

ScanCtrl.GetBarOptionUPCE(out pUpce);

UpceDlg.m_bEnable = pUpce.bMC_UPCE_Enable;
UpceDlg.m_bXNum = pUpce.bMC_UPCE_XNum;
```

---

```
UpceDlg.m_bXCD = pUpce.bMC_UPCE_XCD;  
UpceDlg.m_nConvert = pUpce.nMC_UPCE_Convert;
```

#### 4.6.2.21 ScannerControl.SetBarOptionEAN13

The SetBarOptionEAN13 function set the option of EAN-13 Barcode.

```
void SetBarOptionEAN13(ref MCBBarOption_EAN13 pEan13);
```

##### Parameters

*MCBarOption\_EAN13*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

C#

```
MCBarOption_EAN13 pEan13 = new MCBBarOption_EAN13();  
  
pEan13.bMC_EAN13_Enable = m_bEan13 = Ean13Dlg.m_bEnable;  
pEan13.bMC_BOOKLAND_Enable = m_bBookland = Ean13Dlg.m_bBookland;  
pEan13.bMC_EAN13_XCD = Ean13Dlg.m_bXCD;  
pEan13.bMC_EAN13_AddOn = Ean13Dlg.m_bAddOn;  
  
ScanCtrl.SetBarOptionEAN13(ref pEan13);
```

#### 4.6.2.22 ScannerControl.GetBarOptionEAN13

The SetBarOptionEAN13 function gets the option of EAN-13 Barcode.

```
void GetBarOptionEAN13(out MCBBarOption_EAN13 pEan13);
```

##### Parameters

*MCBarOption\_EAN13*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

C#

```
MCBarOption_EAN13 pEan13 = new MCBBarOption_EAN13();  
  
ScanCtrl.GetBarOptionEAN13(out pEan13);  
  
Ean13Dlg.m_bEnable = pEan13.bMC_EAN13_Enable;  
Ean13Dlg.m_bBookland = pEan13.bMC_BOOKLAND_Enable;  
Ean13Dlg.m_bXCD = pEan13.bMC_EAN13_XCD;  
Ean13Dlg.m_bAddOn = pEan13.bMC_EAN13_AddOn;
```

#### 4.6.2.23 ScannerControl.SetBarOptionEAN8

The SetBarOptionEAN8 function sets the option of EAN-8 Barcode.

```
void SetBarOptionEAN8(ref MCBBarOption_EAN8 pEan8);
```

**Parameters**

*MCBarOption\_EAN8*

Specifies pointer to the MCSSLibNet Structure

**Return Values**

None

**Remarks**

Use after successful initialization of the scanner

---

C#

```
MCBarOption_EAN8 pEan8 = new MCBarOption_EAN8();

pEan8.bMC_EAN8_Enable = m_bEan8 = Ean8Dlg.m_bEnable;
pEan8.bMC_EAN8_XCD = Ean8Dlg.m_bXCD;
pEan8.nMC_EAN8_AS_EAN13 = Ean8Dlg.m_bEAN8asEAN13;

ScanCtrl.SetBarOptionEAN8(ref pEan8);
```

---

**4.6.2.24 ScannerControl.GetBarOptionEAN8**

The GetBarOptionEAN8 function gets the option of EAN-8 Barcode.

---

```
void GetBarOptionEAN8(out MCBarOption_EAN8 pEan8);
```

---

**Parameters**

*MCBarOption\_EAN8*

Specifies pointer to the MCSSLibNet Structure

**Return Values**

None

**Remarks**

Use after successful initialization of the scanner

---

C#

```
MCBarOption_EAN8 pEan8 = new MCBarOption_EAN8();

ScanCtrl.GetBarOptionEAN8(out pEan8);

Ean8Dlg.m_bEnable = pEan8.bMC_EAN8_Enable;
Ean8Dlg.m_bXCD = pEan8.bMC_EAN8_XCD;
Ean8Dlg.m_bEAN8asEAN13 = pEan8.nMC_EAN8_AS_EAN13;
```

---

**4.6.2.25 ScannerControl.SetBarOptionCODE39**

The SetBarOptionCODE39 function set the option of CODE39 Barcode.

---

```
void SetBarOptionCODE39(ref MCBarOption_CODE39 pCode39);
```

---

**Parameters**

*MCBarOption\_CODE39*

Specifies pointer to the MCSSLibNet Structure

**Return Values**

None

**Remarks**

Use after successful initialization of the scanner

---

C#

```
MBarOption_CODE39 pCode39 = new MBarOption_CODE39();

pCode39.bMC_CODE39_Enable = m_bCode39 = Code39Dlg.m_bEnable;
pCode39.bMC_CODE32_Enable = m_bCode32 = Code39Dlg.m_bCode32;
pCode39.bMC_PZN_Enable = m_bPzn = Code39Dlg.m_bPzn;
pCode39.bMC_CODE39_CDV = Code39Dlg.m_bCDV;
pCode39.bMC_CODE39_XCD = Code39Dlg.m_bXCD;
pCode39.bMC_CODE39_FullASCII = Code39Dlg.m_bFullASCII;
pCode39.nMC_CODE39_MinLen = Code39Dlg.m_nMinLen;
pCode39.nMC_CODE39_MaxLen = Code39Dlg.m_nMaxLen;

ScanCtrl.SetBarOptionCODE39(ref pCode39);
```

---

#### 4.6.2.26 ScannerControl.GetBarOptionCODE39

The GetBarOptionCODE39 function gets the option of CODE39 Barcode.

---

```
void GetBarOptionCODE39(out MBarOption_CODE39 pCode39);
```

---

##### Parameters

*MBarOption\_CODE39*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

---

C#

```
MBarOption_CODE39 pCode39 = new MBarOption_CODE39();

ScanCtrl.GetBarOptionCODE39(out pCode39);

Code39Dlg.m_bEnable = pCode39.bMC_CODE39_Enable;
Code39Dlg.m_bCode32 = pCode39.bMC_CODE32_Enable;
Code39Dlg.m_bPzn = pCode39.bMC_PZN_Enable;
Code39Dlg.m_bCDV = pCode39.bMC_CODE39_CDV;
Code39Dlg.m_bXCD = pCode39.bMC_CODE39_XCD;
Code39Dlg.m_bFullASCII = pCode39.bMC_CODE39_FullASCII;
Code39Dlg.m_nMinLen = pCode39.nMC_CODE39_MinLen;
Code39Dlg.m_nMaxLen = pCode39.nMC_CODE39_MaxLen;
```

---

#### 4.6.2.27 ScannerControl.SetBarOptionCODE128

The SetBarOptionCODE128 function sets the option of CODE128 Barcode.

---

```
void SetBarOptionCODE128(ref MBarOption_CODE128 pCode128);
```

---

##### Parameters

*MBarOption\_CODE128*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner



---

C#

```
MCBarOption_CODE128 pCode128 = new MCBarOption_CODE128();

pCode128.bMC_CODE128_Enable = m_bCode128 = Code128Dlg.m_bEnable;
pCode128.bMC_UCCEAN128_Enable = m_bUccean128 = Code128Dlg.m_bUccean128;
pCode128.nMC_CODE128_MinLen = Code128Dlg.m_nMinLen;
pCode128.nMC_CODE128_MaxLen = Code128Dlg.m_nMaxLen;

ScanCtrl.SetBarOptionCODE128(ref pCode128);
```

---

#### 4.6.2.28 ScannerControl.SetBarOptionCODE128

The GetBarOptionCODE128 function gets the option of CODE128 Barcode.

---

```
void GetBarOptionCODE128(out MCBarOption_CODE128 pCode128);
```

---

##### **Parameters**

*MCBarOption\_CODE128*

Specifies pointer to the MCSSLibNet Structure

##### **Return Values**

None

##### **Remarks**

Use after successful initialization of the scanner

---

C#

```
MCBarOption_CODE128 pCode128 = new MCBarOption_CODE128();

ScanCtrl.GetBarOptionCODE128(out pCode128);

Code128Dlg.m_bEnable = pCode128.bMC_CODE128_Enable;
Code128Dlg.m_bUccean128 = pCode128.bMC_UCCEAN128_Enable;
Code128Dlg.m_nMinLen = pCode128.nMC_CODE128_MinLen;
Code128Dlg.m_nMaxLen = pCode128.nMC_CODE128_MaxLen;
```

---

#### 4.6.2.29 ScannerControl.SetBarOptionCODE93

The SetBarOptionCODE93 function sets the option of CODE93 Barcode.

---

```
void SetBarOptionCODE93(ref MCBarOption_CODE93 pCode93);
```

---

##### **Parameters**

*MCBarOption\_CODE93*

Specifies pointer to the MCSSLibNet Structure

##### **Return Values**

None

##### **Remarks**

Use after successful initialization of the scanner

---

C#

```
MCBarOption_CODE93 pCode93 = new MCBarOption_CODE93();

pCode93.bMC_CODE93_Enable = m_bCode93 = Code93Dlg.m_bEnable;
pCode93.nMC_CODE93_MinLen = Code93Dlg.m_nMinLen;
pCode93.nMC_CODE93_MaxLen = Code93Dlg.m_nMaxLen;

ScanCtrl.SetBarOptionCODE93(ref pCode93);
```

---

#### 4.6.2.30 ScannerControl.GetBarOptionCODE93

The GetBarOptionCODE93 function gets the option of CODE93 Barcode.

---

```
void GetBarOptionCODE93(out MCBBarOption_CODE93 pCode93);
```

---

##### Parameters

*MCBarOption\_CODE93*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_CODE93 pCode93 = new MCBBarOption_CODE93();

ScanCtrl.GetBarOptionCODE93(out pCode93);

Code93Dlg.m_bEnable = pCode93.bMC_CODE93_Enable;
Code93Dlg.m_nMinLen = pCode93.nMC_CODE93_MinLen;
Code93Dlg.m_nMaxLen = pCode93.nMC_CODE93_MaxLen;
```

#### 4.6.2.31 ScannerControl.SetBarOptionCODE35

The SetBarOptionCODE35 function sets the option of CODE35 Barcode.

---

```
void SetBarOptionCODE35(ref MCBBarOption_CODE35 pCode35);
```

---

##### Parameters

*MCBarOption\_CODE35*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_CODE35 pCode35 = new MCBBarOption_CODE35();

pCode35.bMC_CODE35_Enable = m_bCode35 = Code35Dlg.m_bEnable;

ScanCtrl.SetBarOptionCODE35(ref pCode35);
```

#### 4.6.2.32 ScannerControl.GetBarOptionCODE35

The GetBarOptionCODE35 function gets the option of CODE35 Barcode.

---

```
void GetBarOptionCODE35(out MCBBarOption_CODE35 pCode35);
```

---

##### Parameters

*MCBarOption\_CODE35*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_CODE35 pCode35 = new MCBarOption_CODE35();

ScanCtrl.GetBarOptionCODE35(out pCode35);

Code35Dlg.m_bEnable = pCode35.bMC_CODE35_Enable;
```

---

#### 4.6.2.33 ScannerControl.SetBarOptionCODE11

The SetBarOptionCODE11 function sets the option of CODE11 Barcode.

---

```
void SetBarOptionCODE11(ref MCBarOption_CODE11 pCode11);
```

---

### Parameters

*MCBarOption\_CODE11*

Specifies pointer to the MCSSLibNet Structure

### Return Values

None

### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_CODE11 pCode11 = new MCBarOption_CODE11();

pCode11.bMC_CODE11_Enable = m_bCode11 = Code11Dlg.m_bEnable;
pCode11.bMC_CODE11_XCD = Code11Dlg.m_bXCD;
pCode11.nMC_CODE11_CDV = Code11Dlg.m_nCDV;
pCode11.nMC_CODE11_MinLen = Code11Dlg.m_nMinLen;
pCode11.nMC_CODE11_MaxLen = Code11Dlg.m_nMaxLen;

ScanCtrl.SetBarOptionCODE11(ref pCode11);
```

---

#### 4.6.2.34 ScannerControl.GetBarOptionCODE11

The GetBarOptionCODE11 function gets the option of CODE11 Barcode.

---

```
void GetBarOptionCODE11(out MCBarOption_CODE11 pCode11);
```

---

### Parameters

*MCBarOption\_CODE11*

Specifies pointer to the MCSSLibNet Structure

### Return Values

None

### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_CODE11 pCode11 = new MCBarOption_CODE11();

ScanCtrl.GetBarOptionCODE11(out pCode11);

Code11Dlg.m_bEnable = pCode11.bMC_CODE11_Enable;
Code11Dlg.m_bXCD = pCode11.bMC_CODE11_XCD;
```

---

```
Code11Dlg.m_nCDV = pCode11.nMC_CODE11_CDV;  
Code11Dlg.m_nMinLen = pCode11.nMC_CODE11_MinLen;  
Code11Dlg.m_nMaxLen = pCode11.nMC_CODE11_MaxLen;
```

#### 4.6.2.35 ScannerControl.SetBarOptionI2OF5

The SetBarOptionCODEI2OF5 function sets the option of Interleaved 2of5 Barcode.

```
void SetBarOptionCODEI2OF5(ref MCBAROption_I2OF5 pI2of5);
```

##### Parameters

*MCBarOption\_I2OF5*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

C#

```
MCBarOption_I2OF5 pI2of5 = new MCBAROption_I2OF5();  
  
pI2of5.bMC_I2OF5_Enable = m_bI2of5 = I2of5Dlg.m_bEnable;  
pI2of5.bMC_ITF14_Enable = I2of5Dlg.m_bItf14;  
pI2of5.bMC_MATRIX2OF5_Enable = I2of5Dlg.m_bMatrix2of5;  
pI2of5.bMC_DLOGIG_Enable = I2of5Dlg.m_bDlogic;  
pI2of5.bMC_INDUSTRIY_Enable = I2of5Dlg.m_bIndustry;  
pI2of5.bMC_IATA_Enable = I2of5Dlg.m_bIata;  
pI2of5.bMC_I2OF5_CDV = I2of5Dlg.m_bCDV;  
pI2of5.bMC_I2OF5_XCD = I2of5Dlg.m_bXCD;  
pI2of5.nMC_I2OF5_MinLen = I2of5Dlg.m_nMinLen;  
pI2of5.nMC_I2OF5_MaxLen = I2of5Dlg.m_nMaxLen;  
  
ScanCtrl.SetBarOptionI2OF5(ref pI2of5);
```

#### 4.6.2.36 ScannerControl.GetBarOptionI2OF5

The SetBarOptionCODEI2OF5 function gets the option of Interleaved 2of5 Barcode.

```
void GetBarOptionI2OF5(out MCBAROption_I2OF5 pI2of5);
```

##### Parameters

*MCBarOption\_I2OF5*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

C#

```
MCBarOption_I2OF5 pI2of5 = new MCBAROption_I2OF5();  
  
ScanCtrl.GetBarOptionI2OF5(out pI2of5);  
  
I2of5Dlg.m_bEnable = pI2of5.bMC_I2OF5_Enable;  
I2of5Dlg.m_bItf14 = pI2of5.bMC_ITF14_Enable;  
I2of5Dlg.m_bMatrix2of5 = pI2of5.bMC_MATRIX2OF5_Enable;  
I2of5Dlg.m_bDlogic = pI2of5.bMC_DLOGIG_Enable;
```

```
I2of5Dlg.m_bIndustry = pI2of5.bMC_INDUSTRY_Enable;
I2of5Dlg.m_bIata = pI2of5.bMC_IATA_Enable;
I2of5Dlg.m_bCDV = pI2of5.bMC_I2OF5_CDV;
I2of5Dlg.m_bXCD = pI2of5.bMC_I2OF5_XCD;
I2of5Dlg.m_nMinLen = pI2of5.nMC_I2OF5_MinLen;
I2of5Dlg.m_nMaxLen = pI2of5.nMC_I2OF5_MaxLen;
```

#### 4.6.2.37 ScannerControl.SetBarOptionCODABAR

The SetBarOptionCODABAR function sets the option of CODABAR Barcode.

---

```
void SetBarOptionCODABAR(ref MCBAROption_CODABAR pCodabar);
```

---

##### Parameters

*MCBarOption\_CODABAR*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_CODABAR pCodabar = new MCBAROption_CODABAR();

pCodabar.bMC_CODABAR_Enable = m_bCodabar = CodabarDlg.m_bEnable;
pCodabar.bMC_CODABAR_XSS = CodabarDlg.m_bXSS;
pCodabar.nMC_CODABAR_MinLen = CodabarDlg.m_nMinLen;
pCodabar.nMC_CODABAR_MaxLen = CodabarDlg.m_nMaxLen;

ScanCtrl.SetBarOptionCODABAR(ref pCodabar);
```

#### 4.6.2.38 ScannerControl.GetBarOptionCODABAR

The SetBarOptionCODABAR function gets the option of CODABAR Barcode.

---

```
void GetBarOptionCODABAR(out MCBAROption_CODABAR pCodabar);
```

---

##### Parameters

*MCBarOption\_CODABAR*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_CODABAR pCodabar = new MCBAROption_CODABAR();

ScanCtrl.GetBarOptionCODABAR(out pCodabar);

CodabarDlg.m_bEnable = pCodabar.bMC_CODABAR_Enable;
CodabarDlg.m_bXSS = pCodabar.bMC_CODABAR_XSS;
CodabarDlg.m_nMinLen = pCodabar.nMC_CODABAR_MinLen;
CodabarDlg.m_nMaxLen = pCodabar.nMC_CODABAR_MaxLen;
```

#### 4.6.2.39 ScannerControl.SetBarOptionMSI

The SetBarOptionMSI function sets the option of MSI Barcode.

---

```
void SetBarOptionMSI(ref MCBAROption_MSI pMsi);
```

---

##### Parameters

*MCBarOption\_MSI*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_MSI pMsi = new MCBAROption_MSI();

pMsi.bMC_MSI_Enable = m_bMsi = MsiDlg.m_bEnable;
pMsi.bMC_MSI_CDV = MsiDlg.m_bCDV;
pMsi.bMC_MSI_XCD = MsiDlg.m_bXCD;
pMsi.nMC_MSI_MinLen = MsiDlg.m_nMinLen;
pMsi.nMC_MSI_MaxLen = MsiDlg.m_nMaxLen;

ScanCtrl.SetBarOptionMSI(ref pMsi);
```

---

#### 4.6.2.40 ScannerControl.GetBarOptionMSI

The GetBarOptionMSI function gets the option of MSI Barcode.

---

```
void GetBarOptionMSI(out MCBAROption_MSI pMsi);
```

---

##### Parameters

*MCBarOption\_MSI*

Specifies pointer to the MCSSLibNet Structure

##### Return Values

None

##### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_MSI pMsi = new MCBAROption_MSI();

ScanCtrl.GetBarOptionMSI(out pMsi);

MsiDlg.m_bEnable = pMsi.bMC_MSI_Enable;
MsiDlg.m_bCDV = pMsi.bMC_MSI_CDV;
MsiDlg.m_bXCD = pMsi.bMC_MSI_XCD;
MsiDlg.m_nMinLen = pMsi.nMC_MSI_MinLen;
MsiDlg.m_nMaxLen = pMsi.nMC_MSI_MaxLen;
```

---

#### 4.6.2.41 ScannerControl.SetBarOptionPLESSEY

The SetBarOptionPLESSEY function sets the option of PLESSEY Barcode.

---

```
void SetBarOptionPLESSEY(ref MCBAROption_PLESSEY pPlessey);
```

---

##### Parameters

*MCBarOption\_PLESSEY*

Specifies pointer to the MCSSLibNet Structure

#### Return Values

None

#### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_PLESSEY pPlessey = new MCBarOption_PLESSEY();

pPlessey.bMC_PLESSEY_Enable = m_bPlessey = PlesseyDlg.m_bEnable;
pPlessey.bMC_PLESSEY_CDV = PlesseyDlg.m_bCDV;
pPlessey.bMC_PLESSEY_XCD = PlesseyDlg.m_bXCD;
pPlessey.nMC_PLESSEY_MinLen = PlesseyDlg.m_nMinLen;
pPlessey.nMC_PLESSEY_MaxLen = PlesseyDlg.m_nMaxLen;

ScanCtrl.SetBarOptionPLESSEY(ref pPlessey);
```

#### 4.6.2.42 ScannerControl.GetBarOptionPLESSEY

The GetBarOptionPLESSEY function gets the option of PLESSEY Barcode.

---

```
void GetBarOptionPLESSEY(out MCBarOption_PLESSEY pPlessey);
```

---

#### Parameters

*MCBarOption\_PLESSEY*

Specifies pointer to the MCSSLibNet Structure

#### Return Values

None

#### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_PLESSEY pPlessey = new MCBarOption_PLESSEY();

ScanCtrl.GetBarOptionPLESSEY(out pPlessey);

PlesseyDlg.m_bEnable = pPlessey.bMC_PLESSEY_Enable;
PlesseyDlg.m_bCDV = pPlessey.bMC_PLESSEY_CDV;
PlesseyDlg.m_bXCD = pPlessey.bMC_PLESSEY_XCD;
PlesseyDlg.m_nMinLen = pPlessey.nMC_PLESSEY_MinLen;
PlesseyDlg.m_nMaxLen = pPlessey.nMC_PLESSEY_MaxLen;
```

#### 4.6.2.43 ScannerControl.SetBarOptionGS1

The SetBarOptionGS1 function sets the option of GS1 Barcode.

---

```
void SetBarOptionGS1(ref MCBarOption_GS1 pGsl);
```

---

#### Parameters

*MCBarOption\_GS1*

Specifies pointer to the MCSSLibNet Structure

#### Return Values

None

#### Remarks

Use after successful initialization of the scanner

---

C#

```
MCBarOption_GS1 pGs1 = new MCBarOption_GS1();

pGs1.bMC_GS1_Enable = m_bGs1 = Gs1Dlg.m_bEnable;
pGs1.bMC_GS1LIM_Enable = m_bGs1Lim = Gs1Dlg.m_bGs1Lim;
pGs1.bMC_GS1EXP_Enable = m_bGs1Exp = Gs1Dlg.m_bGs1Exp;

ScanCtrl.SetBarOptionGS1(ref pGs1);
```

---

#### 4.6.2.44 ScannerControl.GetBarOptionGS1

The GetBarOptionGS1 function gets the option of GS1 Barcode.

---

```
void GetBarOptionGS1(out MCBarOption_GS1 pGs1);
```

---

##### **Parameters**

*MCBarOption\_GS1*

Specifies pointer to the MCSSLibNet Structure

##### **Return Values**

None

##### **Remarks**

Use after successful initialization of the scanner

---

C#

```
MCBarOption_GS1 pGs1 = new MCBarOption_GS1();

ScanCtrl.GetBarOptionGS1(out pGs1);

Gs1Dlg.m_bEnable = pGs1.bMC_GS1_Enable;
Gs1Dlg.m_bGs1Lim = pGs1.bMC_GS1LIM_Enable;
Gs1Dlg.m_bGs1Exp = pGs1.bMC_GS1EXP_Enable;
```

---

#### 4.6.2.45 ScannerControl.SetBarOptionTELEPEN

The SetBarOptionTELEPEN function sets the option of TELEPEN Barcode.

---

```
void SetBarOptionTELEPEN(ref MCBarOption_TELEPEN pTelepen);
```

---

##### **Parameters**

*MCBarOption\_TELEPEN*

Specifies pointer to the MCSSLibNet Structure

##### **Return Values**

None

##### **Remarks**

Use after successful initialization of the scanner

---

C#

```
MCBarOption_TELEPEN pTelepen = new MCBarOption_TELEPEN();

pTelepen.bMC_TELEPEN_Enable = TelepenDlg.m_bEnable;
pTelepen.bMC_TELEPEN_OldStyle = TelepenDlg.m_bOldStyle;

ScanCtrl.SetBarOptionTELEPEN(ref pTelepen);
```

---



#### 4.6.2.46 ScannerControl.GetBarOptionTELEPEN

The GetBarOptionTELEPEN function gets the option of TELEPEN Barcode.

---

```
void GetBarOptionTELEPEN(out MCBarOption_TELEPEN pTelepen);
```

---

##### **Parameters**

*MCBarOption\_TELEPEN*

Specifies pointer to the MCSSLibNet Structure

##### **Return Values**

None

##### **Remarks**

Use after successful initialization of the scanner

---

C#

```
MCBarOption_TELEPEN pTelepen = new MCBarOption_TELEPEN();

ScanCtrl.GetBarOptionTELEPEN(out pTelepen);

TelepenDlg.m_bEnable = pTelepen.bMC_TELEPEN_Enable;
TelepenDlg.m_bOldStyle = pTelepen.bMC_TELEPEN_OldStyle;
```

---

## 4.7 Scanner 2D (Imager)

This section provides description of the functions and DLLs which are used to manage 2D scanner module.

### Required Products

#### For C++

Required header:

M3MobileImager.h

Required lib:

M3MobileImager.lib

Required DLL:

M3MobileImager.dll

#### For C#

Required DLL:

M3MobileImager.dll  
M3MobileImagerNet.dll

### Supported Product

M3 GREEN with 2D scanner that uses software decoder

## 4.7.1 Reference and Function List for C++

### Definitions

#### #define macro

#define SYM_ENABLE	0x00000001	Enable Symbology bit
#define SYM_CHECK_ENABLE	0x00000002	Enable usage of check character.
#define SYM_CHECK_TRANSMIT	0x00000004	Send check character.
#define SYM_START_STOP_XMIT	0x00000008	Include the start and stop characters in the decoded result string.
#define SYM_ENABLE_APPEND_MODE	0x00000010	Code39 append mode.
#define SYM_ENABLE_FULLASCII	0x00000020	Enable Code39 Full ASCII.
#define SYM_NUM_SYS_TRANSMIT	0x00000040	UPC-A/UPC-E Send Num Sys
#define SYM_2_DIGIT_ADDENDA	0x00000080	Enable 2 digit Addenda (UPC and EAN).
#define SYM_5_DIGIT_ADDENDA	0x00000100	Enable 5 digit Addenda (UPC and EAN).
#define SYM_ADDENDA_REQUIRED	0x00000200	Only allow codes with addenda (UPC and EAN).
#define SYM_ADDENDA_SEPARATOR	0x00000400	Include Addenda separator space in returned string.
#define SYM_EXPANDED_UPCE	0x00000800	Extended UPC-E.
#define SYM_UPCE1_ENABLE	0x00001000	UPC-E1 enable (use SYMBOLOGY_ENABLE for UPC-E0).
#define SYM_ENABLE_MESA_IMS	0x00020000	Mesa IMS enable.
#define SYM_ENABLE_MESA_1MS	0x00040000	Mesa 1MS enable.
#define SYM_ENABLE_MESA_3MS	0x00080000	Mesa 3MS enable.
#define SYM_ENABLE_MESA_9MS	0x00100000	Mesa 9MS enable.
#define SYM_ENABLE_MESA_UMS	0x00200000	Mesa UMS enable.
#define SYM_ENABLE_MESA_EMS	0x00400000	Mesa EMS enable.
#define SYM_TELEPEN_OLD_STYLE	0x04000000	Telepen Old Style mode.:
#define SYM_COMPOSITE_UPC	0x00002000	Enable UPC Composite codes.
#define SYM_POSICODE_LIMITED_1	0x08000000	PosiCode Limited of 1
#define SYM_POSICODE_LIMITED_2	0x10000000	PosiCode Limited of 2
#define SYM_MAXICODE_CARRIERMSGONLY	0x40000000	maxicode option
#define SYM_EAN13_ISBN	0x80000000	ean13 isbn
#define MAX_TEMPLATE_LEN	256	OCR TEMPLATE length

#define MAX_GROUP_H_LEN	256	OCR Group H length
#define MAX_GROUP_G_LEN	256	OCR Group G length
#define MAX_CHECK_CHAR_LEN	64	OCR check length

## 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

## Structure

**ID\_SYM** – This enumeration is Symbology ID

```
typedef enum {
    ID_AZTEC = 0,      Aztec Code           SymFlagsRange
    ID_MESA,           Aztec Mesas          SymFlagsOnly
    ID_CODABAR,        Codabar              SymFlagsRange
    ID_CODE11,         Code 11               SymFlagsRange
    ID_CODE128,        Code 128              SymFlagsRange
    ID_CODE39,         Code 39               SymFlagsRange
    ID_CODE49,         Code 49               SymFlagsRange
    ID_CODE93,         Code 93               SymFlagsRange
    ID_COMPOSITE,      Composite Code        SymFlagsRange
    ID_DATAMATRIX,     Data Matrix           SymFlagsRange
    ID_EAN8,           EAN-8                 SymFlagsOnly
    ID_EAN13,          ENA-13                SymFlagsOnly
    ID_INT25,          Interleaved 2 of 5    SymFlagsRange
    ID_MAXICODE,       MaxiCode              SymFlagsRange
    ID_MICROPDF,       Micro PDF417          SymFlagsRange
    ID_OCR,            Ocr                   SymCodeOCR
    ID_PDF417,         PDF417                SymFlagsRange
    ID_POSTNET,        Postnet               SymFlagsOnly
    ID_QR,             QR Code               SymFlagsRange
    ID_RSS,            Reduced Space Symbology (RSS) SymFlagsRange
    ID_UPCA,           UPC-A                 SymFlagsOnly
    ID_UPCE0,          UPC-E                 SymFlagsOnly
    ID_UPCE1,          UPC-E1                SymFlagsOnly
    ID_ISBT,           ISBT                  SymFlagsOnly
    ID_BPO,            British Post          SymFlagsOnly
    ID_CANPOST,        Canadian Post         SymFlagsOnly
    ID_AUSPOST,        Australian Post       SymFlagsOnly
    ID_IATA25,         Straight 2 of 5 IATA      SymFlagsRange
    ID_CODABLOCK,      Codablock             SymFlagsRange
    ID_JAPOST,         Japanese Post         SymFlagsOnly
    ID_PLANET,         Planet Code           SymFlagsOnly
    ID_DUTCHPOST,      KIX (Netherlands) Post      SymFlagsOnly
    ID_MSI,            MSI Code              SymFlagsRange
    ID_TLCODE39,       TCIF Linked Code 39 (TLC39) SymFlagsOnly
    ID_TRIOPTIC,       Trioptic Code         SymFlagsOnly
    ID_CODE32,         Code 32               SymFlagsOnly
    ID_STRT25,         Straight 2 of 5 Industrial      SymFlagsRange
    ID_MATRIX25,       Matrix 2 of 5          SymFlagsRange
    ID_PLESSEY,        Plessey Code          SymFlagsRange
    ID_CHINAPOST,      China Post           SymFlagsRange
    ID_KOREAPOST,      Korean Post          SymFlagsRange
    ID_TELEPEN,        Telepen              SymFlagsRange
    ID_CODE16K,        Code 16k             SymFlagsRange
    ID_POSICODE,       PosiCode              SymFlagsRange
    ID_COUPONCODE,     UPC-A with Extended Coupon Code SymFlagsOnly
}
```

```

    ID_USPS4CB,      USPS 4-State Customer      SymFlagsOnly
    ID_IDTAG,        UPU 4 State ID Tag          SymFlagsOnly
    ID_GS1_128,      GS1 128                    SymFlagsRange
    ID_GEN_CODE128,  general code 128            SymFlagsRange
    ID_ALL = 100
} ID_SYM ;;

```

**SetupType** - This structure is enumerated type for specifying whether a read configuration item call should return the current settings or the imager default setting.

```

typedef enum {
    SETUP_DEFAULT = 0,
    SETUP_CURRENT
} SetupType;

```

**IMAGER\_VERSION\_INFO** - This structure is for version information.

```

typedef struct _tagVERSION_INFO{
    TCHAR    tcAPIRev[ MAX_VERSION_STRING_LEN ];
    TCHAR    tcDecoderRev[ MAX_VERSION_STRING_LEN ];
    TCHAR    tcScanDriverRev[ MAX_VERSION_STRING_LEN ];
    TCHAR    tcEtcInfo[1000];
    DWORD    dwFirmwareVersion;
    DWORD    dwFirmwareChecksum
    DWORD    dwEngineId
} IMAGER_VERSION_INFO, *PIMAGER_VERSION_INFO ;

```

**EVENT\_TYPE** - Type of data causing the event notification.

```

typedef enum {
    BARCODE_EVENT = 0,
    IMAGE_EVENT,
    TEXT_MSG_EVENT,
    INTELIMG_BARCODE_EVENT,
    INTELIMG_IMAGE_EVENT,
    TRIGGER_EVENT
} EventType_t, EVENT_TYPE, *PEVENT_TYPE;

```

**DECODE\_MSG** - Data structure that holds the decoded bar code message.

```

typedef struct _tagDECODE_MSG{
    DWORD dwStructSize;
    TCHAR pchMessage[4096];
    TCHAR chCodeID;
    TCHAR chSymLetter;
    TCHAR chSymModifier;
    DWORD nLength;
} DECODE_MSG, *PDECODE_MSG;

```

**SymFlagsOnly** - This structure for symbologies with no specified minimum or maximum length.

```

typedef struct __tagSymFlagsOnly{
    DWORD dwFlags;
}SymFlagsOnly, *PSymFlagsOnly;

```

**SymFlagsRange** - This structure for symbologies with minimum and maximum length.

```

typedef struct __tagSymFlagsRange{
    DWORD dwFlags;
    DWORD dwMinLen;
    DWORD dwMaxLen;
} SymFlagsRange, *PSymFlagsRange;

```

**SymCodeOCR** - This structure for unusual OCR.

```

typedef enum {
    OCR_MODE_DISABLED = 0,
    OCR_MODE_A,

```

```

    OCR_MODE_B,
    OCR_MODE_MONEY,
    OCR_MODE_MICR_UNSUPPORTED,
}OCRMode;

typedef enum {
    OCR_DIRECTION_LeftToRight = 0,
    OCR_DIRECTION_TopToBottom,
    OCR_DIRECTION_RightToLeft,
    OCR_DIRECTION_BottomToTop,
}OCRDirection;

typedef struct __tagSymCodeOCR{
    OCRMode ocrMode;
    OCRDirection ocrDirection;
    TCHAR tcTemplate[ MAX_TEMPLATE_LEN ];
    TCHAR tcGroupG[MAX_GROUP_G_LEN ];
    TCHAR tcGroupH[MAX_GROUP_H_LEN ];
    TCHAR tcCheckChar[ MAX_CHECK_CHAR_LEN ];
}SymCodeOCR, *PSymCodeOCR;

```

**ScanIlluminat** - Enumerated integer type that identifies possible illumination modes used during image acquisition.

```

typedef enum {
    SCAN_ILLUM_AIMER_OFF = 0,
    SCAN_ILLUM_ONLY_ON,
    SCAN_AIMER_ONLY_ON,
    SCAN_ILLUM_AIMER_ON,
} ScanIlluminat

```

#### 4.7.1.1 Connect

This function connects the engine.

---

```
BOOL Connect(  
    BOOL On  
)
```

---

##### Parameters

*On*

If this value is true, connects the engine.

If this value is false, disconnects the engine.

##### Return Value

true indicates success.

false indicates failure.

#### 4.7.1.2 ScanRead

This function causes the engine to start scanning for a decodable symbol.

---

```
BOOL ScanRead(  
    DWORD          dwTimeout  
    PDECODE_MSG    pmsg  
)
```

---

##### Parameters

*dwTimeout*

Maximum time (in seconds) to attempt to decode before declaring a no decode.

A ScannerTimeoutDelegate event specifies that the timeout is whatever is currently set on the imager.

A value of 0 indicates no timeout.

*pmsg*

Reference to a DECODE\_MSG structure to receive the data decoded.

##### Return Value

true indicates success.

false indicates failure.

#### 4.7.1.3 CancelIO

This function cancels the current bar code capture.

---

```
BOOL CancelIO( )
```

---

##### Return Value

true indicates success.

false indicates failure.

#### 4.7.1.4 AimerOn

This function turns the engine's aiming mechanism on or off.

---

```
BOOL AimerOn(  
    BOOL On  
)
```

---

##### Parameters

*On*

If this value is true, turns on the aimer.

If this value is false, turns off the aimer.

#### Return Value

true indicates success.

false indicates failure.

### 4.7.1.5 LightsOn

This function turns the engine's illumination LEDs on and off.

---

```
BOOL LightsOn(  
    BOOL On  
)
```

---

#### Parameters

*On*

If TRUE, the illumination LEDs are turned on; otherwise they're turned off.

#### Return Value

true indicates success.

false indicates failure.

### 4.7.1.6 SetScanningLightsMode

This function gives the user the ability to select what the illumination and aimer do during imaging.

---

```
BOOL SetScanningLightsMode(  
    ScanIlluminat nIllumMode  
)
```

---

#### Parameters

*ScanIlluminat*

SCAN_ILLUM_AIMER_OFF=0	Neither aimers nor illumination
SCAN_ILLUM_ONLY_ON	Illumination only
SCAN_AIMER_ONLY_ON	Aimers only
SCAN_ILLUM_AIMER_ON	Both aimers and illumination

#### Return Value

true indicates success.

false indicates failure.

### 4.7.1.7 SetScanMethods

This function registers handle or event for scanning.

---

```
BOOL SetScanMethods(  
    HANDLE          hEventHandle  
    HWND            hWnd  
    EVENTCALLBACK   EventCallback  
)
```

---

#### Parameters

*hEventHandle*

If you want to use event, register event.

*hWnd*

If you want to use window message, register window handle.

*EventCallback*

If you want to use callback function, register callback function.



You don't use parameter, parameter is null.

#### Return Value

true indicates success.  
false indicates failure.

#### 4.7.1.8 GetScanResult

This function retrieves the data from the last signal event (bar code capture).

---

```
BOOL GetScanResult(  
    EventType_t      *pEventType  
    PVOID            pResultStruct  
)
```

---

#### Parameters

*pEventType*

Reference to a EVENT\_TYPE structure to receive event type.

*pResultStruct*

Reference to a DECODE\_MSG structure to receive the data decoded.

#### Return Value

true indicates success.  
false indicates failure.

#### 4.7.1.9 DefaultSymbology

This function sets the specified symbologies to their factory default configurations.

---

```
BOOL DefaultSymbology(  
    int      nSymId  
)
```

---

#### Parameters

*nSymId*

One of the symbology enumerated types , e.g., SYM\_CODE39, SYM\_OCR, or SYM\_ALL to default all symbologies.

#### Return Value

true indicates success.  
false indicates failure.

#### 4.7.1.10 SetEnableDisableSymbology

This function sets state of the specified symbologies.

---

```
BOOL SetEnableDisableSymbology(  
    int      nSymId  
    BOOL     bEnable  
)
```

---

#### Parameters

*nSymId*

One of the symbology enumerated types , e.g., SYM\_CODE39, SYM\_OCR, or SYM\_ALL to default all symbologies.

*bEnable*

If this value is true, symbology of nSymId is enable. If this value is false, symbology of nSymId is disable.

### Return Value

true indicates success.  
false indicates failure.

#### 4.7.1.11 GetEnableDisableSymbology

This function gets state of the specified symbologies.

---

```
BOOL GetEnableDisableSymbology(  
    StupType  etType  
    int       nSymId  
    BOOL      *bEnable  
)
```

---

### Parameters

#### *SetType*

Use SETUP\_TYPE\_CURRENT for the current settings, or SETUP\_TYPE\_DEFAULT for the customer default settings.

#### *nSymId*

One of the symbology enumerated types , e.g., SYM\_CODE39, SYM\_OCR, or SYM\_ALL to default all symbologies.

#### *bEnable*

If this value is true, symbology of nSymId is enable. If this value is false, symbology of nSymId is disable.

### Return Value

true indicates success.  
false indicates failure

#### 4.7.1.12 GetInfo

This function gets version information.

---

```
BOOL GetInfo(  
    PIMAGER_VERSION_INFO info  
)
```

---

### Parameters

#### *verinfo*

reference to a IMAGER\_VERSION\_INFO structure to receive version information.

### Return Value

true indicates success.  
false indicates failure.

#### 4.7.1.13 ScanLed

This function turns the terminal's barcode scan led on or off.

---

```
BOOL ScanLed(  
    BOOL  On  
)
```

---

### Parameters

#### *On*

If On is true, turn on barcode scan led.  
If On is false, turn off barcode scan led.

### Return Value

true indicates success.  
false indicates failure.

#### 4.7.1.14 ReadSymbologyFlagsOnlyConfig

This function is used to get the symbology-specific options (symbologies with no specified minimum or maximum length).

---

```
BOOL ReadSymbologyConfig(  
    SetupType SetType  
    int nSymbol  
    PSymFlagsOnly config  
)
```

---

##### Parameters

###### *SetType*

Use SETUP\_TYPE\_CURRENT for the current settings, or SETUP\_TYPE\_DEFAULT for the customer default settings.

###### *nSymId*

One of the symbology enumerated types , e.g., SYM\_CODE39, SYM\_OCR.

###### *config*

Reference to structure with the symbology-specific options.

##### Return Value

true indicates success.  
false indicates failure.

#### 4.7.1.15 ReadSymbologyFlagsRangeConfig

This function is used to get the symbology-specific options (symbologies with minimum and maximum length).

---

```
BOOL ReadSymbologyConfig(  
    SetupType SetType  
    int nSymbol  
    PSymFlagsRange config  
)
```

---

##### Parameters

###### *SetType*

Use SETUP\_TYPE\_CURRENT for the current settings, or SETUP\_TYPE\_DEFAULT for the customer default settings.

###### *nSymId*

One of the symbology enumerated types , e.g., SYM\_CODE39, SYM\_OCR.

###### *config*

Reference to structure with the symbology-specific options.

##### Return Value

true indicates success.  
false indicates failure.

#### 4.7.1.16 ReadSymbologyOCRConfig

This function is used to get the ocr symbology-specific options.

---

```
BOOL ReadSymbologyConfig(  

```

---

---

```
    SetupType SetType  
    PSymCodeOCR config  
)
```

---

#### Parameters

##### *SetType*

Use SETUP\_TYPE\_CURRENT for the current settings, or SETUP\_TYPE\_DEFAULT for the customer default settings.

##### *config*

Reference to structure with the symbology-specific options.

#### Return Value

true indicates success.

false indicates failure.

#### 4.7.1.17 WriteSymbologyFlagsOnlyConfig

This function is used to set the symbology-specific options (symbologies with no specified minimum or maximum length).

---

```
BOOL WriteSymbologyConfig(  
    int nSymbol  
    PSymFlagsOnly config  
)
```

---

#### Parameters

##### *nSymId*

One of the symbology enumerated types , e.g., SYM\_CODE39, SYM\_OCR.

##### *config*

set to structure with the symbology-specific options.

#### Return Value

true indicates success.

false indicates failure.

#### 4.7.1.18 WriteSymbologyFlagsRangeConfig

This function is used to set the symbology-specific options (symbologies with minimum and maximum length).

---

```
BOOL WriteSymbologyConfig(  
    int nSymbol  
    SymFlagsRange config  
)
```

---

#### Parameters

##### *nSymId*

One of the symbology enumerated types , e.g., SYM\_CODE39, SYM\_OCR.

##### *config*

set to structure with the symbology-specific options.

#### Return Value

true indicates success.

false indicates failure.

#### 4.7.1.19 WriteSymbologyOCRConfig

This function is used to set the ocr symbology-specific options.

---

```
BOOL WriteSymbologyConfig(  
    SymCodeOCR config  
)
```

---

### **Parameters**

*nSymId*

One of the symbology enumerated types , e.g., SYM\_CODE39, SYM\_OCR.

*config*

set to structure with the symbology-specific options.

### **Return Value**

true indicates success.

false indicates failure.