



PT-9230 Mobile Computer SDK Programming Manual

DOC NO. UM-PT905-011

Sep. 2012

Version 3.0

©2010-2012 ARGOX Information Co., Ltd.

<http://www.argox.com>

Table of Contents

OVERVIEW.....	2
SDK FUNCTIONS.....	3
SYSAPIX.DLL.....	4
AUDIO RELATED FUNCTIONS.....	9
<i>Audio_GetVolume</i>	9
<i>Audio_SetVolume</i>	10
BATTERY RELATED FUNCTION.....	11
<i>GetBatteryStatus</i>	11
DISPLAY RELATED FUNCTIONS.....	13
<i>BacklightOn</i>	13
<i>Display_QueryBacklightIntensity</i>	14
<i>GetBacklightStatus</i>	16
<i>PowerOnLCD</i>	17
<i>SetBacklightPWM</i>	18
<i>EnableTouchPanel</i>	20
<i>GetTouchPanelStatus</i>	21
KEYPAD RELATED FUNCTIONS.....	22
<i>EnablePowerButton</i>	22
<i>GetKeypadAlphaMode</i>	23
<i>SendKbdVisualKey</i>	24
<i>SetKeypadAlphaMode</i>	25
LED RELATED FUNCTIONS.....	26
<i>GetKeypadLEDStatus</i>	26
<i>GoodReadLEDon</i>	27
<i>KeypadLEDon</i>	28
<i>QueryKeypadLEDIntensity</i>	29
<i>SetKeypadPWM</i>	31
SYSTEM RELATED FUNCTIONS.....	32
<i>CallSuspend</i>	32
<i>EnableAutoConnect</i>	33
<i>RegisterAlphaKeyNotification</i>	34
<i>ShowChineseIME</i>	35
<i>ShowDesktop</i>	36
<i>ShowExploreToolbar</i>	37
<i>ShowTaskbar</i>	38

<i>UnRegisterAlphaKeyNotification</i>	39
<i>RegisterTriggerKeyNotification</i>	40
<i>UnregisterTriggerKeyNotification</i>	41
VIBRATOR RELATED FUNCTIONS.....	42
<i>VibratorOn</i>	42
WLAN RELATED FUNCTION.....	43
<i>WL_Enable</i>	43
<i>WL_Disable</i>	44
BLUETOOTH RELATED FUNCTION.....	45
<i>BT_On</i>	45
<i>BT_Off</i>	46
<i>SetDiscoverMode</i>	47
<i>GetDiscoverMode</i>	48
<i>SetSPPService</i>	49
<i>GetSPPService</i>	50
<i>SetFTPService</i>	51
<i>GetFTPService</i>	52
<i>SetFTPWriteable</i>	53
<i>GetFTPWriteable</i>	54
<i>SetFTPShareFolder</i>	55
<i>GetFTPShareFolder</i>	56
<i>InitSearchBTDevice</i>	57
<i>FindNextBTDevice</i>	58
<i>EndSearchBTDevice</i>	60
<i>InitSearchFTPDevice</i>	61
<i>FindFirstFTPDevice</i>	62
<i>FindNextFTPDevice</i>	64
<i>PairDevice</i>	66
<i>UnPairDevice</i>	67
<i>GetComInfo</i>	68
<i>ConnectDevice</i>	70
<i>GetConnectStatus</i>	71
<i>GetSPPClientChannel</i>	72
<i>FindFirstFTPFile</i>	73
<i>FindNextFTPFile</i>	74
<i>GetFTPFile</i>	75
<i>PutFTPFile</i>	76
<i>CreateFTPFolder</i>	77
<i>DeleteFTPFolder</i>	78

<i>DeleteFTPFile</i>	79
CAMERA RELATED FUNCTION.....	80
<i>Camera_On</i>	80
<i>Camera_Off</i>	81
<i>SetPreviewSize</i>	82
<i>GetPreviewSize</i>	83
<i>EnablePreview</i>	84
<i>SetStillCaptureSize</i>	85
<i>GetStillCaptureSize</i>	86
<i>StartStillCapture</i>	87
<i>SetFlash</i>	88
<i>GetFlash</i>	89
<i>SetDarkMode</i>	90
<i>GetDarkMode</i>	91
<i>SetInvert</i>	92
<i>GetInvert</i>	93
GPRS RELATED FUNCTION.....	94
<i>GPRS_On</i>	94
<i>GPRS_Off</i>	95
<i>GetGPRSPowerStatus</i>	96
<i>SendATCommand</i>	97
<i>GetATResponse</i>	98
<i>GetIMEINumber</i>	99
<i>GetIMSI</i>	100
<i>GetModuleInfo</i>	101
<i>GetSignalQuality</i>	102
<i>ConnectRAS</i>	103
<i>DisconnectRAS</i>	104
<i>GetRASConnStatus</i>	105
<i>CreateRASEntry</i>	106
<i>DeleteRASEntry</i>	107
<i>ChangeRASEntryName</i>	108
<i>ChangeRASEntryProperty</i>	109
<i>GetRASEntryProperty</i>	110
<i>GetRASEntryCount</i>	111
<i>MenuRASEntries</i>	112
<i>GetPINCounter</i>	113
<i>GetPINStatus</i>	114
<i>SetPINLock</i>	115

<i>GetPINLockStatus</i>	116
<i>CheckPINCode</i>	117
<i>CheckPUKCode</i>	118
<i>ChangePINCode</i>	119
<i>ReadPhonebook</i>	120
<i>WritePhonebook</i>	121
<i>DeletePhonebook</i>	122
<i>ReadMultiPhonebook</i>	123
<i>GetPhonebookTotal</i>	125
<i>GetPhoneMaxLength</i>	126
<i>SendSMS</i>	127
<i>ReadSMS</i>	128
<i>DeleteSMS</i>	130
<i>ReadMultiSMS</i>	131
<i>WriteStorageSMS</i>	133
<i>SendStorageSMS</i>	134
<i>GetSMSTotal</i>	135
<i>SetSMSCentre</i>	136
<i>GetSMSCentre</i>	137
<i>SMS_Register</i>	138
<i>SMS_UnRegister</i>	139
GPS.....	140
BLUETOOTHSTRUCTURE.....	141
<i>CONNECT_INFO Structure</i>	141
<i>FTP_FILE Structure</i>	142
GPRSSTRUCTURE.....	143
<i>RAS_ENTRY Structure</i>	143
<i>PHONEBOOK_INFO Structure</i>	145
<i>SMS_INFO Structure</i>	146
RFIDDLL	147
RFIDRELATEDFUNCTION	149
<i>OpenComPort</i>	149
<i>CloseComPort</i>	150
<i>GetFWVersion</i>	151
<i>APIVersion</i>	152
<i>SetWorkingType</i>	153
<i>AntennaContro</i>	154
<i>Inventory15693</i>	155

<i>ReadSingleBlock</i>	156
<i>WriteSingleBlock</i>	157
<i>WriteAFL</i>	158
<i>WriteDSFID</i>	159
<i>LockAFL</i>	160
<i>LockDSFID</i>	161
<i>LockBlock</i>	162
<i>TagSystemInfo</i>	163
<i>TagStayQuiet</i>	164
<i>TagSelect</i>	165
<i>TagResetToReady</i>	166
<i>OpenCard1443A</i>	167
<i>ReadMifareBlock</i>	168
<i>WriteMifareBlock</i>	169
<i>ReadUltraLightBlock</i>	170
<i>WriteUltraLightBlock</i>	171
<i>GetUid1443B</i>	172
<i>SRIX4KChipID</i>	173
<i>SRIX4KReadBlock</i>	174
<i>SRIX4KWriteBlock</i>	175
RFIDREQUESTFLAGS.....	176
SRIX4KFLOWCHART.....	177
SCANAPIAXDLL	178
API_SCANRELATEDFUNCTIONS.....	183
<i>API_Register</i>	183
<i>API_Unregister</i>	184
<i>API_GetBarData</i>	185
<i>API_GetBarDataLength</i>	187
<i>API_GetBarType</i>	188
<i>API_GetError</i>	190
<i>API_GetSysError</i>	191
<i>API_GoodRead</i>	192
<i>API_LoadSettingsFromFile</i>	193
<i>API_Reset</i>	194
<i>API_ResetBarData</i>	195
<i>API_SaveSettingsToFile</i>	196
<i>API_SaveSettingsToScanner</i>	197
<i>S2K_IsLoad</i>	198

<i>S2K_Load</i>	199
<i>SCAN_QueryStatus</i>	200
<i>SCAN_SendCommand</i>	201
<i>SCAN_ResumeSystem</i>	202
<i>SCAN_BatchSetting</i>	203
<i>SCAN_BatchRead</i>	204
SCAN2KEYRELATEDFUNCTIONS.....	205
<i>PT_OpenScan2Key</i>	205
<i>PT_CloseScan2Key</i>	206
<i>PT_SetToDefault</i>	207
SCANNERRELATEDFUNCTIONS.....	208
<i>PT_EnableScanner</i>	208
<i>PT_DisableScanner</i>	209
<i>PT_CheckBarcodeData</i>	210
<i>PT_GetBarcodeData</i>	211
<i>PT_SetDefault</i>	213
SCANKEYRELATEDFUNCTIONS.....	214
<i>EnableTriggerKey</i>	214
<i>GetLibraryVersion</i>	215
<i>GetTriggerKeyStatus</i>	216
<i>PressTriggerKey</i>	217
<i>TriggerStatus</i>	218
IMAGERELATEDFUNCTIONS.....	219
<i>IMG_Open</i>	219
<i>IMG_Close</i>	220
<i>IMG_SetPictureSize</i>	221
<i>IMG_GetPictureSize</i>	222
<i>IMG_SetPictureFormat</i>	223
<i>IMG_GetPictureFormat</i>	224
<i>IMG_CapturePicture</i>	225
<i>IMG_SetInvert</i>	226
<i>IMG_GetInvert</i>	227
<i>IMG_SetRotation</i>	228
<i>IMG_GetRotation</i>	229
<i>IMG_SetProperty</i>	230
<i>IMG_GetProperty</i>	231
<i>IMG_SetRevise</i>	232
<i>IMG_GetRevise</i>	233
<i>IMG_SetQualityJPEG</i>	234

<i>IMG_GetQualityJPEG</i>	235
<i>IMG_SetFilter</i>	236
<i>IMG_GetFilter</i>	237
<i>IMG_SetTransmit</i>	238
<i>IMG_GetTransmit</i>	239
<i>IMG_SetScanTimeout</i>	240
<i>IMG_GetScanTimeout</i>	241
<i>IMG_Default</i>	242
SCANSTRUCTURE.....	243
<i>ScannerSetting Structure</i>	243
<i>GeneralSetting Structure</i>	248
<i>Code11_Setting Structure</i>	249
<i>Code39_Setting Structure</i>	250
<i>Code93_Setting Structure</i>	251
<i>Code128_Setting Structure</i>	252
<i>Codabar_Setting Structure</i>	253
<i>EAN8_Setting Structure</i>	254
<i>EAN13_Setting Structure</i>	255
<i>Industrial25_Setting Structure</i>	256
<i>Interleaved25_Setting Structure</i>	257
<i>MSI_Setting Structure</i>	258
<i>UK_Setting Structure</i>	259
<i>Telepen_Setting Structure</i>	260
<i>UPCA_Setting Structure</i>	261
<i>UPCE_Setting Structure</i>	262
<i>Matrix25_Setting Structure</i>	263
<i>PDF417_Setting Structure</i>	264
<i>Code16K_Setting Structure</i>	265
<i>ChinaPost_Setting Structure</i>	266
<i>RSS14_Setting Structure</i>	267
<i>RSSLimited_Setting Structure</i>	268
<i>RSSExpanded_Setting Structure</i>	269
<i>Pharmacode_Setting Structure</i>	270
<i>MicroPDF_Setting Structure</i>	271
<i>EANUCC_Setting Structure</i>	272
<i>IATA25_Setting Structure</i>	273
<i>Trioptic_Setting Structure</i>	274
<i>PosiCode_Setting Structure</i>	275

<i>CodablockF_Setting Structure</i>	276
<i>Code49_Setting Structure</i>	277
<i>TLC39_Setting Structure</i>	278
<i>Planet_Setting Structure</i>	279
<i>Postnet_Setting Structure</i>	280
<i>AustralianPost_Setting Structure</i>	281
<i>BritishPost_Setting Structure</i>	282
<i>CanadianPost_Setting Structure</i>	283
<i>JapanesePost_Setting Structure</i>	284
<i>KIXPost_Setting Structure</i>	285
<i>KoreaPost_Setting Structure</i>	286
<i>USPS_Setting Structure</i>	287
<i>UPU_Setting Structure</i>	288
<i>Aztec_Setting Structure</i>	289
<i>DataMatrix_Setting Structure</i>	290
<i>MaxiCode_Setting Structure</i>	291
<i>QRCode_Setting Structure</i>	292
<i>ChineseSensible_Setting Structure</i>	293
<i>OCR_Setting Structure</i>	294
SCAN COMMAND TABLE	295
FUNCTION RETURN VALUES	313

Overview

The *Argox* PT-90 Mobile Computer Software Developer Kit (SDK) Programming Manual is prepared to assist programmers on developing application programs using *Argox* PT-90 Mobile Computers under Microsoft® Windows® CE6.0 Operating System. It gives all the details needed to call functional subroutines controlling the devices on the *Argox* PT-90 Mobile Computer or access value-added devices on board such as Scanning and Wireless module.

This Programming Manual is organized as two major sections, one for the system related functions and the other for value-added scanning functions with the following information:

- Argox Mobile Computer standard Application Programming Interface (API) Definitions for system related functions:

Audio

Display

Keypad

Led and Vibrator Indicators

Battery Status

System Settings

Bluetooth

WLAN

- Argox Scanning module Application Programming Interface (API) Definitions

API definitions illustrate how to call a given functional subroutine. The API definitions are structured with information including: prototypes, parameters, return values, examples, and requirements of each API. The “Requirements” section gives information on whether or not a device supports a specific API function and the files to be included.

SDK Functions

When using SDK to develop their own application program, the programmer should link DLL file or LIB file, then, include header file SYSAPIAX.H.

The following two examples are given to show how to use LIB file and DLL file while developing an application program. We will use *Visual Studio 2005* to illustrate.

Example 1: Using LIB file.

First, programmer should include sysapiax.lib in the application project.

```
#include "Sysapiax.h"

main()
{
    .....
    SetBacklightPWM(100, 100);
    .....
}
```

Example 2: Using DLL file.

```
HINSTANCE dllHandle = NULL;

typedef DWORD (_stdcall *pfnSetBacklightPWM)(int nACPowerPercent, int
nBatteryPercent);

pfnSetBacklightPWM    m_SetBacklightPWM;

main()
{
    dllHandle = LoadLibrary(L"SYSAPIAX.dll");
    m_SetBacklightPWM = (pfnSetBacklightPWM) ::GetProcAddress(dllHandle,
_T("SetBacklightPWM"));
    m_SetBacklightPWM(0, 0);
    FreeLibrary(dllHandle);
}
```

SYSAPIAX.DLL

In PT-90 SDK, we provide SYSAPIAX.DLL which includes several functions to allow programmer to control device drivers and system functions. Programmer can use WINCE develop tool like *Visual Studio 2005* to develop application programs. Descriptions of all these functions are given below.

Audio Related Functions

- [Audio_GetVolume](#) – Query current volume setting.
- [Audio_SetVolume](#) – Set level of audio volume.

Battery Related Function

- [GetBatteryStatus](#) – Gets main battery status.

Display Related Functions

- [BacklightOn](#) – Turn ON or OFF screen backlight.
- [Display_QueryBacklightIntensity](#) – Query back-light intensity.
- [GetBacklightStatus](#) – Gets screen backlight status.
- [PowerOnLCD](#) – Turn ON or OFF the power of LCD.
- [SetBacklightPWM](#) – Adjusts screen back-light brightness.
- [EnableTouchPanel](#) – ENABLE or DISABLE touch panel.
- [GetTouchPanelStatus](#) – Get touch panel status.

KeyPad Related Functions

- [EnablePowerButton](#) – ENABLE or DISABLE Power button.
- [GetAlphaMode](#) – Get the current keypad input MODE.
- [SendKbdVisualKey](#) – Sends a virtual key to key buffer.
- [SetAlphaMode](#) – Change keypad input MODE.

LED Related Functions

- [GetKeypadLEDStatus](#) – Gets keypad backlight LED status.
- [GoodReadLEDOn](#) – Turn ON or OFF good read LED.
- [KeypadLEDOn](#) – Turn ON or OFF keypad backlight LED.
- [QueryKeypadLEDIntensity](#) – Query keypad backlight LED brightness.
- [SetKeypadPWM](#) – Adjusts keypad backlight LED brightness.

System Related Functions

- [CallSuspend](#) – Enter SUSPEND mode.
- [EnableAutoConnect](#) – Turn auto-connect ON or OFF.
- [RegisterAlphaKeyNotification](#) – Register a request to send a prompt message when the ALPHA key is pressed.
- [ShowChineseIME](#) – DISPLAY or HIDE the Chinese IME.
- [ShowDeskTop](#) – DISPLAY or HIDE all icons on desktop.
- [ShowExploreToolbar](#) – DISPLAY or HIDE toolbar on windows explorer.
- [ShowTaskbar](#) – DISPLAY or HIDE taskbar.
- [UnregisterAlphaKeyNotification](#) – UNREGISTER prompt message request for ALPHA key.
- [RegisterTriggerKeyNotification](#) – Register a request to send a prompt message when the trigger key is pressed.
- [UnregisterTriggerKeyNotification](#) – UNREGISTER prompt message request for trigger key.

Vibrator Related Functions

- [VibratorOn](#) – ON or OFF vibration indicator.

WLAN Related Functions

- [WL_Enable](#) – ENABLE WLAN.
- [WL_Disable](#) – DISABLE WLAN.

Bluetooth Related Functions

- [BT_On](#) – Enable Bluetooth.
- [BT_Off](#) – Disable Bluetooth.
- [SetDiscoverMode](#) – Enable/Disable the terminal is discoverable.
- [GetDiscoverMode](#) –Query terminal discoverable status.
- [SetSPPService](#) – Enable/Disable SPP Service.
- [GetSPPService](#) –Query SPP Service.
- [SetFTPService](#) – Enable/Disable FTP service.
- [GetFTPService](#) – Query FTP service status.
- [SetFTPWriteable](#) – Enable/Disable FTP service writeable.
- [GetFTPWriteable](#) – Query FTP service writeable status.
- [SetFTPShareFolder](#) – Setup the FTP share folder in terminal.
- [GetFTPShareFolder](#) – Query current FTP share folder in terminal.
- [InitSearchBTDevice](#) – Initial search information.

-
-
- [FindNextBTDevice](#) – retrieves the results of an Bluetooth device.
 - [EndSearchBTDevice](#) – frees the search handle.
 - [InitSearchFTPDevice](#) – Initial search the device supported FTP service.
 - [FindFirstFTPDevice](#) – Get first device supported FTP service position.
 - [FindNextFTPDevice](#) – Get next device supported FTP service position.
 - [PairDevice](#) – Pair with device.
 - [UnPairDevice](#) – Unpair with device.
 - [GetComInfo](#) – Get com identifier index and amount.
 - [ConnectDevice](#) – Connect to Bluetooth device for SPP or FTP.
 - [GetConnectStatus](#) – Query the device connected status.
 - [GetSPPClientChannel](#) – Get SPP channel.
 - [FindFirstFTPFile](#) – Get first file information from share folder in connected device.
 - [FindNextFTPFile](#) – Get next file information from share folder in connected device.
 - [GetFTPFile](#) – Get file from share folder in the connected device.
 - [PutFTPFile](#) – Send file to share folder in the connected device.
 - [CreateFTPFolder](#) – Create a new folder to share folder in the connected device.
 - [DeleteFTPFolder](#) – Delete folder from share folder in connected device.
 - [DeleteFTPFile](#) – Delete file from share folder in connected device.

Camera Related Functions

- [Camera_On](#) – Enable Camera.
- [Camera_Off](#) – Disable Camera.
- [SetPreviewSize](#) – Set preview window x-axis coordinate 、y-axis coordinate 、width 、height.
- [GetPreviewSize](#) – Get preview window x-axis coordinate 、y-axis coordinate 、width 、height.
- [EnablePreview](#) – Enable/Disable preview window.
- [SetStillCaptureSize](#) – Set still capture image pixel.
- [GetStillCaptureSize](#) – Get still capture image pixel.
- [StartStillCapture](#) – Start execute still capture active.
- [SetFlash](#) – Enable/Disable flash light.
- [GetFlash](#) – Query flash light status.
- [SetDarkMode](#) – Enable/Disable dark mode.
- [GetDarkMode](#) – Query dark mode status.
- [SetInvert](#) – Set media stream invert, include flip 、mirror.
- [GetInvert](#) – Get media stream invert, include flip 、mirror.

GPRS Related Functions

- [GPRS_On](#) – Enable GPRS.
- [GPRS_Off](#) – Disable GPRS.
- [GetGPRSPowerStatus](#) – Query GPRS current power status.
- [SendATCommand](#) – Send AT command and verify required response.
- [GetATResponse](#) – Get response from module buffer.
- [GetIMEINumber](#) – Query IMEI number.
- [GetIMSI](#) – Query IMSI number.
- [GetModuleInfo](#) – Query module information.
- [GetSignalQuality](#) – Query signal strength of GSM/GPRS network.
- [ConnectRAS](#) – Establishes a RAS connection.
- [DisconnectRAS](#) – Disconnect current RAS connection.
- [GetRASConnStatus](#) – Query RAS connection status.
- [CreateRASEntry](#) – Create a new RAS entry.
- [DeleteRASEntry](#) – Delete a RAS entry.
- [ChangeRASEntryName](#) – Change RAS entry name.
- [ChangeRASEntryProperty](#) – Change RAS entry property.
- [GetRASEntryProperty](#) – Query RAS entry property.
- [GetRASEntryCount](#) – Query RAS entry count.
- [MenuRASEntries](#) – Menu all RAS entries, and get specified entry name.
- [GetPINCounter](#) – Query still available count for entering the currently required password.
- [GetPINStatus](#) – Query current authentication code status.
- [SetPINLock](#) – Set PIN code lock or unlock.
- [GetPINLockStatus](#) – Query the PIN code lock status.
- [CheckPINCode](#) – Enter PIN code passwords, and check correctness.
- [CheckPUKCode](#) – Enter PUK code passwords, and check correctness.
- [ChangePINCode](#) – Allows defining new PIN code password.
- [ReadPhonebook](#) – Read a phonebook entry.
- [WritePhonebook](#) – Write a phonebook entry.
- [DeletePhonebook](#) – Delete a phonebook entry.
- [ReadMultiPhonebook](#) – Read multitude phonebook entries.
- [GetPhonebookTotal](#) – Query phonebook entry amount, include used 、total.
- [GetPhoneMaxLength](#) – Query maximum length of number field and text field.
- [SendSMS](#) – Send a SMS message.
- [ReadSMS](#) – Read a SMS message.
- [DeleteSMS](#) – Delete a SMS message.

-
- [ReadMultiSMS](#) – Read multitude SMS messages.
 - [WriteStorageSMS](#) – Write a SMS message to memory storage.
 - [SendStorageSMS](#) – Send a SMS message from memory storage.
 - [GetSMSTotal](#) – Query SMS message amount, include used 、total.
 - [SetSMSCentre](#) – Set phone number of SMS centre.
 - [GetSMSCentre](#) – Get phone number of SMS centre.
 - [SMS_Register](#) – Register the application to SYSAPIAX.dll.
 - [SMS_UnRegister](#) – Un-register the application from SYSAPIAX.dll.

Bluetooth Structure

- [CONNECT_INFO Structure](#) – CONNECT_INFO Information used by ConnectDevice.
- [FTP_FILE Structure](#) – FTP_FILE Information used by FindFirstFTPFile and FindNextFTPFile.

GPRS Structure

- [RAS_ENTRY Structure](#) – RAS_ENTRY Information used by CreateRASEntry 、ChangeRASEntryProperty and GetRASEntryProperty.
- [PHONEBOOK_INFO Structure](#) – PHONEBOOK_INFO Information used by ReadMultiPhonebook.
- [SMS_INFO Structure](#) – SMS_INFO Information used by ReadMultiSMS.

Audio Related Functions

Audio_GetVolume

To query the current audio volume level setting.

```
DWORD Audio_GetVolume
{
    LPDWORD lpdwVolume
}
```

Parameters

lpdwVolume

[out] The current volume level setting.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#).

Example

```
DWORD dwResult, dwVolume;
dwResult = Audio_GetVolume(&dwVolume);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Audio_GetVolume fail"));
else
{
    CString strTemp;
    strTemp.Format(_T("Volume:      %d"), dwVolume);
    AfxMessageBox(strTemp);
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

Audio_SetVolume

To set the audio volume level.

```
DWORD Audio_SetVolume
{
    DWORD dwVolume
}
```

Parameters

dwVolume

[in] Specifies a new volume level setting. The default level is 0x99999999.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#).

Example

```
DWORD dwResult,dwVolume;
dwVolume=0x11111111;
dwResult=Audio_SetVolume(dwVolume);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Audio_SetVolume fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

Battery Related Function

GetBatteryStatus

To get main battery status.

```
int GetBatteryStatus  
{  
}  
}
```

Parameters

None.

Returned Values

The return value can be one of the values in the following table.

Return value	Description
0	battery high
1	battery low
2	battery critical
3	battery charging
4	no battery
5	battery unknown

Example

```
switch (GetBatteryStatus())  
{  
case 0:  
    AfxMessageBox(_T("Battery High"));  
    break;  
case 1:  
    AfxMessageBox(_T("Battery Low"));  
    break;  
case 2:  
    AfxMessageBox(_T("Battery Critical"));  
    break;  
case 3:  
    AfxMessageBox(_T("Battery Charging"));
```

```
        break;
    case 4:
        AfxMessageBox(_T("No Battery"));
        break;
    case 5:
        AfxMessageBox(_T("Battery Unknown"));
        break;
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

Display Related Functions

BacklightOn

To turn ON or OFF the LCD screen back-light.

```
DWORD BacklightOn
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether to turn ON screen back-light(TRUE) or turn OFF screen back-light(FALSE).

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

After this action turning ON or OFF the screen back-light, the back-light will be always ON or OFF. The back-light setting of display properties in control panel does not work until the terminal been reseted.

Example

```
DWORD dwResult;
dwResult = BacklightOn(TRUE);
if(dwResult != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("BacklightOn fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

Display_QueryBacklightIntensity

To return the back-light intensity of external power and battery power:

```
DWORD Display_QueryBacklightIntensity
{
    LPDWORD lpdwACBacklight,
    LPDWORD lpdwBatteryBacklight
}
```

Parameters

lpdwACBacklight

[out] The backlight intensity of external power.

lpdwBatteryBacklight

[out] The backlight intensity of battery power.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_NULLPTR](#).

Remarks

The parameters will be one of the values in the following table.

Backlight intensity	Backlight brightness
4	super
3	normal
2	fine
1	micro
0	off

Example

```
DWORD dwResult, dwValue1, dwValue2;
dwResult = Display_QueryBacklightIntensity(&dwValue1, &dwValue2);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Display_QueryBacklightIntensity fail"));
else
{
    CString strTemp;
    strTemp.Format(_T("AC backlight intensity: %d, Battery backlight intensity: %d"), dwValue1,
```

```
dwValue2);  
    AfxMessageBox(strTemp);  
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetBacklightStatus

To get screen back-light status.

```
DWORD GetBacklightStatus  
{  
}  
}
```

Parameters

None.

Returned Values

The returned value indicates whether screen back-light is:

1 = screen back-light is ON; or

0 = screen back-light is oFF.

Example

```
DWORD dwResult;  
dwResult = GetBacklightStatus();  
if(dwResult == 1)  
    AfxMessageBox(_T("Backlight on"));  
else  
    AfxMessageBox(_T("Backlight off"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

PowerOnLCD

To turn ON or OFF the LCD screen power:

```
DWORD PowerOnLCD
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether turn on LCD power(TRUE) or turn off LCD power(FALSE).

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

After calling this function with “bOn” FALSE, terminal will only turn OFF the LCD power. It means that terminal is still working. You should either call this function again to turn ON the LCD power or to reset terminal to use the terminal with the LCD screen ON.

Example

```
DWORD dwResult;
dwResult = PowerOnLCD(FALSE); //power off LCD
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("PowerOnLCD fail"));
Sleep(3000);
dwResult = PowerOnLCD(TRUE); //power on LCD
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("PowerOnLCD fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetBacklightPWM

To adjust the LCD screen back-light brightness.

```
DWORD SetBacklightPWM
{
    int nACPowerPercent,
    int nBatteryPercent
}
```

Parameters

nACPowerPercent, *nBatteryPercent*

[in] One is the brightness level setting when the terminal is using AC power and the other is the brightness level setting when the terminal is using battery power. These two settings must be one of the values in the table below.

nPercent	Backlight brightness
100	super
75	normal
50	fine
25	micro
0	off

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

The Back-light Setting function in the Control Panel sets LCD screen back-light brightness level. Calling this function will also change the brightness level in Back-light Setting. You can use this function or Back-light Setting function in the Control Panel to adjust back-light brightness level.

Example

```
DWORD dwResult = SetBacklightPWM(100,100);
if(dwResult != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("SetBacklightPWM fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.x.lib

Link DLL: sysapi.x.dll

Device: PT90

EnableTouchPanel

To ENABLE or DISABLE the touch panel.

```
DWORD EnableTouchPanel
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to enable(TRUE) or disable(FALSE) the touch panel.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#).

Example

```
DWORD dwResult = EnableTouchPanel(TRUE);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Enable touch panel fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetTouchPanelStatus

To get touch panel status.

```
DWORD GetTouchPanelStatus  
{  
    LPBOOL lpbEnable  
}
```

Parameters

lpbEnable

[out] Receive the touch panel status. The returned value can be one of the values in the table below

Value	Touch panel status
0	Touch panel disable
1	Touch panel enable

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#).

Example

```
BOOL bEnable;  
  
DWORD dwResult = GetTouchPanelStatus(&bEnable);  
  
if(dwResult != E_FUNC_SUCCEED)  
    AfxMessageBox(_T("Get touch panel status fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

KeyPad Related Functions

EnablePowerButton

To ENABLE or DISABLE the POWER button.

```
DWORD EnablePowerButton
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether to ENABLE the POWER button(TRUE) or to DISABLE the POWER button(FALSE).

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

If the *bOn* parameter is FALSE, the POWER button will be DISABLED. The POWER button will not work when been pressed. If the terminal enters suspend mode, the POWER button will work one time only to wake up the terminal. When the terminal wakes up, the POWER button will be DISABLED again until this function been called with parameter TRUE to ENABLE the POWER button.

Example

```
DWORD dwResult;
dwResult = EnablePowerButton(FALSE);
if(dwResult != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("EnablePowerButton fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetKeypadAlphaMode

To get the current keypad INPUT mode.

DWORD *GetKeypadAlphaMode*

```
{  
}  
}
```

Parameters

None.

Returned Values

The returned value can be one of the values in the table below.

Return value	Alpha mode
0	numeric mode
1	lowercase letter mode
2	uppercase letter mode

Example

```
DWORD dwResult;  
dwResult = GetKeypadAlphaMode();  
switch (dwResult){  
case 0:  
    AfxMessageBox(_T("Numeric mode"));  
    break;  
case 1:  
    AfxMessageBox(_T("Lowercase letter mode"));  
    break;  
case 2:  
    AfxMessageBox(_T("Uppercase letter mode"));  
    break;  
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SendKbdVisualKey

To send a VIRTUAL KEY to key buffer:

```
DWORD SendKbdVisualKey
{
    BYTE Key
}
```

Parameters

Key

[in] Specifies a virtual-key code.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_PAR_ERROR](#).

Example

```
CString strTemp;
strTemp = "VisualKey";
for(int i=0; i<strTemp.GetLength(); i++)
    SendKbdVisualKey((unsigned char)strTemp.GetAt(i));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetKeypadAlphaMode

To change keypad INPUT mode.

```
DWORD SetKeypadAlphaMode  
{  
    int nMode  
}
```

Parameters

nMode

[in] Flags for setting INPUT mode. This parameter must be one of the values in the table below.

Value	Alpha mode
0	numeric mode
1	lowercase letter mode
2	uppercase letter mode

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Example

```
DWORD dwResult;  
dwResult = SetKeypadAlphaMode(1);  
if(dwResult != E_FUNC_SUCCEED)  
    AfxMessageBox(_T("SetKeypadAlphaMode fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

LED Related Functions

GetKeypadLEDStatus

To get keypad back-light LED status.

```
BOOL GetKeypadLEDStatus  
{  
}  
}
```

Parameters

None.

Returned Values

The returned value indicates whether keypad back-light LED is ON(TRUE) or OFF(FALSE).

Example

```
BOOL bResult;  
bResult = GetKeypadLEDStatus();  
if(bResult == TRUE)  
    AfxMessageBox(_T("Keypad LED on"));  
else if(bResult == FALSE)  
    AfxMessageBox(_T("Keypad LED off"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GoodReadLEDOn

To turn ON or OFF the goodread LED.

```
DWORD GoodReadLEDOn
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether to turn ON(TRUE) or OFF(FALSE) the goodread LED.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Example

```
DWORD dwResult;
dwResult = GoodReadLEDOn(TRUE);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("GoodReadLEDOn fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

KeypadLEDOn

To always turn ON or OFF the keypad LED.

```
DWORD KeypadLEDOn
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether to turn ON (TRUE) or OFF (FALSE) the keypad LED.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

The KeyPad LED Setting in Control Panel is used to set the Keypad LED operation to meet actual application requirements. Calling this function will set the KeyPad LED to always ON or OFF. Programmer can use this function or KeyPad LED Setting in the Control Panel to always turn ON or OFF the keypad LED.

Example

```
DWORD dwResult;
dwResult = KeypadLEDOn(TRUE);
if(dwResult != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("KeypadLEDOn fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

QueryKeypadLEDIntensity

To return the Keypad LED Intensity Setting when using external power and using battery power:

```
DWORD QueryKeypadLEDIntensity
{
    LPDWORD lpdwACKeypadLED,
    LPDWORD lpdwBatteryKeypadLED
}
```

Parameters

lpdwACKeypadLED

[out] The Keypad LED Intensity Setting using external power.

lpdwBatteryKeypadLED

[out] The Keypad LED Intensity Setting using battery power.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_NULLPTR](#).

Remarks

The parameters will be one of the values in the table below.

Keypad LED Intensity	Keypad LED Brightness
1	on
0	off

Example

```
DWORD dwResult, dwValue1, dwValue2;
dwResult = Display_QueryKeypadLEDIntensity(&dwValue1, &dwValue2);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("QueryKeypadLEDIntensity fail"));
else
{
    CString strTemp;
    strTemp.Format(_T("AC Keypad LED intensity: %d, Battery Keypad LED intensity: %d"), dwValue1,
dwValue2);
    AfxMessageBox(strTemp);
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetKeypadPWM

To adjust Keypad LED Brightness.

```
DWORD SetKeypadPWM
{
    int nACPowerPercent,
    int nBatteryPercent
}
```

Parameters

nACPowerPercent, nBatteryPercent

[in] One is to set Keypad LED Brightness setting when using AC power and the other is to set Keypad LED Brightness setting when using battery. These two parameters must be one of the values in the table below.

nPercent	keypad LED brightness
100	on
0	off

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

The Timeout&Brightness function in Control Panel can set Keypad LED Brightness. Calling this function will also change the Keypad LED Brightness in Timeout&Brightness function. Programmer can use either this function or Timeout&Brightness function in Control Panel to adjust the Keypad LED Brightness level.

Example

```
DWORD dwResult = SetKeypadPWM(100,100);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("SetKeypadPWM fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

System Related Functions

CallSuspend

To force the terminal entering SUSPEND mode.

```
void CallSuspend  
{  
}  
}
```

Parameters

None.

Returned Values

None.

Example

```
//suspend device  
CallSuspend();
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

EnableAutoConnect

To turn the AUTOCONNECT function ON or OFF:

```
BOOLEnableAutoConnect  
{  
    BOOLbEnable  
}
```

Parameters

bEnable

[in] Flag that indicates whether ActiveSync is being automatically executed (TRUE) or not (FALSE) when user plugging host interface cable into the terminal.

Returned Values

Returning TRUE if the operation is successful. otherwise, FALSE.

Remarks

If calling EnableAutoConnect with bEnable set to TRUE, the terminal will automatically execute ActiveSync program when user plug cable into the terminal. If calling EnableAutoConnect with bEnable set to FALSE, the terminal will not automatically execute ActiveSync program when user plug cable into the terminal.

Example

```
BOOL bResult;  
bResult = EnableAutoConnect(TRUE);  
if(bResult == FALSE)  
    AfxMessageBox(_T("EnableAutoConnect fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

RegisterAlphaKeyNotification

To Register the application to SYSAPIAX.dll, so that SYSAPIAX.dll will send a window message to the application when the Alpha Key is pressed.

```
DWORD RegisterAlphaKeyNotification
{
    HANDLE hWnd,
    UINT uMsg
}
```

Parameters

hWnd

[in] The handling window of the application to receive the message.

uMsg

[in] The message value to be sent when Alpha Key is pressed.

Returned Values

Return 0 if the operation is successful, otherwise return 1.

Remarks

The application should call *UnregisterAlphaKeyNotification* function to unregister the prompt message from the dll.

Example

```
if(RegisterAlphaKeyNotification(this->m_hWnd,WM_USER+0x0001))
    AfxMessageBox(_T("RegisterAlphaKeyNotification FAIL!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ShowChineseIME

To SHOW or HIDE ChineseIME function display.

```
BOOL ShowChineseIME  
{  
    BOOL bShow  
}
```

Parameters

bShow

[in] Flag that indicates whether to SHOW(TRUE) or HIDE (FALSE) the Chinese IME function display.

Returned Values

Returning TRUE if the operation is successful, otherwise FALSE.

Remarks

The Chinese IME is only supported in Chinese OS. It will work after calling this function then reset the terminal.

Example

```
BOOL bResult;  
bResult = ShowChineseIME(TRUE);  
if(bResult == FALSE)  
    AfxMessageBox(_T("ShowChineseIME fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ShowDesktop

To SHOW or HIDE Desktop function icon display.

```
BOOLShowDesktop  
{  
    BOOLbShow  
}
```

Parameters

bShow

[in] Flag that indicates whether to SHOW(TRUE) or HIDE(FALSE) the Desktop function icon display.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Remarks

After calling this function with parameter FALSE, the terminal will HIDE all icons on Desktop. After calling this function with parameter TRUE, the terminal will SHOW all icons on Desktop.

Example

```
BOOL bResult;  
bResult = ShowDesktop(TRUE);  
if(bResult == FALSE)  
    AfxMessageBox(_T("ShowDesktop fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ShowExploreToolbar

To SHOW or HIDE Internet Explorer toolbar function display in Windows IE.

```
BOOL ShowExploreToolbar  
{  
    BOOL bShow  
}
```

Parameters

bShow

[in] Flag that indicates whether to SHOW(TRUE) or HIDE(FALSE) the Internet Explorer toolbar in Windows IE.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Remarks

The *ShowExploreToolbar* function only affects Windows Internet Explorers been opened already.

Example

```
BOOL bResult;  
bResult = ShowExploreToolbar(TRUE);  
if(bResult == FALSE)  
    AfxMessageBox(_T("ShowExploreToolbar fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ShowTaskbar

To SHOW or HIDE Taskbar function display.

```
BOOL ShowTaskbar  
{  
    BOOL bShow  
}
```

Parameters

bShow

[in] Flag that indicates whether to SHOW(TRUE) or HIDE(FALSE) Taskbar display.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Remarks

After calling this function, the terminal will SHOW or HIDE Taskbar. If Taskbar is hidden by this function, it needs to call this function to display Taskbar again.

Example

```
BOOL bResult;  
bResult = ShowTaskbar(TRUE);  
if(bResult == FALSE)  
    AfxMessageBox(_T("ShowTaskbar fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

UnRegisterAlphaKeyNotification

To Unregister AlphaKey Notification function request so that the application will no longer receive Alpha Key been pressed notification messages.

```
DWORD UnregisterAlphaKeyNotification
{
    HANDLE hWnd
}
```

Parameters

hWnd

[in] The handling window of the application.

Returned Values

Returning 0 if the operation is successful, otherwise return 1.

Example

```
if(UnregisterAlphaKeyNotification(this->m_hWnd))
    AfxMessageBox(_T("UnregisterAlphaKeyNotification FAIL!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

RegisterTriggerKeyNotification

To Register the application to SYSAPIAX.dll, so that SYSAPIAX.dll will send a window message to the application when the trigger Key is pressed or released

```
DWORD RegisterTriggerKeyNotification
```

```
{  
    HANDLE hWnd,  
    UINT uMsg  
}
```

Parameters

hWnd

[in] The handling window of the application to receive the message.

uMsg

[in] The message value to be sent when trigger Key is pressed or released

Returned Values

Return 0 if the operation is successful, otherwise return 1.

Remarks

The application should call `UnregisterTriggerKeyNotification` function to unregister the prompt message from the dll. The `wParam` parameter of window message return the status of trigger key.

wParam	Trigger Key
0	The trigger key is released.
1	The trigger key is pressed.

Example

```
if(RegisterTriggerKeyNotification(this->m_hWnd,WM_USER+0x0001))  
    AfxMessageBox(_T("RegisterTriggerKeyNotification FAIL!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

UnregisterTriggerKeyNotification

To Unregister triggerkey Notification function request so that the application will no longer receive trigger key been pressed or released notification messages.

```
DWORD UnregisterTriggerKeyNotification  
{  
    HANDLE hWnd  
}
```

Parameters

hWnd

[in] The handling window of the application.

Returned Values

Returning 0 if the operation is successful, otherwise return 1.

Example

```
if(UnregisterTriggerKeyNotification(this->m_hWnd))  
    AfxMessageBox(_T("UnregisterTriggerKeyNotification FAIL!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

Vibrator Related Functions

VibratorOn

To turn ON or OFF the Vibration indicator.

```
DWORD VibratorOn
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether to turn ON(TRUE) or OFF(FALSE) the vibration indicator..

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

You can use this function to activate the vibration indicator of the terminal to alert the user that something is happening. Calling this function will not change the “Scanner Vibrator” setting.

Example

```
DWORD dwResult;
dwResult = VibratorOn(TRUE);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("VibratorOn fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

WLAN Related Function

WL_Enable

To ENABLE the WLAN function.

```
BOOL WL_Enable  
{  
}  
}
```

Parameters

None.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Example

```
BOOL bResult;  
bResult = WL_Enable();  
if(bResult == FALSE)  
    AfxMessageBox(_T("Wireless enable fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

WL_Disable

To DISABLE the WLAN function.

```
BOOL WL_Disable  
{  
}  
}
```

Parameters

None.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Example

```
BOOL bResult;  
bResult = WL_Disable();  
if(bResult == FALSE)  
    AfxMessageBox(_T("Wireless disable fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

Bluetooth Related Function

BT_On

To ENABLE the Bluetooth function and power.

DWORD BT_On

```
{  
}
```

Parameters

None.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [BT_ERR_CREATE_FAIL](#), [BT_ERR_INUSING](#).

Example

```
BOOL bResult;  
bResult = BT_On();  
if(bResult != E_FUNC_SUCCEED)  
    AfxMessageBox(_T("Bluetooth enable fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

BT_Off

To DISABLE the Bluetooth function and power.

```
void BT_Off  
{  
}
```

Parameters

None.

Returned Values

None

Example

```
BT_Off();
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetDiscoverMode

Enable or disable terminal discovered mode.

```
DWORD SetDiscoverMode
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to enable (TRUE) or disable (FALSE) the terminal discovered mode

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned value is [BT_ERR_SETTING_FAIL](#).

Example

```
If(SetDiscoverMode(TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox("Setting fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetDiscoverMode

Get terminal current discovered status.

```
BOOL GetDiscoverMode  
{  
}  
}
```

Parameters

None

Returned Values

Return TRUE if terminal can be discovered, otherwise return FALSE.

Example

```
if(GetDiscoverMode())  
    AfxMessageBox(_T("Discover mode is enable"));  
Else  
    AfxMessageBox(_T("Discover mode is disable"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetSPPService

Enable or disable Bluetooth serial port profile service.

```
DWORD SetSPPService  
{  
    BOOL bEnable  
}
```

Parameters

bEnable

[in] Flag that indicates whether to enable (TRUE) or disable (FALSE) the serial port profile service mode

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [BT_ERR_SETTING_FAIL](#), [BT_ERR_REG_DEV_FAIL](#), [BT_ERR_SPP_COM_FAIL](#).

Example

```
If(SetSPPService(TRUE) != E_FUNC_SUCCEED)  
    AfxMessageBox("Setting fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetSPPService

Get terminal current serial port profile service status.

```
BOOL GetSPPService
```

```
{  
}
```

Parameters

None

Returned Values

Return TRUE if SPP service is enable, otherwise return FALSE.

Example

```
if(GetSPPService())  
    AfxMessageBox(_T("SPP service is enable"));  
Else  
    AfxMessageBox(_T("SPP service is disable"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetFTPService

Enable or disable File Transfer Profile service.

```
DWORD SetFTPService
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to enable (TRUE) or disable (FALSE) the File Transfer Profile service mode

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [BT_ERR_SETTING_FAIL](#).

Example

```
if(SetFTPService(TRUE) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Set FTP service fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetFTPService

Get terminal current File Transfer Profile service status.

```
BOOL GetFTPService  
{  
}  
}
```

Parameters

None

Returned Values

Return TRUE if FTP service is enable, otherwise return FALSE.

Example

```
if(GetFTPService())  
    AfxMessageBox(_T("FTP service is enable"));  
Else  
    AfxMessageBox(_T("FTP service is disable"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetFTPWriteable

Enable or disable File Transfer Profile writable.

```
DWORD SetFTPWriteable  
{  
    BOOL bWriteable  
}
```

Parameters

bWriteable

[in] Flag that indicates whether to enable (TRUE) or disable (FALSE) the File Transfer Profile writable mode

Returned Values

Return [E_FUNC_SUCCEED](#) if the operation is successful.

Example

```
if(SetFTPWriteable(TRUE) != E_FUNC_SUCCEED)  
    AfxMessageBox(_T("Set FTP writeable fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetFTPWriteable

Get terminal current File Transfer Profile writeable status.

```
BOOL GetFTPWriteable  
{  
}  
}
```

Parameters

None

Returned Values

Return TRUE if FTP writeable is enable, otherwise return FALSE.

Example

```
if(GetFTPWriteable())  
    AfxMessageBox(_T("FTP service is writeable"));  
else  
    AfxMessageBox(_T("FTP service is diswriteable));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetFTPShareFolder

Setup the File Transfer Profile share folder.

```
DWORD SetFTPShareFolder
{
    WCHAR*strShareFolder
}
```

Parameters

strShareFolder

[in] The folder for File Transfer Profile.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned value is [E_FUNC_PAR_ERROR](#).

Example

```
if(SetFTPShareFolder("\\Temp") != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set FTP Share Folder fail!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetFTPShareFolder

Get terminal current File Transfer Profile share folder.

```
DWORD GetFTPShareFolder
{
    WCHAR *strShareFolder,
    int *nFolderLen
}
```

Parameters

strShareFolder

[out] The buffer to receive the share folder string

nFolderLen

[in/out] The **strShareFolder** buffer max size. If terminal current share folder length > nFolderLen, the nFolderLen receive current share folder length.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_PARAMETER_ERROR](#), [BT_ERR_INSUFFICIENT](#).

Remarks

If function return [BT_ERR_INSUFFICIENT](#), nFolderLen will receive the share folder length of terminal.

Example

```
WCHAR *strFolder;
int nFolderLen = 256;
strFolder = new WCHAR[nMax];
DWORD dwErr = GetFTPShareFolder(strFolder, &nFolderLen);
If(dwErr == BT\_ERR\_INSUFFICIENT){
    Delete strFolder;
    strFolder = new WCHAR[nFolderLen];
    GetFTPShareFolder(strFolder, &nFolderLen);
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

InitSearchBTDevice

This function initiates search information.

```
DWORD InitSearchBTDevice
{
    HANDLE *hLookup
}
```

Parameters

hLookup

[out] Handle to be used when calling the **FindNextBTDevice** & **EndSearchBTDevice** function

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [BT_ERR_DEVICE_ERROR](#).

Remarks

Must call **EndSearchBTDevice** function frees the handle after calls to the **InitSearchBTDevice** and **FindNextBTDevice** function.

Example

```
DWORD dwRe;
HANDLE hLookup;
ULONGLONG btAddress;
WCHAR szDeviceName[128];
dwRe = InitSearchBTDevice(&hLookup)
while(dwRe == E_FUNC_SUCCEEDED){
    dwRe = FindNextBTDevice(hLookup, szDeviceName, &btAddress, 256);
    if(dwRe == BT_ERR_DEVICE_ERROR)
        break;
    .....
}
EndSearchBTDevice(hLookup);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

FindNextBTDevice

This function retrieves the results of an nearby Bluetooth device search.

```
DWORD FindNextBTDevice
{
    HANDLE hLookup,
    LPCTSTR szDeviceName,
    ULONGLONG *btAddress,
    int nNameLen
}
```

Parameters

hLookup

[in] Handle obtained from **InitSearchBTDevice** function

szDeviceName

[out] The buffer to receive the device name string

btAddress

[out] Receive the device address of 64-bit unsigned integer

nNameLen

[in] The **szDeviceName** buffer max size. If terminal device name length > nNameLen, the

szDeviceName buffer store data of nNameLen length

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are

[E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [BT_ERR_DEVICE_ERROR](#).

Remarks

Must call **EndSearchBTDevice** function frees the handle after calls to the **InitSearchBTDevice** and

FindNextBTDevice function.

Example

```
#define GET_NAP(_bt_addr) ((USHORT)((( _bt_addr) & (ULONGLONG)0xFFFF00000000)>>(8*4)))
#define GET_SAP(_bt_addr) ((ULONG)((( _bt_addr) & (ULONGLONG)0x0000FFFFFFFF)>>(0)))

DWORD dwRe;
HANDLE hLookup;
ULONGLONG btAddress;
WCHAR szAddress[16], szDeviceName[128];

dwRe = InitSearchBTDevice(&hLookup)
while(dwRe == E_FUNC_SUCCEED){
```

```
dwRe = FindNextBTDevice(hLookup, szDeviceName, &btAddress, 256);
if(dwRe == BT_ERR_DEVICE_ERROR)
    break;
    .....
wsprintf(szAddress, L"%04X%08X", GET_NAP(btAddress), GET_SAP(btAddress));
    .....
}
EndSearchBTDevice(hLookup);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

EndSearchBTDevice

This function frees the search handle.

```
DWORD EndSearchBTDevice
{
    HANDLE hLookup
}
```

Parameters

hLookup

[in] Handle obtained from **InitSearchBTDevice** function

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [BT_ERR_DEVICE_ERROR](#).

Example

```
DWORD dwRe;
HANDLE hLookup;
ULONGLONG btAddress;
WCHAR szDeviceName[128];
dwRe = InitSearchBTDevice(&hLookup)
while(dwRe == E_FUNC_SUCCEEDED){
    dwRe = FindNextBTDevice(hLookup, szDeviceName, &btAddress, 256);
    if(dwRe == BT_ERR_DEVICE_ERROR)
        break;
    .....
}
EndSearchBTDevice(hLookup);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

InitSearchFTPDevice

Initial search Bluetooth device support File Transfer Profile service.

DWORD InitSearchFTPDevice

```
{  
}
```

Parameters

None

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [BT_ERR_DEVICE_ERROR](#).

Example

```
DWORD dwPos, dwRe;  
WCHAR szDeviceName[128];  
ULONGLONG btAddress;  
If(InitSearchFTPDevice() == E_FUNC_SUCCEEDED){  
    dwRe = FindFirstFTPDevice(&dwPos, szDeviceName, &btAddress, 256);  
    while(dwRe == E_FUNC_SUCCEEDED){  
        FindNextFTPDevice(&dwPos, szDeviceName, &btAddress, 256);  
        if(dwRe != E_FUNC_SUCCEEDED)  
            break;  
        .....  
    }  
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

FindFirstFTPDevice

Get the first searched device position information after calling `InitSearchFTPDevice()`.

```
DWORD FindFirstFTPDevice
{
    DWORD *dwPos,
    LPTSTR szDeviceName,
    ULONGLONG *btAddress,
    int nNameLen
}
```

Parameters

dwPos

[in/out] a reference to a position value returned by **FindFirstBTDevice** or **FindNextBTDevice** function

szDeviceName

[out] The buffer to receive the device name string

btAddress

[out] Receive the device address of 64-bit unsigned integer

nNameLen

[in] The **szDeviceName** buffer max size. If terminal device name length > nNameLen, the

szDeviceName buffer store data of nNameLen length

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are

[E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [BT_ERR_DEVICE_ERROR](#).

Example

```
#define GET_NAP(_bt_addr) ((USHORT)((( _bt_addr) & (ULONGLONG)0xFFFF00000000)>>(8*4)))
#define GET_SAP(_bt_addr) ((ULONG)((( _bt_addr) & (ULONGLONG)0x0000FFFFFFFF)>>(0)))

DWORD dwPos, dwRe;
WCHAR szDeviceName[128], szAddress[16];
ULONGLONG btAddress;

If(InitSearchFTPDevice() == E_FUNC_SUCCEED){
    dwRe = FindFirstFTPDevice(&dwPos, szDeviceName, &btAddress, 256);
    while(dwRe == E_FUNC_SUCCEED){
        FindNextFTPDevice(&dwPos, szDeviceName, &btAddress, 256);
        if(dwRe != E_FUNC_SUCCEED)
            break;
        .....
    }
```

```
        wprintf(szAddress, L"%04X%08X", GET_NAP(btAddress), GET_SAP(btAddress));  
        .....  
    }  
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

FindNextFTPDevice

Get the next searched device position information.

```
DWORD FindNextFTPDevice
{
    DWORD *dwPos,
    LPTSTR szDeviceName,
    ULONGLONG *btAddress,
    int nNameLen
}
```

Parameters

dwPos

[in/out] a reference to a position value returned by **FindFirstBTDevice** or **FindNextBTDevice** function

szDeviceName

[out] The buffer to receive the device name string

btAddress

[out] Receive the device address of 64-bit unsigned integer

nNameLen

[in] The **szDeviceName** buffer max size. If terminal device name length > nNameLen, the

szDeviceName buffer store data of nNameLen length

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are

[E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [BT_ERR_DEVICE_ERROR](#).

Example

```
#define GET_NAP(_bt_addr) ((USHORT)((( _bt_addr) & (ULONGLONG)0xFFFF00000000)>>(8*4))
#define GET_SAP(_bt_addr) ((ULONG)((( _bt_addr) & (ULONGLONG)0x0000FFFFFFFF)>>(0))

DWORD dwPos, dwRe;
WCHAR szDeviceName[128], szAddress[16];
ULONGLONG btAddress;

If(InitSearchFTPDevice() == E_FUNC_SUCCEED){
    dwRe = FindFirstFTPDevice(&dwPos, szDeviceName, &btAddress, 256);
    while(dwRe == E_FUNC_SUCCEED){
        FindNextFTPDevice(&dwPos, szDeviceName, &btAddress, 256);
        if(dwRe != E_FUNC_SUCCEED)
            break;
        .....
    }
```

```
        wprintf(szAddress, L"%04X%08X", GET_NAP(btAddress), GET_SAP(btAddress));  
        .....  
    }  
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

PairDevice

Pair terminal with other device.

```
DWORD PairDevice
{
    ULONGLONG btAddress
    unsigned char PinCode[16]
}
```

Parameters

btAddress

[in] The device address for pair with

PinCode

[in] The pin code for connection

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [BT_ERR_PAIR_FAIL](#), [BT_ERR_DEVICE_ERROR](#).

Example

```
if(PairDevice(btAddress, PinCode) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Pair fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

UnPairDevice

Unpair terminal with other device.

```
DWORD UnPairDevice
{
    ULONGLONG btAddress
}
```

Parameters

btAddress

[in] The device address for unpair

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned value is [BT_ERR_DEVICE_ERROR](#).

Example

```
PairDevice(btAddress, PinCode);

.....

UnPairDevice(btAddress);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetComInfo

Get com identifier index and amount from device hardware.

```
DWORD GetComInfo
{
    int *nComSum,
    LPCWSTR pComValue,
    int *nComValueLen
}
```

Parameters

nComSum

[out] Receive the device com amount

pComValue

[out] The buffer to receive the device com identifier index

nComValueLen

[in/out] The **pComValue** buffer max size. If terminal com value length > *nComValueLen*, the *nComValueLen* receive current com value length.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned value is [E_FUNC_ERROR](#), [BT_ERR_INSUFFICIENT](#).

Remarks

If function return [BT_ERR_INSUFFICIENT](#), *nComValueLen* will receive the com value length of terminal.

Example

```
WCHAR *pComValue;
int nComSum=0, nComValueLen=10;
pComValue=new WCHAR[nComValueLen];
DWORD dwErr=GetComInfo(&nComSum, pComValue, &nComValueLen);
If(dwErr==BT_ERR_INSUFFICIENT){
    Delete pComValue;
    pComValue=new WCHAR[nComValueLen];
    GetComInfo(&nComSum, pComValue, &nComValueLen);
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ConnectDevice

Connect to Bluetooth device for SPP or FTP.

```
DWORD ConnectDevice
{
    ULONGLONG btAddress,
    CONNECT_INFO *Info,
    BOOL nConnect
}
```

Parameters

btAddress

[in] The device address for connect

Info

[in] The device connect information, see [CONNECT_INFO](#) data structure.

nConnect

[in] Connect status. 1 → connect, 0 → disconnect

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [BT_ERR_CHANNEL](#), [BT_ERR_REG_DEV_FAIL](#), [BT_ERR_DEVICE_ERROR](#).

Example

```
if(ConnectDevice(btAddress, &Info, 1) == E_FUNC_SUCCEED){
    .....
}
ConnectDevice(btAddress, &Info, 0);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetConnectStatus

Query the device connect status.

```
DWORD GetConnectStatus
{
    ULONGLONG btAddress,
    int nConnectType,
    LPCWSTR pCom,
    int* nStatus
}
```

Parameters

btAddress

[in] Bluetooth device address

nConnectType

[in] Connect profile type. 1 → Serial port profile, 2 → File transfer profile

pCom

[in] The connect com for Serial port profile, must be four characters long. Contains "COM"+com identifier index, for example "COM7". If **nConnectType** parameter is 2 (FTP), **pCom** ben't to check

nStatus

[out] The device connect status

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned value is [E_FUNC_PAR_ERROR](#).

Example

```
GetConnectStatus(btAddress, 1, _T("COM7"), &nStatus);
if(nStatus)
    AfxMessageBox(_T("SPP Connect!!"));
else
    AfxMessageBox(_T("SPP Disconnect"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetSPPClientChannel

Get the device serial port profile channel.

```
DWORD GetSPPClientChannel
{
    ULONGLONG btAddress,
    int*nChannel
}
```

Parameters

btAddress

[in] The device address which to get SPP channel

nChannel

[out] Receive queried channel

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [BT_ERR_DEVICE_ERROR](#).

Example

```
if(GetSPPClientChannel(btAddress, &nChannel) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Get channel fail!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

FindFirstFTPFile

Get first file information from share folder in the connected device.

```
DWORD FindFirstFTPFile
{
    WCHAR*path,
    FTP_FILE*File
}
```

Parameters

path

[in] The path of connected device for search file.

File

[out] The first searched file information in the path, see [FTP_FILE](#) data structure.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_PAR_ERROR](#), [BT_ERR_DEVICE_NOT_CONNECT](#), [BT_ERR_FTP_DIR_FAIL](#), [BT_ERR_FTP_EMPTY_FILE](#).

Example

```
FTP_FILE File;
DWORD dwErr = FindFirstFTPFile(_T("\\"), &File);
If(dwErr == E_FUNC_SUCCEED){
    Do{
        .....
        dwErr = FindNextFTPFile(&File);
    }while(dwErr == E_FUNC_SUCCEED);
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

FindNextFTPFile

Get next file information from share folder in the connected device.

```
DWORD FindNextFTPFile
{
    FTP_FILE*File
}
```

Parameters

File

[out] The received file information, see [FTP_FILE](#) data structure.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_PAR_ERROR](#), [BT_ERR_DEVICE_NOT_CONNECT](#), [BT_ERR_FTP_EMPTY_FILE](#).

Example

```
FTP_FILE File;
DWORD dwError = FindFirstFTPFile(_T("\\"), &File);
If(dwErr == E_FUNC_SUCCEED){
    Do{
        .....
        dwErr = FindNextFTPFile(&File);
    }while(dwErr == E_FUNC_SUCCEED);
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetFTPFile

Get file from share folder in the connected device.

```
DWORD GetFTPFile
{
    LPCWSTR pTargetFile
}
```

Parameters

pTargetFile

[in] The file to get from connected device

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [BT_ERR_DEVICE_NOT_CONNECT](#).

Example

```
if(GetFTPFile(_T("\\record.txt")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get file fail!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

PutFTPFile

Send file to share folder in the connected device.

```
DWORD PutFTPFile
{
    LPCWSTR pSourceFile,
    LPCWSTR pTargetPath
}
```

Parameters

pSourceFile

[in] The source file in the share folder to transfer to connected device.

pTargetPath

[in] The target path in the connected device to save file.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [BT_ERR_DEVICE_NOT_CONNECT](#).

Example

```
if(PutFTPFile(_T("\\Temp\\record.txt"), _T("\\Collect")) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Put file fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

CreateFTPFolder

Create a new folder to share folder in the connected device.

```
DWORD CreateFTPFolder
{
    LPCWSTR pTarget
}
```

Parameters

pTarget

[in] The folder which be created to share folder in the connected device

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [BT_ERR_DEVICE_NOT_CONNECT](#).

Example

```
if(CreateFTPFolder(_T("\\FTPFolder") != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Create folder fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

DeleteFTPFolder

Delete folder from share folder in connected device.

```
DWORD DeleteFTPFolder
{
    LPCWSTR pTarget
}
```

Parameters

pTarget

[in] The folder will be deleted from share folder in the connected device

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [BT_ERR_DEVICE_NOT_CONNECT](#).

Example

```
if(!DeleteFTPFolder(_T("\\FTPFolder")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Delete folder fail!!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

DeleteFTPFile

Delete file from share folder in connected device.

```
DWORD DeleteFTPFile
{
    LPCWSTR pTarget
}
```

Parameters

pTarget

[in] The file will be deleted from share folder in the connected device.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [BT_ERR_DEVICE_NOT_CONNECT](#).

Example

```
if(!DeleteFTPFile(_T("\\FTPFolder\\record.txt")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Delete file fail!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

Camera Related Function

Camera_On

To ENABLE the Camera function and initial.

```
DWORD Camera_On
{
    HWND hVideoWnd
}
```

Parameters

hVideoWnd

[in] The handle of new owner window. This parameter can set NULL.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are

[CAM_ERR_CREATE_FAIL](#), [CAM_ERR_VERSION_ERROR](#), [CAM_ERR_GET_DEVICE_FAIL](#),
[CAM_ERR_VIDEO_WINDOW_FAIL](#), [CAM_ERR_STILL_SETTING_FAIL](#).

Example

```
if(Camera_On(NULL) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Camera on fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

Camera_Off

To DISABLE the Camera function and release.

```
void Camera_Off
{
}
```

Parameters

None.

Returned Values

None

Example

```
BT_Off();
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetPreviewSize

To set the x-axis coordinate 、y-axis coordinate 、width 、height of the preview window.

```
DWORD SetPreviewSize
{
    UINT uiPosX
    UINT uiPosY
    UINT uiWidth
    UINT uiHeight
}
```

Parameters

uiPosX

[in] Specifies the x-axis coordinate to be set. Range: 0~240.

uiPosY

[in] Specifies the y-axis coordinate to be set. Range: 0~320.

uiWidth

[in] Specifies the width to be set. Range: 0~240.

uiHeight

[in] Specifies the height to be set. Range: 0~320.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [CAM_ERR_DEVICE_ERROR](#).

Example

```
UINT uiPosX, uiPosY, uiWidth, uiHeight;
uiPosX=0;
uiPosY=0;
uiWidth=240;
uiHeight=320;
if(SetPreviewSize(uiPosX, uiPosY, uiWidth, uiHeight) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set preview size fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetPreviewSize

To get the x-axis coordinate 、y-axis coordinate 、width 、height of the preview window.

```
DWORD GetPreviewSize
{
    UINT*uiPosX
    UINT*uiPosY
    UINT*uiWidth
    UINT*uiHeight
}
```

Parameters

uiPosX

[out] Specifies the x-axis coordinate to be retrieved.

uiPosY

[out] Specifies the y-axis coordinate to be retrieved.

uiWidth

[out] Specifies the width to be retrieved.

uiHeight

[out] Specifies the height to be retrieved.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [CAM_ERR_DEVICE_ERROR](#).

Example

```
UINT uiPosX, uiPosY, uiWidth, uiHeight;
if(GetPreviewSize(&uiPosX, &uiPosY, &uiWidth, &uiHeight) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Get preview size fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

EnablePreview

Change to ENABLE or DISABLE of the preview window.

```
DWORD EnablePreview
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) preview window.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [CAM_ERR_DEVICE_ERROR](#).

Example

```
if(EnablePreview(TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Enable preview fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetStillCaptureSize

To set the image pixel for still picture capture.

```
DWORD SetStillCaptureSize
{
    DWORD dwImageSize
}
```

Parameters

dwImageSize

[in] Flags for setting image pixel. This parameterer must be one of the values in the table below.

Value	Image pixel
0	QSXGA(2560*1944)
1	SXGA(1280*960)

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [CAM_ERR_STILL_SETTING_FAIL](#), [CAM_ERR_DEVICE_ERROR](#).

Example

```
DWORD dwSize;

dwSize = 1;

if(SetStillCaptureSize(dwSize) != E_FUNC_SUCCEED)

    AfxMessageBox(_T("Set still capture size fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetStillCaptureSize

To get the image pixel setting value for still picture capture.

```
DWORD GetStillCaptureSize
{
    DWORD*dwImageSize
}
```

Parameters

dwImageSize

[out] Receive the image pixel setting value.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [CAM_ERR_DEVICE_ERROR](#).

Example

```
DWORD dwSize;
if(GetStillCaptureSize(&dwSize) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get still capture size fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

StartStillCapture

To Start with still picture capture and save to file path.

```
DWORD StartStillCapture
{
    LPTSTR lpPathName
    BOOL bSound
}
```

Parameters

lpPathName

[in] Pointer to the file save path after capture still picture. Max size of 256.

bSound

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) shutter sound.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are

[E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [CAM_ERR_NOT_SAVE_PATH](#),
[CAM_ERR_STILL_SOUND_FAIL](#), [CAM_ERR_DEVICE_ERROR](#).

Example

```
WCHAR szPictureName[256];
wcscpy(szPictureName, L'\\My Documents\\Img_01.jpg');
if(StartStillCapture(szPictureName, TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Still capture fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetFlash

To set ENABLE or DISABLE the flash light.

```
DWORD SetFlash
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) flash light.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Example

```
if(SetFlash(TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Enable flash fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetFlash

To get the flash light setting value.

```
BOOL GetFlash  
{  
}
```

Parameters

None.

Returned Values

Return TRUE if flash light is enable, otherwise return FALSE

Example

```
if(GetFlash() != TRUE)  
    AfxMessageBox(_T("Flash is disable));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetDarkMode

To set ENABLE or DISABLE the dark mode

```
DWORD SetDarkMode
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) dark mode.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Example

```
if(SetDarkMode(TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Dark mode is disable"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetDarkMode

To get the dark mode setting value.

```
BOOL GetDarkMode  
{  
}
```

Parameters

None.

Returned Values

Return TRUE if dark mode is enable, otherwise return FALSE

Example

```
if(GetDarkMode() != TRUE)  
    AfxMessageBox(_T("Dark mode is disable));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetInvert

To set the media stream invert, include flip 、mirror:

```
DWORD SetInvert
{
    BOOL bFlip
    BOOL bMirror
}
```

Parameters

bFlip

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) media stream invert of upside down.

bMirror

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) media stream invert of left to right.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Example

```
if(SetInvert(1, 1) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set invert fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetInvert

To get the media stream invert setting value, include flip 、mirror.

```
DWORD GetInvert
{
    BOOL *bFlip
    BOOL *bMirror
}
```

Parameters

bFlip

[out] Receive the media stream flip setting value.

bMirror

[out] Receive the media stream mirror setting value.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#).

Example

```
BOOL bFlip, bMirror;
if(GetInvert(&bFlip, &bMirror) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Get invert fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GPRS Related Function

GPRS_On

To ENABLE the GPRS function and initial.

```
DWORD GPRS_On  
{  
}  
}
```

Parameters

None

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [GPRS_ERR_CREATE_FAIL](#), [GPRS_ERR_VERSION_ERROR](#).

Example

```
if(GPRS_On(NULL) != E_FUNC_SUCCEED)  
    AfxMessageBox(_T("GPRS on fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GPRS_Off

To DISABLE the GPRS function and release.

```
void GPRS_Off  
{  
}  
}
```

Parameters

None.

Returned Values

None

Example

```
GPRS_Off();
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetGPRSPowerStatus

Get current power status of GPRS module.

```
BOOL GetGPRSPowerStatus  
{  
    BOOL *bStatus  
}
```

Parameters

bStatus

[out] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) GPRS power.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Example

```
BOOL bStatus;  
if(GetGPRSPowerStatus(&bStatus) != TRUE)  
    AfxMessageBox(_T("Get GPRS power status fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SendATCommand

Send specify AT command and verify required response.

```
DWORD SendATCommand
{
    char*lpCmd,
    char*lpResponse,
    DWORD dwResLen,
    DWORD dwWaitTime
}
```

Parameters

lpCmd

[in] Specifies AT command. Max size of 256.

lpResponse

[out] Receive the required response data.

dwResLen

[in] The **lpResponse** buffer max size.

dwWaitTime

[in] The time interval for wait response, in milliseconds.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are

[E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#),
[GPRS_ERR_BUFFER_INSUFFICIENT](#).

Remarks

.If receive data incomplete, maybe matter of wait time. Can adjust **dwWaitTime** parameter to try.

Example

```
char szResponse[512];
if(SendATCommand("at+gsn", szResponse, sizeof(szResponse), 0) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Send AT command fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetATResponse

Get response data from module buffer.

```
DWORD GetATResponse
{
    char*lpResponse,
    DWORD dwResLen,
    DWORD dwWaitTime
}
```

Parameters

lpResponse

[out] Receive the response data.

dwResLen

[in] The *lpResponse* buffer max size.

dwWaitTime

[in] The time interval for wait response, in milliseconds.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#), [GPRS_ERR_BUFFER_INSUFFICIENT](#).

Example

```
char szResponse[512];
if(GetATResponse(szResponse, sizeof(szResponse), 0) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get AT response fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetIMEINumber

Get module International Mobile Equipment Identity (IMEI) number.

```
DWORD GetIMEINumber
{
    LPTSTR lpIMEI
}
```

Parameters

lpResponse

[out] Receive the module IMEI number.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
WCHAR szIMEI[128];
if(GetIMEINumber(szIMEI) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Get IMEI number fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

*GetIMSI*Number

Get SIM card International Mobile Subscriber Identity (IMSI) number.

```
DWORD GetIMSINumber  
{  
    LPTSTR lpIMSI  
}
```

Parameters

lpIMSI

[out] Receive the SIM card IMSI number.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
WCHAR szIMSI[128];  
if(GetIMSINumber(szIMSI) != E_FUNC_SUCCEED)  
    AfxMessageBox(_T("Get IMSI number fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetModuleInfo

Get module identification information, include manufacture 、 module 、 revision.

```
DWORD GetATResponse
{
    LPTSTR lpManufacture,
    LPTSTR lpModule,
    LPTSTR lpRevision
}
```

Parameters

lpManufacture

[out] Receive the manufacture identification.

lpModule

[out] Receive the product model identification.

lpRevision

[out] Receive the firmware version identification.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
WCHAR szManufacture[128], szModule[128], szRevision[128];
if(GetModuleInfo(szManufacture, szModule, szRevision) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get module information fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetSignalQuality

Get signal strength from the module to the GSM/GPRS network.

```
DWORD GetSignalQuality
{
    DWORD*dwQuality
}
```

Parameters

dwQuality

[out] Receive the signal strength of GSM/GPRS network.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
DWORD dwQuality;
if(GetSignalQuality(&dwQuality) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get signal quality fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ConnectRAS

Establishes a RAS connection. Supports callback information.

```
DWORD ConnectRAS
{
    LPTSTR lpEntryName,
    HWND hNotifyWnd
}
```

Parameters

lpEntryName

[in] The entry name to use to establish the connect.

hNotifyWnd

[in] Pointer to a window handles to receive **RasDial** event notifications (for WinCE is WM_RASDIALEVENT). If **lpEntryName** is not NULL, the **RasDial** call operates asynchronously, send a message for each RAS event. If **lpEntryName** is NULL, the **RasDial** call operates synchronously, **RasDial** does not return until the connection has succeed or failed. See Microsoft documentation for WM_RASDIALEVENT message related.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(ConnectRAS(_T("GPRS Connection"), NULL) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Connect RAS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

DisconnectRAS

Disconnect current RAS connection.

```
DWORD DisconnectRAS  
{  
}  
}
```

Parameters

None

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(DisconnectRAS() != E_FUNC_SUCCEED)  
    AfxMessageBox(_T("Disconnect RAS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetRASConnStatus

Get status from the RAS connection opened with ConnectRAS.

```
DWORD GetRASConnStatus  
{  
    DWORD*dwStatus  
}
```

Parameters

dwStatus

[out] Receive the current status of the specified RAS connection.

Value	Connect Status
0	Disconnection or failed connection
1	Successfully established connection
2	Other connect status

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
DWORD dwStatus;  
if(GetRASConnStatus(&dwStatus) != E_FUNC_SUCCEEDED)  
    AfxMessageBox(_T("Get RAS connect status fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

CreateRASEntry

Create a new RAS entry.

```
DWORD CreateRASEntry
{
    LPCTSTR lpEntryName,
    RAS_ENTRY* Entry
}
```

Parameters

lpEntryName

[in] The RAS entry name of a new entry to be created.

Entry

[in] The connect entry information, see [RAS_ENTRY](#) data structure.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_ENTRY_ALREADY_EXIST](#), [GPRS_ERR_ENTRY_NOT_EXIST](#).

Example

```
RAS_ENTRY lpEntry;
lpEntry.dwCountryCode = 1;
wcscpy(lpEntry.szAreaCode, _T("425"));
.....
if(CreateRASEntry(_T("My GPRS"), &lpEntry) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Create RAS entry fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

DeleteRASEntry

Delete a RAS entry.

```
DWORD DeleteRASEntry
{
    LPTSTR lpEntryName
}
```

Parameters

lpEntryName

[in] The RAS entry name of an existing entry to be deleted.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Example

```
if(!DeleteRASEntry(_T("My GPRS")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Delete RAS entry fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ChangeRASEntryName

Change the name of a RAS entry.

```
DWORD ChangeRASEntryName
{
    LPTSTR lpOldEntry,
    LPTSTR lpNewEntry
}
```

Parameters

lpOldEntry

[in] The old RAS entry name of an existing entry.

lpNewEntry

[in] The new RAS entry name of an not exist entry.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_ENTRY_ALREADY_EXIST](#), [GPRS_ERR_ENTRY_NOT_EXIST](#).

Example

```
if(ChangeRASEntryName(_T("My GPRS"),_T("My GPRS 2")) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Change RAS entry name fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ChangeRASEntryProperty

Change the property of a RAS entry.

```
DWORD ChangeRASEntryProperty
{
    LPTSTR lpEntryName,
    RAS_ENTRY* Entry
}
```

Parameters

lpEntryName

[in] The RAS entry name of an existing entry to be change.

Entry

[in] The connect entry information, see [RAS_ENTRY](#) data structure.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_ENTRY_ALREADY_EXIST](#), [GPRS_ERR_ENTRY_NOT_EXIST](#).

Example

```
RAS_ENTRY lpEntry;
lpEntry.dwCountryCode = 2;
wcscpy(lpEntry.szAreaCode, _T("850"));
.....
if(ChangeRASEntryProperty(_T("My GPRS"), &lpEntry) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Change RAS entry property fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetRASEntryProperty

Get the property of a RAS entry.

```
DWORD GetRASEntryProperty
{
    LPTSTR lpEntryName,
    RAS_ENTRY* Entry
}
```

Parameters

lpEntryName

[in] The RAS entry name of an existing entry to be get information.

Entry

[out] Receive the connect entry information, see [RAS_ENTRY](#) data structure.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_ENTRY_NOT_EXIST](#).

Example

```
RAS_ENTRY lpEntry;
if(GetRASEntryProperty(_T("My GPRS"), &lpEntry) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get RAS entry property fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetRASEntryCount

Get entry count for RAS connection.

```
DWORD GetRASEntryCount
{
    int *nCount
}
```

Parameters

nCount

[out] Receive the RAS entry count.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#).

Example

```
int iCount;
if(GetRASEntryCount(&iCount) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Get RAS entry count fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

MenuRASEntries

Menu all existing RAS entries, and get specified entry name.

```
DWORD MenuRASEntries  
{  
    int nIndex,  
    LPTSTR lpEntryName  
}
```

Parameters

nIndex

[in] The zero-based index of a RAS entry.

lpEntryName

[out] Receive the specified RAS entry name.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Example

```
WCHAR szEntryName[20];  
if(MenuRASEntries(1, szEntryName) != E_FUNC_SUCCEED)  
    AfxMessageBox(_T("Menu RAS entries fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetPINCounter

Get count of attempts still available for entering the currently required password, for example the PIN · PUK. . .

```
DWORD GetPINCounter  
{  
    int *nCount  
}
```

Parameters

nCount

[out] Receive the number of attempts still available.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
int iCount;  
if(GetPINCounter(&iCount) != E_FUNC_SUCCEED)  
    AfxMessageBox(_T("Get PIN counter fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetPINStatus

Get the current authentication code status of SIM card

```
DWORD GetPINStatus  
{  
    DWORD*dwStatus  
}
```

Parameters

dwStatus

[out] Receive the authentication code status. The return value can be one of the values in the following table.

Value	Authentication Code Status
0	"ERROR", SIM card maybe not exist
1	"READY", PIN has already been entered
2	"SIMPIN", wait SIM PIN enter
3	"SIMPUK", wait SIM PUK enter
4	Other authentication code

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
DWORD dwStatus;  
if(GetPINStatus(&dwStatus) != E_FUNC_SUCCEEDED)  
    AfxMessageBox(_T("Get PIN status fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetPINLock

Set PIN code lock or unlock.

```
DWORD SetPINLock
{
    LPTSTR lpPIN,
    BOOL bLock
}
```

Parameters

lpPIN

[in] PIN code passwords, size of 4~8 character.

bLock

[in] Flag that indicates whether to lock (TRUE) or unlock (FALSE) the PIN code.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(SetPINLock(_T("0000"), TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set PIN lock fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetPINLockStatus

Get the PIN code lock status of SIM card

```
DWORD GetPINLockStatus
{
    BOOL *bStatus
}
```

Parameters

bStatus

[out] Receive the PIN code lock status. TRUE indicates lock is active, FALSE indicates lock is inactive.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
BOOL bStatus;
if(GetPINLockStatus(&bStatus) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get PIN lock status fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

CheckPINCode

Enter PIN code passwords, and check correctness.

```
DWORD GheckPINCode
{
    LPTSTR lpPIN,
    BOOL *bMatch
}
```

Parameters

lpPIN

[in] PIN code passwords, size of 4~8 character.

bMatch

[out] Receive the check result. TRUE indicates passwords is ok, FALSE indicates passwords is error.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
BOOL bMatch;
if(CheckPINCode(_T("0000"), &bMatch) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Check PIN code fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

CheckPUKCode

Enter PUK code passwords, and check correctness.

```
DWORD CheckPUKCode
{
    LPTSTR lpPUK,
    LPTSTR lpNewPIN,
    BOOL *bMatch
}
```

Parameters

lpPUK

[in] PUK code passwords, size of 8 character.

lpNewPIN

[in] New PIN code passwords, size of 4~8 character.

bMatch

[out] Receive the check result. TRUE indicates passwords is ok, FALSE indicates passwords is error.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
BOOL bMatch;
if(CheckPUKCode(_T("12345678"), _T("0000"), &bMatch) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Check PUK code fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ChangePINCode

Allows defining new PIN code password.

```
DWORD ChangePINCode
{
    LPTSTR lpOldPIN,
    LPTSTR lpNewPIN
}
```

Parameters

lpOldPIN

[in] Old PIN code passwords, size of 4~8 character.

lpNewPIN

[in] New PIN code passwords, size of 4~8 character.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(ChangePINCode(_T("0000"),_T("1111")) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Change PIN code fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ReadPhonebook

Read a phonebook entry in SIM card.

```
DWORD ReadPhonebook
{
    int nIndex,
    LPTSTR lpNumber,
    LPTSTR lpName
}
```

Parameters

nIndex

[in] The one-based index of a phonebook entry.

lpNumber

[out] Receive the phonebook entry number

lpName

[out] Receive the phonebook entry text.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
WCHAR szNumber[30], szName[30];
if(ReadPhonebook(1, szNumber, szName) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Read phonebook fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

WritePhonebook

Write a phonebook entry in SIM card.

```
DWORD WritePhonebook
{
    int nIndex,
    LPTSTR lpNumber,
    LPTSTR lpName,
    int nNumType
}
```

Parameters

nIndex

[in] The one-based index of a phonebook entry. If this parameter is 0, the phonebook entry will automatic write empty location.

lpNumber

[in] Phonebook entry number

lpName

[in] Phonebook entry text.

nNumType

[in] Define the used type of number:

Value	Number Type
129	Normal code
145	International access code

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(WritePhonebook(0, _T("+886912345678"), _T("test"), 145) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Write phonebook fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

DeletePhonebook

Delete a phonebook entry in SIM card.

```
DWORD DeletePhonebook
{
    int nIndex
}
```

Parameters

nIndex

[in] The one-based index of a phonebook entry.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(!DeletePhonebook(1) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Delete phonebook fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ReadMultiPhonebook

Read multitude phonebook entries in SIM card.

```
DWORD ReadMultiPhonebook
{
    int nStartIndex,
    int nEndIndex,
    DWORD *dwResLen,
    PHONEBOOK_INFO *Info,
    DWORD *dwInfoLen,
    int *nCount
}
```

Parameters within

nStartIndex

[in] The first location number to start reading, one-based index.

nEndIndex

[in] The last location number to stop reading, one-based index.

dwResLen

[in/out] The response buffer max size. If response buffer length > dwResLen, the dwResLen receive current response buffer length.

Info

[in/out] Pointer to a buffer that receives an array of [PHONEBOOK_INFO](#) structure. Before calling the function, an application must set the **dwSize** member of the first **PHONEBOOK_INFO** structure in the buffer to **sizeof(PHONEBOOK_INFO)** in order to identify the version of the structure being passed.

dwInfoLen

[in/out] Pointer to a variable that contains the size, in bytes, of the buffer specified by **Info**. On return, the function sets this variable to the number of bytes required to successfully complete the call.

nCount

[out] Receive the **Info** count.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#), [GPRS_ERR_BUFFER_INSUFFICIENT](#).

Example

```
#define LPPHONEBOOK_INFO PHONEBOOK_INFO*
.....
```

```

int iCount;
DWORD dwRet, dwInfoLen, dwResLen=10000;
PHONEBOOK_INFO PhonebookInfo={0};
LPPHONEBOOK_INFO lpPhonebookInfo=&PhonebookInfo;
dwInfoLen = sizeof(PHONEBOOK_INFO);
PhonebookInfo.dwSize = sizeof(PHONEBOOK_INFO);
if(dwRet=ReadMultiPhonebook(1, 10, &dwResLen, lpPhonebookInfo, &dwInfoLen, &iCount))
{
    if(dwRet==GPRS_ERR_BUFFER_INSUFFICIENT)
    {
        lpPhoneInfo=new PHONEBOOK_INFO(dwInfoLen/sizeof(PHONEBOOK_INFO));
        lpPhonebookInfo->dwSize=sizeof(PHONEBOOK_INFO);
        dwRet=ReadMultiPhonebook(1, 10, &dwResLen, lpPhonebookInfo, &dwInfoLen, &iCount);
        if(dwRet != E_FUNC_SUCCEEDED)
            AfxMessageBox(_T("Read multitude phonebook fail!"));
    }
    else
        AfxMessageBox(_T("Read multitude phonebook fail!"));
}

```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetPhonebookTotal

Get amount of phonebook entry in SIM card, include used + total.

```
DWORD GetPhonebookTotal
{
    int *nUsed,
    int *nTotal
}
```

Parameters

nUsed

[out] Receive the amount of used.

nTotal

[out] Receive the amount of allow maximum.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
int iUsed, iTotal;
if(GetPhonebookTotal(&iUsed, &iTotal) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get phonebook total fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetPhoneMaxLength

Get maximum length of number field and text field by SIM card.

```
DWORD GetPhoneMaxLength
{
    int *nNumberLen,
    int *nNameLen
}
```

Parameters

nNumberLen

[out] Receive the maximum length of phone number.

nNameLen

[out] Receive the maximum length of phone text.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
int iNumberLen, iNameLen;
if(GetPhoneMaxLength(&iNumberLen, &iNameLen) != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("Get phone maximum length fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SendSMS

Send SMS message to specified phone number.

```
DWORD SendSMS
{
    LPTSTR lpCenterNum,
    LPTSTR lpSendNum,
    LPTSTR lpData
}
```

Parameters

lpCenterNum

[in] The phone number of short message service center (SMSC), max size of 30. If this parameter is NULL, will use storage SMSC in terminal to send.

lpSendNum

[in] The phone number of recipient, max size of 30.

lpData

[in] SMS message data, max size of 70.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(SendSMS(_T(""),_T("0912345678"),_T("Send Test")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Send SMS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ReadSMS

Read a SMS message in SIM card.

```
DWORD ReadSMS
{
    int nIndex,
    int *nStatus,
    LPTSTR lpNumber,
    LPTSTR lpData,
    LPTSTR lpTimeStamp
}
```

Parameters

nIndex

[in] The one-based index of a SMS message.

nStatus

[out] Receive the message status. The return value can be one of the values in the following table.

Value	Message Status
0	"REC UNREAD", received unread message
1	"REC READ", received read message
2	"STO UNSENT", stored unsent message
3	"STO SENT", stored sent message

lpNumber

[out] Receive the sender phone number.

lpData

[out] Receive the SMS message data.

lpTimeStamp

[out] Receive the service center time stamp. Format: "yy/MM/dd,hh:mm:ss", where characters indicate year (two last digits), month, day, hour, minute, second.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
int iStatus;
WCHAR szNumber[30], szData[512], szTimpStamp[30];
if(ReadSMS(1, &iStatus, szNumber, szData, szTimpStamp) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Read SMS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

DeleteSMS

Delete a SMS message in SIM card.

```
DWORD DeleteSMS
{
    int nIndex
}
```

Parameters

nIndex

[in] The one-based index of a SMS message.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if>DeleteSMS(1) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Delete SMS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

ReadMultiSMS

Read multitude SMS messages in SIM card.

```
DWORD ReadMultiSMS
{
    int nStatus,
    DWORD *dwResLen,
    SMS_INFO *Info,
    DWORD *dwInfoLen,
    int *nCount
}
```

Parameters

nStatus

[in] The message status can be one of the values in the following table.

Value	Message Status
0	"REC UNREAD", received unread messages
1	"REC READ", received read messages
2	"STO UNSENT", stored unsent messages
3	"STO SENT", stored sent messages
4	"ALL", all messages

dwResLen

[in/out] The response buffer max size. If response buffer length > dwResLen, the dwResLen receive current response buffer length.

Info

[in/out] Pointer to a buffer that receives an array of [SMS_INFO](#) structure. Before calling the function, an application must set the **dwSize** member of the first **SMS_INFO** structure in the buffer to **sizeof(SMS_INFO)** in order to identify the version of the structure being passed.

dwInfoLen

[in/out] Pointer to a variable that contains the size, in bytes, of the buffer specified by **Info**. On return, the function sets this variable to the number of bytes required to successfully complete the call.

nCount

[out] Receive the **Info** count.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#), [GPRS_ERR_BUFFER_INSUFFICIENT](#).

Example

```
#define LPSMS_INFO SMS_INFO*

.....

int iCount;

DWORD dwRet, dwInfoLen, dwResLen=10000;

SMS_INFO SMSInfo={0};

LPSMS_INFO lpSMSInfo=&SMSInfo;

dwInfoLen=sizeof(SMS_INFO);

SMSInfo.dwSize=sizeof(SMS_INFO);

if(dwRet=ReadMultiSMS(4, &dwResLen, lpSMSInfo, &dwInfoLen, &iCount))
{
    if(dwRet==GPRS_ERR_BUFFER_INSUFFICIENT)
    {
        lpSMSInfo=new SMS_INFO(dwInfoLen/sizeof(SMS_INFO));
        lpSMSInfo->dwSize=sizeof(SMS_INFO);
        dwRet=ReadMultiSMS(4, &dwResLen, lpSMSInfo, &dwInfoLen, &iCount);
        if(dwRet!=E_FUNC_SUCCEED)
            AfxMessageBox(_T("Read multitude SMS fail!"));
    }
    else
        AfxMessageBox(_T("Read multitude SMS fail!"));
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

WriteStorageSMS

Write a SMS message to SIM memory storage.

```
DWORD WriteStorageSMS
{
    int *nIndex,
    LPTSTR lpCenterNum,
    LPTSTR lpSendNum,
    LPTSTR lpData
}
```

Parameters

nIndex

[out] Receive the one-based index of storage SMS message.

lpCenterNum

[in] The phone number of short message service center (SMSC) , max size of 30. If this parameter is NULL, will use storage SMSC in terminal to send.

lpSendNum

[in] The phone number of recipient, max size of 30.

lpData

[in] SMS message data, max size of 70.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
int iIndex;
if(WriteStorageSMS(&iIndex, _T(""), _T("0912345678"), _T("Write Storage Test")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Write storage SMS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SendStorageSMS

Send a SMS message from SIM memory storage.

```
DWORD SendStorageSMS
{
    int nIndex
}
```

Parameters

nIndex

[in] The one-based index of storage SMS message.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(SendStorageSMS(1) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Send storage SMS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetSMSTotal

Get amount of SMS message in SIM card, include used \ total.

```
DWORD GetSMSTotal
{
    int *nUsed,
    int *nTotal
}
```

Parameters

nUsed

[out] Receive the amount of used.

nTotal

[out] Receive the amount of allow maximum.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
int iUsed, iTotal;
if(GetSMSTotal (&iUsed, &iTotal) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get SMS total fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SetSMSCentre

Set phone number of short message service center (SMSC).

```
DWORD SetSMSCentre
{
    LPTSTR lpNumber
}
```

Parameters

lpNumber

[in] SMS centre phone number.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(SetSMSCentre(_T("+886912345678")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set SMS centre fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

GetSMSCentre

Get phone number of short message service center (SMSC).

```
DWORD GetSMSCentre
{
    LPTSTR lpNumber,
    int nMaxLen
}
```

Parameters

lpNumber

[out] Receive the SMS centre phone number.

nMaxLen

[in] The **lpNumber** buffer max size.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
WCHAR szNumber[30];
if(GetSMSCentre(szNumber, 30) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get SMS centre fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT90

SMS_Register

To register the SMS application to SYSAPIAX.dll so that SYSAPIAX.dll can communicate with the application.

```
DWORD SMS_Register
{
    HWND hNotifyWnd,
    UINT uNewSMSNotify,
    UINT uNewPhoneNotify
}
```

Parameters

hNotifyWnd

[in] The window handling the function library will send message to report.

uNewSMSNotify

[in] Application-defined message identifier. The system uses this for notification message that it sends to the window identified in **hNotifyWnd**. These notifications are sent when receive a new SMS of GPRS module.

uNewPhoneNotify

[in] Application-defined message identifier. The system uses this for notification message that it sends to the window identified in **hNotifyWnd**. These notifications are sent when receive a new phone call of GPRS module.

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(SMS_Register(hWnd, WM_APP+1, WM_APP+2) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("SMS_Register fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SMS_UnRegister

To unregister the SMS application from SYSAPIAX.dll.

```
DWORD SMS_UnRegister  
{  
}  
}
```

Parameters

None

Returned Values

If the action succeeds, the returned value is [E_FUNC_SUCCEED](#). If the action fails, possible returned values are [E_FUNC_ERROR](#), [GPRS_ERR_DEVICE_ERROR](#).

Example

```
if(SMS_UnRegister() != E_FUNC_SUCCEED)  
    AfxMessageBox(_T("SMS_UnRegister fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GPS

The GPS receiver on the PT90 uses COM8 to output NMEA messages, when the COM port is opened. Using Microsoft Standard API to open serial port and receive NMEA data. For details, please refer to NMEA0183 ver3.0.

Serial Configuration	GPS receiver
Com port	8
Baud rate	9600
Data bits	8
Parity	None
Stop bits	1

Requirements

OS Versions: Windows CE 6.0 or beyond.

Device: PT90

Bluetooth Structure

CONNECT_INFO Structure

This setting file contains information used by [ConnectDevice](#).

```
Struct CONNECT_INFO
{
    int nChannel;
    int nConnectType;
    WCHAR strCom[6];
}
```

Members

nChannel

The connect channel for Serial port profile

nConnectType

Connect profile type. 1 → Serial port profile, 2 → File transfer profile

strCom

The connect com for Serial port profile, must be four characters long.. Contains "COM"+com identifier index, for example "COM7". If **nConnectType** member is 2 (FTP), **strCom** ben't to check

Structure Information

Header: sysapi.h

Device: PT90

FTP_FILE Structure

This setting file contains information used by [FindFirstFTPFile](#), [FindNextFTPFile](#).

```
Struct FTP_FILE
{
    int nFileType;
    WCHAR strPath[260];
    WCHAR strFile[260];
    DWORD dwFileSize;
}
```

Members

nFileType

File object profile type. 0 → File, 1 → Folder

strPath

The file path

strFile

The file name; It will be null if the object is a folder

dwFileSize

The file size , in bytes; It will be 0 if the object is a folder.

Structure Information

Header: sysapi.h

Device: PT90

GPRS Structure

RAS_ENTRY Structure

This setting file contains information used by [CreateRASEntry](#), [ChangeRASEntryProperty](#), [GetRASEntryProperty](#).

```
Struct CONNECT_INFO
{
    DWORD dwCountryCode;
    WCHAR szAreaCode[10];
    WCHAR szPhoneNumber[128];
    WCHAR szExtraCmd[330];
    WCHAR szUserName[256];
    WCHAR szPassword[256];
    WCHAR szDomain[15];
}
```

Members

dwCountryCode

Specifies the country/region code portion of the phone number.

szAreaCode

Null-terminated string that contains the area code.

szPhoneNumber

The phone number of RAS entry.

szExtraCmd

Specifies the special modem commands may be inserted into the dial string.

szUserName

Null-terminated string that contains the user's name. This string is used to authenticate the user's access to the remote access server.

szPassword

Null-terminated string that contains the user's password. This string is used to authenticate the user's access to the remote access server.

szDomain

Null-terminated string that contains the domain on which authenticate is to occur. An empty string ("") specifies the domain in which the remote access server is a member. An asterisk specifies the domain stored in the RAS for the entry.

Structure Information

Header: sysapi.h

Device: PT90

PHONEBOOK_INFO Structure

This setting file contains information used by [ReadMultiPhonebook](#).

```
Struct PHONEBOOK_INFO
{
    DWORD dwSize;
    int nIndex;
    WCHAR szNumber[30];
    WCHAR szName[30];
}
```

Members

dwSize

Specifies the structure size, in bytes.

nIndex

The one-based index of a phonebook entry.

szNumber

Phonebook entry number.

szName

Phonebook entry text.

Structure Information

Header: sysapi.h

Device: PT90

SMS_INFO Structure

This setting file contains information used by [ReadMultiSMS](#).

```
Struct SMS_INFO
{
    DWORD dwSize;
    int nIndex;
    int nStatus;
    WCHAR szNumber[30];
    WCHAR szData[160];
    WCHAR szTimeStamp[30];
}
```

Members

dwSize

Specifies the structure size, in bytes.

nIndex

The one-based index of a SMS message.

nStatus

Message status. The return value can be one of the values in the following table.

Value	Message Status
0	"REC UNREAD", received unread message
1	"REC READ", received read message
2	"STO UNSENT", stored unsent message
3	"STO SENT", stored sent message

szNumber

Sender phone number.

szData

SMS message data.

szTimeStamp

Service center time stamp. Format: "yy/MM/dd,hh:mm:ss", where characters indicate year (two last digits), month, day, hour, minute, second.

Structure Information

Header: sysapi.h

Device: PT90

RFID.DLL

We supply RFID.DLL which includes several functions to allow programmer to control the on-board RFID device. Programmer can use WINCE develop tool like *Visual Studio 2005* to develop application programs. Descriptions of all these functions are given below.

RFID Related Functions

- [OpenComPort](#) – Open com port to connect RFID module.
- [CloseComPort](#) – Close com port and disconnect RFID module.
- [GetFWVersion](#) – Query RFID module firmware version.
- [APIVersion](#) – Query RFID API version.
- [SetWorkingType](#) – Set RFID module working type with different tag type.
- [AntennaContro](#) – Enable/Disable antenna to save power.
- [Inventory15693](#) – Query 15693 tag unique identifier (UID).
- [ReadSingleBlock](#) – Read data of single block from 15693 tag.
- [WriteSingleBlock](#) – Write data of single block to 15693 tag.
- [WriteAFI](#) – Write application family identifier (AFI) to 15693 tag.
- [WriteDSFID](#) – Write data structure format identifier (DSFID) to 15693 tag.
- [LockAFI](#) – Lock AFI on 15693 tag.
- [LockDSFID](#) – Lock DSFID on 15693 tag.
- [LockBlock](#) – Lock single block on 15693 tag.
- [TagSystemInfo](#) – Query tag information from 15693 tag.
- [TagStayQuiet](#) – Set 15693 tag to stay quiet state.
- [TagSelect](#) – Set 15693 tag to select state.
- [TagResetToReady](#) – Set select or quiet tag to ready state.
- [OpenCard14443A](#) – Open 14443A tag of mifare type and get tag UID.
- [ReadMifareBlock](#) – Read data of single block from 14443A mifare S50/S70 tag.
- [WriteMifareBlock](#) – Write data of single block to 14443A mifare S50/S70 tag.
- [ReadUltraLightBlock](#) – Read data of single block from 14443A mifare ultralight tag.
- [WriteUltraLightBlock](#) – Write data of single block to 14443A mifare ultralight tag.
- [GetUid14443B](#) – Query 14443B tag UID.
- [SRIX4KChipID](#) – Query 14443B SRIX4K tag chip ID.
- [SRIX4KReadBlock](#) – Read data of single block from 14443B SRIX4K tag.
- [SRIX4KWriteBlock](#) – Write data of single block to 14443B SRIX4K tag.

When user wants to use this library, user should link RFID.DLL, RFID.LIB and the relate functions header file (RFID.H).

RFID Related Function

OpenComPort

Open com port to connect the RFID module.

```
int OpenComPort
{
    int iCom
}
```

Parameters

iCom

[in] The RFID module com port number: PT90 only specified com 6.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(OpenComPort(6) !=0)
    AfxMessageBox(_T("RFID open com fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

CloseComPort

Close com port and disconnect the RFID module connection.

```
int CloseComPort  
{  
}
```

Parameters

None.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(CloseComPort() !=0)  
    AfxMessageBox(_T("RFID close com fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

GetFWVersion

Get RFID module firmware version.

```
int GetFWVersion
{
    char lpOutVersion[]
}
```

Parameters

lpOutVersion

[out] Receive the RFID module firmware version.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cVersion[20];
if(GetFWVersion(cVersion) != 0)
    AfxMessageBox(_T("Get FW version fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

APIVersion

Get RFID API version.

```
int APIVersion
{
    char lpOutVersion[]
}
```

Parameters

lpOutVersion

[out] Receive the RFID API version.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cVersion[20];
if(APIVersion(cVersion) != 0)
    AfxMessageBox(_T("Get API version fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

SetWorkingType

Set RFID module working type with different tag type, and this function should be called before read card.

```
int SetWorkingType
{
    int iType,
    int iHalfPower
}
```

Parameters

iType

[in] The RFID module working type. This parameter must be one of the values in the table below.

Value	Working type
1	15693
2	14443A
3	14443B

iHalfPower

[in] The RFID module power type. This parameter must be one of the values in the table below.

Value	Power type
0	full power
1	Half power

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(SetWorkingType(1,0) != 0)
    AfxMessageBox(_T("Set working type fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

AntennaContro

Enable/Disable antenna to save power.

```
int AntennaContro
{
    int iSelect
}
```

Parameters

iSelect

[in] Flag that indicates whether to Enable(1) or Disable(0) the antenna.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(AntennaContro(1) !=0)
    AfxMessageBox(_T("Antenna control fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

Inventory15693

Get ISO 15693 tag unique identifier (UID).

```
int Inventory15693
{
    char lpFlag[],
    char lpAFI[],
    char lpOutUID[]
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	0	Slots_flag 0 - 16 slots 1 - 1 slots	0	0	1	0	0

lpAFI

[in] This parameter not support and should be set to "".

lpOutUID

[out] Receive the tag unique identifier.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cUID[40];
if (Inventory15693("04", "", cUID) != 0)
    AfxMessageBox(_T("Inventory 15693 fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

ReadSingleBlock

Read data of single block from ISO 15693 tag.

```
int ReadSingleBlock
{
    int iBlock,
    char lpFlag[],
    char lpAddressUID[],
    char lpOutData[]
}
```

Parameters

iBlock

[in] The block index of ISO 15693 tag. Please refer to tag respective document.

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

lpOutData

[out] Receive the block data.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cData[128];
if(ReadSingleBlock(8, "00", "", cData) != 0)
    AfxMessageBox(_T("Read single block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

WriteSingleBlock

Write data of single block to ISO 15693 tag.

```
int WriteSingleBlock
{
    int iBlock,
    char lpFlag[],
    char lpAddressUID[],
    char lpData[]
}
```

Parameters

iBlock

[in] The block index of ISO 15693 tag. Please refer to tag respective document.

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

lpData

[in] The block data. Data lengths refer to tag respective document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(WriteSingleBlock(8, "40", "", "12345678") != 0)
    AfxMessageBox(_T("Write single block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

WriteAFI

Write application family identifier (AFI) to ISO 15693 tag.

```
int WriteAFI
{
    char lpFlag[],
    char lpAddressUID[],
    char lpData[]
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

lpData

[in] The AFI data, 2 byte.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(WriteAFI("40", "", "11") != 0)
    AfxMessageBox(_T("Write AFI fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

WriteDSFID

Write data structure format identifier (DSFID) to ISO 15693 tag.

```
int WriteDSFID
{
    char lpFlag[],
    char lpAddressUID[],
    char lpData[]
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

lpData

[in] The DSFID data, 2 byte.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(WriteDSFID("40", "", "22") != 0)
    AfxMessageBox(_T("Write DSFID fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

LockAFI

Lock application family identifier (AFI) on ISO 15693 tag.

```
int LockAFI
{
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When applies this function, AFI will lock and be unable to restore on ISO 15693 tag.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(LockAFI("20", cUID) != 0)
    AfxMessageBox(_T("Lock AFI fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

LockDSFID

Lock data structure format identifier (DSFID) on ISO 15693 tag.

```
int LockDSFID
{
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When applies this function, DSFID will lock and be unable to restore on ISO 15693 tag.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(LockDSFID("20", cUID) != 0)
    AfxMessageBox(_T("Lock DSFID fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

LockBlock

Lock specific block on ISO 15693 tag.

```
int LockBlock
{
    int iBlock,
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

iBlock

[in] The block index of ISO 15693 tag. Please refer to tag respective document.

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When applies this function, the block will lock and be unable to restore on ISO 15693 tag.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(LockBlock(8, "20", cUID) != 0)
    AfxMessageBox(_T("Lock block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

TagSystemInfo

Get tag information from ISO 15693 tag.

```
int TagSystemInfo
{
    char lpFlag[],
    char lpAddressUID[],
    char lpOutData[]
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

lpOutData

[out] Receive the tag information.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cData[128];
if(TagSystemInfo("00", "", cData) != 0)
    AfxMessageBox(_T("Get tag information fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

TagStayQuiet

Set ISO 15693 tag to stay quiet state.

```
int TagStayQuiet
{
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	0	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(TagStayQuiet("20", cUID) != 0)
    AfxMessageBox(_T("Set stay quiet state fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

TagSelect

Set ISO 15693 tag to select state.

```
int TagSelect
{
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	0	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(TagSelect("20", cUID) != 0)
    AfxMessageBox(_T("Set select state fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

TagResetToReady

Set select or quiet tag to ready state.

```
int TagResetToReady
{
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and [RFID Request Flags](#). This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	0	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(TagResetToReady("20", cUID) != 0)
    AfxMessageBox(_T("Set ready state fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

OpenCard1443A

Open ISO 14443A tag of mifare type and get the tag UID.

```
int OpenCard1443A
{
    char lpOutUID[],
    char lpOutType[]
}
```

Parameters

lpOutUID

[out] Receive the tag unique identifier.

lpOutData

[out] Receive the tag type. Please refer to mifare based document For example common values in the following table.

Value	Tag type
0400	Mifare Classic 1k
0200	Mifare Classic 4k
4400	Mifare Ultralight

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cUID[40], cType[20];
if(OpenCard1443A(cUID, cType) != 0)
    AfxMessageBox(_T("Open card 14443A fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

ReadMifareBlock

Read data of single block from ISO 14443A mifare S50/S70 tag.

```
int ReadMifareBlock
{
    int iKeyType,
    char lpKey[],
    int iBlock,
    char lpOutData[]
}
```

Parameters

iKeyType

[in] The key type. This parameter must be one of the values in the table below.

Value	Key type
0	Key A
1	Key B

lpKey

[in] The key string, 12 byte.

iBlock

[in] The block index of mifare S50/S70 tag. Please refer to tag respective document.

lpOutData

[out] Receive the block data.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cData[128];
if(ReadMifareBlock(0, "FFFFFFFFFFFF", 8, cData) != 0)
    AfxMessageBox(_T("Read mifare block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

WriteMifareBlock

Write data of single block to ISO 14443A mifare S50/S70 tag.

```
int WriteMifareBlock
{
    int iKeyType,
    char lpKey[],
    int iBlock,
    char lpData[]
}
```

Parameters

iKeyType

[in] The key type. This parameter must be one of the values in the table below.

Value	Key type
0	Key A
1	Key B

lpKey

[in] The key string, 12 byte.

iBlock

[in] The block index of mifare S50/S70 tag. Please refer to tag respective document.

lpData

[in] The block data, 32byte.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(WriteMifareBlock(0, "FFFFFFFFFFFF", 8, "11111111112222222222333333333344") != 0)
    AfxMessageBox(_T("Write mifare block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

ReadUltraLightBlock

Read data of single block from ISO 14443A mifare ultralight tag.

```
int ReadUltraLightBlock
{
    int iBlock,
    char lpOutData[]
}
```

Parameters

iBlock

[in] The block index of mifare ultralight tag. Please refer to tag respective document.

lpOutData

[out] Receive the block data.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cData[128];
if(ReadUltraLightBlock(8, cData) != 0)
    AfxMessageBox(_T("Read ultralight block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

WriteUltraLightBlock

Write data of single block to ISO 14443A mifare ultralight tag.

```
int WriteUltraLightBlock
{
    int iBlock,
    char lpData[]
}
```

Parameters

iBlock

[in] The block index of mifare ultralight tag. Please refer to tag respective document.

lpData

[in] The block data, 8byte.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(WriteUltralightBlock(8, "12345678") != 0)
    AfxMessageBox(_T("Write ultralight block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

GetUid1443B

Get ISO 14443B tag UID.

```
int GetUid1443B
{
    int iType,
    char lpOutUID[]
}
```

Parameters

iType

[in] The tag type. This parameter must be one of the values in the table below.

Value	Key type
1	14443B
2	SR176
3	SRIX4K

lpOutUID

[out] Receive the tag unique identifier.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When tag type is SRIX4K, must read chip ID before get UID. Refer to [SRIX4K Flow Chart](#).

Example

```
char cUID[40];
if(GetUid1443B(1, cUID) != 0)
    AfxMessageBox(_T("Get UID 14443B fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

SRIX4KChipID

Get ISO 14443B SRIX4K tag chip ID.

```
int SRIX4KChipID
{
    char lpOutID[]
}
```

Parameters

lpOutData

[out] Receive the tag chip ID.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cID[4];
if(SRIX4KChipID(cID) != 0)
    AfxMessageBox(_T("Get SRIX4K chip ID fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

SRIX4KReadBlock

Read data of single block from ISO 14443B SRIX4K tag.

```
int SRIX4KReadBlock
{
    int iBlock,
    char lpOutData[]
}
```

Parameters

iBlock

[in] The block index of SRIX4K tag. Please refer to tag respective document.

lpOutData

[out] Receive the block data.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When applies this function, after must remove tag of RFID field. Otherwise, **SRIX4KChipID** function will not be able to get chip ID. Refer to [SRIX4K Flow Chart](#).

Example

```
char cData[128];
if(SRIX4KReadBlock(8, cData) != 0)
    AfxMessageBox(_T("Read SRIX4K block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

SRIX4KWriteBlock

Write data of single block to ISO 14443B SRIX4K tag.

```
int SRIX4KWriteBlock
{
    int iBlock,
    char lpData[]
}
```

Parameters

iBlock

[in] The block index of SRIX4K tag. Please refer to tag respective document.

lpOutData

[in] The block data, 8byte.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When applies this function, after must remove tag of RFID field. Otherwise, **SRIX4KChipID** function will not be able to get chip ID. Refer to [SRIX4K Flow Chart](#).

Example

```
if(SRIX4KWriteBlock(8, "12345678") != 0)
    AfxMessageBox(_T("Write SRIX4K block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

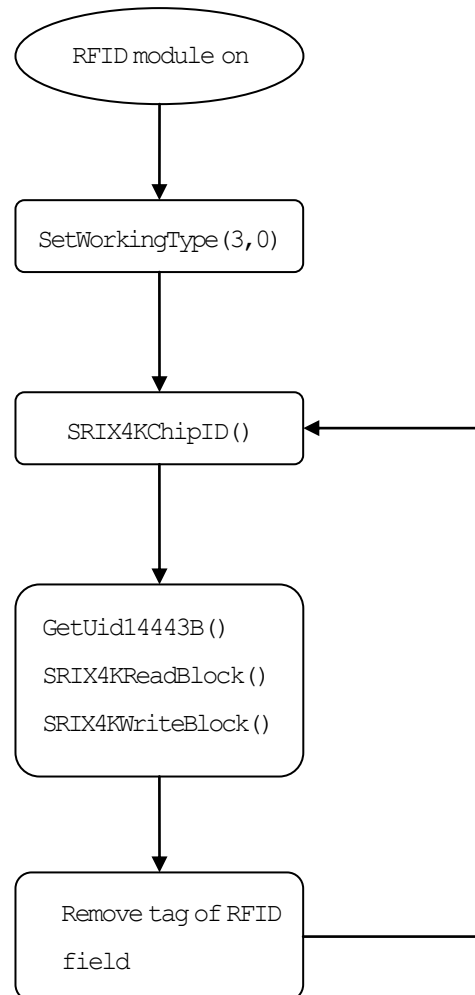
Link DLL: RFID.dll

Device: PT90

RFID Request Flags

Bit	Flag Name	Value	Description
Bit 5	Select_flag	0	Request executed by any tag according to the setting of Address flag.
		1	Request executed only by tag in selected state. The Address flag is set to 0 and the UID field is not included in the request.
Bit 6	Address_flag	0	Request is not addressed. UID field is not included. It can be executed by any tag.
		1	Request is addressed. UID field is included. It is executed only by the tag whose UID matches the UID specified in the request.
Bit 7	Option_flag	0	Meaning is defined by the command description. It is set to 0 if not otherwise defined by the command.
		1	Meaning is defined by the command description.

SRIX4K Flow Chart



SCANAPIAX.DLL

We supply SCANAPIAX.DLL to allow programmer to control the on-board scanning device. There are several functions available for programmer to tailor the application. Programmer can also use Windows CE development tool, such as *Visual Studio 2005*, to develop application program and control the on-board scanning device.

In the SCANAPIAX.DLL library, there are four functional groups can be used to control the scanning device. They are API_SCAN, Scan2Key, Scanner, and Image related functions. Each functional group can be used to control the scanning device in different ways. These four functional groups can not be used at the same time. Programmer should select the most appropriate way to develop target application. The following shows function list of each functional group.

API_SCAN Related Functions

Programmer can use API_SCAN related functions to register application to SCANAPIAX.dll. API_SCAN functions will then send messages to report all activities, including error messages and scan data.

- [API_Register](#) – Register the application to SCANAPIAX.dll
- [API_Unregister](#) – Un-register the application from SCANAPIAX.dll
- [API_GetBarData](#) – Get barcode data into the buffer.
- [API_GetBarDataLength](#) – Return length of the scanned data.
- [API_GetBarType](#) – Return the barcode type.
- [API_GetError](#) – Get the error code.
- [API_GetSysError](#) – Return the system error code.
- [API_GoodRead](#) – Play sound and flash the LED.
- [API_LoadSettingFromFile](#) – Load scanner settings from file.
- [API_Reset](#) – Reset scanner to default settings.
- [API_ResetBarData](#) – Clear data buffer so that the next new scanned data can be load into the buffer.
- [API_SaveSettingToFile](#) – Save the current scanner settings to file.
- [API_SaveSettingsToScanner](#) – Write scanner settings into scanner.
- [S2K_IsLoad](#) – Check if the scan.exe is running or not.
- [S2K_Load](#) – Load or unload the scan.exe.
- [SCAN_QueryStatus](#) – Query scanner settings.
- [SCAN_SendCommand](#) – Send scanner command to change scanner status.
- [SCAN_ResumeSystem](#) – Enable/Disable scan key to resume system.

-
- [SCAN_BatchSetting](#) – Setup scanner in batch command.
 - [SCAN_BatchRead](#) – Read scanner settings in batch command.

Scan2Key Related Functions

Programmer can use Scan2Key related functions to control scan.exe program. When scan.exe is loaded, scanned data will be sent to key buffer. Target application program can retrieve scanned data just like standard keyboard input.

- [PT_OpenScan2Key](#) – Execute scan.exe to scan barcode data into Terminal key buffer.
- [PT_CloseScan2Key](#) – Close scan.exe.
- [PT_SetToDefault](#) – Reset scanner settings to default status.

Scanner Related Functions

Programmer can use Scanner related functions to control scanner module without messages. When target application is using Scanner related functions, the scanned data will be stored in system buffer.

- [PT_EnableScanner](#) – Enable scanner to scan barcode data.
- [PT_DisableScanner](#) – Disable scanner.
- [PT_CheckBarcodeData](#) – Check whether there is barcode data on system buffer.
- [PT_GetBarcodeData](#) – Get barcode data and type from system buffer.
- [PT_SetDefault](#) – Reset scanner settings to default status.

Scan Key Related Functions

- [EnableTriggerKey](#) – Enable and disable scan key.
- [GetLibraryVersion](#) – Get the library version.
- [GetTriggerKeyStatus](#) – Get scan key status.
- [PressTriggerKey](#) – Trigger scan key.
- [TriggerStatus](#) – Get scan key press status.

Image Related Functions

Programmer can use Image related functions to control 2D module without messages. Image related functions is used in 2D module only.

- [IMG_Open](#) – Enable 2D module to capture image.
- [IMG_Close](#) – Disable 2D module.
- [IMG_SetPictureSize](#) – Set VGA resolution · width · height · margin of the capture picture.
- [IMG_GetPictureSize](#) – Get VGA resolution · width · height · margin of the capture picture.

-
-
- [IMG_SetPictureFormat](#) – Set file format of the capture picture.
 - [IMG_GetPictureFormat](#) – Get file format of the capture picture.
 - [IMG_CapturePicture](#) – Capture picture and save to file path.
 - [IMG_SetInvert](#) – Set image invert, include flip 、mirror.
 - [IMG_GetInvert](#) – Get image invert, include flip 、mirror.
 - [IMG_SetRotation](#) – Set image rotate of 0°~270°.
 - [IMG_GetRotation](#) – Get image rotate setting value.
 - [IMG_SetProperty](#) – Set image basic property, include edge sharpen 、brightness 、contrast.
 - [IMG_GetProperty](#) – Get image basic property, include edge sharpen 、brightness 、contrast..
 - [IMG_SetRevise](#) – Set image revise, include compensation 、noise reduction (fly spec).
 - [IMG_GetRevise](#) – Get image revise, include compensation 、noise reduction (fly spec).
 - [IMG_SetQualityJPEG](#) – Set the desired quality when the JPEG image format is selected.
 - [IMG_GetQualityJPEG](#) – Get image JPEG quality setting value.
 - [IMG_SetFilter](#) – Set image filter, include distance filter 、document image filter.
 - [IMG_GetFilter](#) – Get image filter, include distance filter 、document image filter..
 - [IMG_SetTransmit](#) – Set image transmit, include pixel ship 、histogram ship.
 - [IMG_GetTransmit](#) – Get image transmit, include pixel ship 、histogram ship.
 - [IMG_SetScanTimeout](#) – Set a time-out of the 2D module's trigger.
 - [IMG_GetScanTimeout](#) – Get time-out of the 2D module's trigger.
 - [IMG_Default](#) – The 2D module reverts to default setting of image component.

Scan Structure

- [ScannerSetting Structure](#) – Scanner Setting Information used by SCAN_BatchSetting and SCAN_BatchRead.
- [GeneralSetting Structure](#) – Information of Indication, Transmission, Scan, and String settings.
- [Code11 Setting Structure](#) – Information of Code 11 settings.
- [Code39 Setting Structure](#) – Information of Code 39 settings.
- [Code93 Setting Structure](#) – Information of Code 93 settings.
- [Code128 Setting Structure](#) – Information of Code 128 settings.
- [Codabar Setting Structure](#) – Information of Codabar settings.
- [EAN8 Setting Structure](#) – Information of EAN 8 settings.
- [EAN13 Setting Structure](#) – Information of EAN13 settings.

-
- [Industrial25 Setting Structure](#) – Information of Industrial 2 of 5 settings.
 - [Interleaved25 Setting Structure](#) – Information of Interleaved 2 of 5 settings.
 - [MSI Setting Structure](#) – Information of MSI Plessey settings.
 - [UK Setting Structure](#) – Information of UK Plessey settings.
 - [Telepen Setting Structure](#) – Information of Telepen settings.
 - [UPCA Setting Structure](#) – Information of UPCA settings.
 - [UPCE Setting Structure](#) – Information of UPCE settings.
 - [Matrix25 Setting Structure](#) – Information of Matrix25 settings.
 - [PDF417 Setting Structure](#) – Information of PDF-417 settings.
 - [Code16K Setting Structure](#) – Information of Code-16K settings.
 - [ChinaPost Setting Structure](#) – Information of China Post settings.
 - [RSS14 Setting Structure](#) – Information of RSS 14 settings.
 - [RSSLimited Setting Structure](#) – Information of RSS Limited settings.
 - [RSSExpanded Setting Structure](#) – Information of RSS Expanded settings.
 - [Pharmacode Setting Structure](#) – Information of Italian Pharmacode 39 settings.
 - [MicroPDF Setting Structure](#) – Information of MicroPDF settings.
 - [EANUCC Setting Structure](#) – Information of EAN.UCC composite settings.
 - [IATA25 Setting Structure](#) – Information of IATA 2 of 5 settings.
 - [Trioptic Setting Structure](#) – Information of TRI-OPTIC settings.
 - [PosiCode Setting Structure](#) – Information of PosiCode settings.
 - [CodablockF Setting Structure](#) – Information of Codablock F settings.
 - [Code49 Setting Structure](#) – Information of Code 49 settings.
 - [TLC39 Setting Structure](#) – Information of TLC 39 settings.
 - [Planet Setting Structure](#) – Information of Planet Code settings.
 - [Postnet Setting Structure](#) – Information of Postnet settings.
 - [AustralianPost Setting Structure](#) – Information of Australian Post settings.
 - [BritishPost Setting Structure](#) – Information of British Post settings.
 - [CanadianPost Setting Structure](#) – Information of Canadian Post settings.
 - [JapanesePost Setting Structure](#) – Information of Japanese Post settings.
 - [KIXPost Setting Structure](#) – Information of KIX Post settings.
 - [KoreaPost Setting Structure](#) – Information of Korea Post settings.
 - [USPS Setting Structure](#) – Information of USPS settings.
 - [UPU Setting Structure](#) – Information of UPU settings.
 - [Aztec Setting Structure](#) – Information of Aztec Code settings.
 - [DataMatrix Setting Structure](#) – Information of Data Matrix settings.
 - [MaxiCode Setting Structure](#) – Information of MaxiCode settings.
 - [QRCode Setting Structure](#) – Information of QR/MicroQR Code settings.
 - [ChineseSensible Setting Structure](#) – Information of Chinese Sensible Code

settings.

- [OCR Setting Structure](#) – Information of OCR settings.

[Scan Command Table](#)

The Scan Command Table of PT-90 terminal is used for `SCAN_QueryStatus` and `SCAN_SendCommand` functions. The Scan Command provides a different way to setup the scanning device.

When user wants to use this library, user should link `SCANAPIAX.DLL`, `SCANAPIAX.LIB` and the relate functions header file (`SCANAPIAX.H`).

API_SCAN Related Functions

API_Register

To Register the target application to SCANAPIAX.dll so that SCANAPIAX.dll can communicate with the application. It will also set the scanning device to working mode.

```
BOOLAPI_Register  
{  
    HWND hwnd  
}
```

Parameters

hwnd

[in] the window handling the function library will send message to report all activities of the scanning device.

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Remarks

The target application must call API_Unregister to unregister from the dll and close the scanning device after the action been done. The messages can be one of the following::

SM_DATA_READY : Indicating that the barcode data is successfully read and ready for retrieval.

SM_ERROR_SYS : Indicating a system error is caused by calling system function. Calling
API_GetSysError can get the system error code.

SM_ERROR_API : Indicating an error. Calling API_GetError can get the error code.

Example

```
if(!API_Register(theApp.GetMainWnd()->m_hWnd))  
    AfxMessageBox(_T("API_Register FAIL!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_Unregister

To Unregister the target application from SCANAPIAX.dll and close the scanning device.

```
void API_Unregister  
{  
}
```

Parameters

None

Returned Values

None.

Example

```
API_Unregister();
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GetBarData

To Get Barcode into the buffer. When receiving the message SM_DATA_READY, call this function can get the barcode data.

```
UINTAPI_GetBarData
{
    LPBYTE buffer,
    UINT* uiLength,
    UINT* uiBarType
}
```

Parameters

buffer

[out] buffer for string scanned data.

uiLength

[in/out] buffer size

uiBarType

[out] barcode type

Returned Values

Return 1 if the operation is successful, otherwise return 0.

Remarks

If the buffer size is smaller than the scanned data, this function will return 0 and the parameter *uiLength* will return the size of the buffer to adopt the scanned data.

Example

```
if(message == SM_DATA_READY){
    CString strBarData, strBarType;
    UINT uiSize, uiType, i;
    char *pBuf;

    uiSize = uiType = 0;
    API_GetBarData(NULL, &uiSize, &uiType);
    if(uiSize == 0)
        strBarData = _T("No Data");
    else{
        pBuf = (char*)new char[uiSize+1];
```

```
        memset(pBuf, 0, uiSize+1);
        API_GetBarData((LPBYTE)pBuf, &uiSize, &uiType);
        strBarType.Format(_T("%d"), uiType);
        for(i=0; i<strlen(pBuf); i++)
            strBarData += *(pBuf+i);
    }
    AfxMessageBox(_T("Type:") + strBarType + _T("\nBarcode:") + strBarData);
    return 0;
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GetBarDataLength

To Get Length of the scanned data.

```
UINTAPI_GetBarDataLength
{
}
```

Parameters

None

Returned Values

Length of the scanned data

Example

```
if(message == SM_DATA_READY){
    CString strData;
    UINT uiSize, uiType, i, uiLength;
    char *pBuf;
    uiLength=API_GetBarDataLength();
    if(uiLength==0)
        strData=_T("No Data");
    else{
        uiSize=uiLength+1;
        pBuf=(char *)new char[uiSize];
        memset(pBuf, 0, uiSize);
        API_GetBarData((LPBYTE)pBuf, &uiSize, &uiType);
        for(i=0; i<strlen(pBuf); i++)
            strData += *(pBuf+i);
    }
    AfxMessageBox(strData);
    return 0;
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GetBarType

To Get the barcode type.

```
UINTAPI_GetBarType  
{  
}  
}
```

Parameters

None

Returned Values

Always return zero

Remarks

value	Barcode	value	Barcode
BC_CODE11(100)	Code 11	BC_CODABLOCK_F(131)	Codablock F
BC_CODE39(101)	Code 39	BC_CODE49(132)	Code 49
BC_CODE93(102)	Code 93	BC_TLC39(133)	TLC 39
BC_CODE128(103)	Code 128	BC_PLANET(134)	Planet Code
BC_CODABAR(104)	Codabar	BC_POSTNET(135)	Postnet
BC_EAN8(105)	EAN8	BC_AUSTRALIAN_POST(136)	Australian Post
BC_EAN13(106)	EAN13	BC_BRITISH_POST(137)	British post
BC_INTERLEAVED_25(108)	Interleaved 2 of 5	BC_CANADIAN_POST(138)	Canadian Post
BC_MSI_PLESSEY(110)	MSI Plessey	BC_JAPANESE_POST(139)	Japanese Post
BC_UK_PLESSEY(111)	UK Plessey	BC_KIX_POST(140)	KIX Post
BC_TELEPEN(112)	Telepen	BC_KOREA_POST(141)	Korea Post
BC_UPCA(113)	UPCA	BC_USPS(142)	USPS
BC_UPCE(114)	UPCE	BC_UPU(143)	UPU
BC_MATRIX_25(115)	Matrix 25	BC_AZTEC(144)	Aztec Code
BC_PDF417(116)	PDF417	BC_DATA_MATRIX(145)	Data Matrix
BC_CODE16K(117)	Code 16k	BC_MAXI_CODE(146)	MaxiCode
BC_CHINAPOST(118)	China Post	BC_QR_CODE(147)	QR/MicroQR Code
BC_PHARMACODE39(122)	Pharama code39	BC_CHINESE_SENSIBLE(148)	Chinese Sensible Code
BC_MICRO_PDF(123)	Micro PDF	BC_OCR(149)	OCR
BC_TRIOPTIC(126)	TRI-OPTIC	BC_INDUSTRIAL25_IATA25(150)	Industrial 25 / IATA 25
BC_UCCEAN128(128)	UCC/EAN 128	BC_RSS_EANUCC(151)	RSS / EAN UCC
BC_POSI_CODE(130)	PosiCode		

Example

```
uiType=API_GetBarType();
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GetError

To Get the error code.

```
DWORD API_GetError
```

```
{  
}  
}
```

Parameters

None

Returned Values

The returned value can be one of those in the table below:

Constant	Value	Description
ERR_WRITE_FAIL	WM_USER+1	Send commands to scanner module failed.
ERR_SETTING_FAIL	WM_USER+2	Set scanner setting failed.
ERR_SCANNER_NOT_OPEN	WM_USER+3	Open scanner module failed.
ERR_INVALID_FILE	WM_USER+4	Invalid setting file.

Example

```
dwError=API_GetError();  
strMess.Format(_T("API Error Code: %d"), dwError);  
AfxMessageBox(strMess);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GetSysError

To Get the system error code.

```
DWORD API_GetSysError  
{  
}  
}
```

Parameters

None

Returned Values

Returning the system error code that is returned by `GetLastError()`. Descriptions of system error code can be found in MSDN.

Example

```
dwError = API_GetSysError();  
strMess.Format(_T("System Error Code: %d"), dwError);  
AfxMessageBox(strMess);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GoodRead

To activate a buzzer sound when the buzzer of the scanning device is enabled and to flash the LED when the good-read LED of the scanning device is enabled.

```
void API_GoodRead  
{  
}
```

Parameters

None

Returned Values

None.

Remarks

Use *API_GoodRead()* to notify the user that a barcode data is successfully scanned. The buzzer function of the scanning device can be set by Scan Configuration in the control panel. The good-read LED function of the scanning device can be set by *SCAN_SendCommand()* function. If the buzzer and good-read LED functions are disabled, the *API_GoodRead* will do nothing..

Example

```
API_GoodRead();
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_LoadSettingsFromFile

To Load scanner settings from file.

```
BOOLAPI_LoadSettingsFromFile
{
    LPCTSTR filename
}
```

Parameters

filename

[in] the scanner setting file(*.axs)

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Example

```
CString strFile;
CFileDialog  dlg(TRUE, NULL, NULL, OFN_FILEMUSTEXIST|OFN_PATHMUSTEXIST);

if(dlg.DoModal() != IDOK)
    return;

strFile = dlg.GetPathName();
if(theApp.m_API_LoadSettingsFromFile(strFile))
    AfxMessageBox(_T("Load from file Succeed"));
else
    AfxMessageBox(_T("Load from file Fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_Reset

To Reset the scanner settings to default.

```
BOOLAPI_Reset
```

```
{  
}  
}
```

Parameters

None

Returned Values

Return TRUE if the operation is successful, otherwise return FALSE.

Example

```
if(API_Reset())  
    AfxMessageBox(_T("Reset Succeed"));  
else  
    AfxMessageBox(_T("Reset Fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_ResetBarData

To Clear the data buffer to allow the next scanned data coming in..

```
void API_ResetBarData
{
}
```

Parameters

None

Returned Values

None.

Example

```
API_ResetBarData();
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

API_SaveSettingsToFile

To Save current scanner settings to file. The extension file name is “axs”.

```
BOOLAPI_SaveSettingsToFile
{
    LPCTSTR filename
}
```

Parameters

filename

[in] file name of the scanner settings.

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE..

Example

```
CString strFile;
CFileDialog dlg(FALSE, _T("axs"), NULL, OFN_CREATEPROMPT, _T("Scanner Settings Files (*.axs)|*.axs|
"));

if(dlg.DoModal() != IDOK)
    return;

strFile = dlg.GetPathName();
if(API_SaveSettingsToFile(strFile))
    AfxMessageBox(_T("Save to file Succeed"));
else
    AfxMessageBox(_T("Save to file Fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_SaveSettingsToScanner

To Write current scanner settings into the scanner.

```
BOOLAPI_SaveSettingsToScanner  
{  
}  
}
```

Parameters

None

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Example

```
if(API_SaveSettingsToScanner())  
    AfxMessageBox(_T("Save to Scanner Succeed"));  
else  
    AfxMessageBox(_T("Save to Scannere Fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

S2K_IsLoad

To Check if the application program scan.exe(scan barcode and send the scanned data into key buffer) is executing.

```
BOOL S2K_IsLoad  
{  
}  
}
```

Parameters

None

Return Values

The returned value TRUE indicates that scan.exe is running. The returned value FALSE indicates that scan.exe is not running.

Example

```
if(S2K_IsLoad()){  
    AfxMessageBox(_T("scan.exe load"));  
else  
    AfxMessageBox(_T("scan.exe does not load"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

S2K_Load

To Load or Unload the the application program scan.exe.

```
BOOL S2K_Load
{
    BOOL bLoad,
    DWORD dwTimeOut
}
```

Parameters

bLoad

[in] To set TRUE to Load scan.exe and FALSE to Unload scan.exe

dwTimeOut

[in] When Unload scan.exe it will wait until the scan.exe been closed or Timeout by this parameter..

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Example

```
if(S2K_Load(FALSE,1000)){
    AfxMessageBox(_T("unload scan.exe success"));
}
else
    AfxMessageBox(_T("unload scan.exe failed"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

SCAN_QueryStatus

To Query current scanner settings.

```
BOOLSCAN_QueryStatus
{
    int nCommand1,
    int nCommand2,
    char* pReturn
}
```

Parameters

nCommand1

[in] See [scan command table](#).

nCommand2

[in] See [scan command table](#).

pReturn

[out] the current scanner settings. This buffer size must be larger than 100.

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Remarks

The *pReturn* value is depending on *nCommand1* and *nCommand2*. The *nCommand1* and *nCommand2* decide which scanner settings to be queried.

Example

```
char        *pValue;
pValue = (char*)new char[100];
memset(pValue, 0, 100);
//query Buzzer indication setting
SCAN_QueryStatus(5, 3, pValue);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

SCAN_SendCommand

To Send scanner command to change the scanner status.

```
BOOL SCAN_SendCommand
{
    int nCommand1,
    int nCommand2,
    char* pValue
}
```

Parameters

nCommand1

[in] See [scan command table](#).

nCommand2

[in] See [scan command table](#).

pValue

[in] See [scan command table](#).

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Example

```
//Enable Buzzer indication setting
if(SCAN_SendCommand(5,3,"1"))
    AfxMessageBox(_T("Setup complete"));
else
    AfxMessageBox(_T("Setup false"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

SCAN_ResumeSystem

To Enable/Disable scan key to resume system.

```
DWORD SCAN_ResumeSystem
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether to Enable(TRUE) scan key to resume system or Disable(FALSE) scan key to resume system.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
//Enable scan key to resume system
if(SCAN_ResumeSystem(1) == 0)
    AfxMessageBox(_T("Enable scan key to resume system succeed"));
else
    AfxMessageBox(_T("Enable scan key to resume system fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

SCAN_BatchSetting

To Setup all scanner settings in batch command.

```
DWORD SCAN_BatchSetting
{
    ScannerSetting setting
}
```

Parameters

setting

[in] The [ScannerSetting](#) data structure.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#).. If this action fails, possible returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#), [E_FUNC_SETTING_FAIL](#).

Example

```
ScannerSetting    setting;
setting.generalsetting.m_uiLED=0;
setting.Code11.m_uiRead=1;
setting.Code39.m_uiRead=1;
.....
SCAN_BatchSetting(setting);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

SCAN_BatchRead

To Read all scanner settings in batch command.

```
DWORD SCAN_BatchRead
{
    ScannerSetting *setting
}
```

Parameters

setting

[in] Pointer to [ScannerSetting](#) data structure.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, possible returned values are [E_FUNC_SCANNER_NOT_OPEN](#), [E_FUNC_PAR_ERROR](#).

Example

```
ScannerSetting    setting;
SCAN_BatchRead(&setting);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

Scan2Key Related Functions

PT_OpenScan2Key

To Execute scan.exe to scan barcode and send the scanned data into terminal key buffer.

```
BOOL PT_OpenScan2Key  
{  
}
```

Parameters

None

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Example

```
BOOL bResult;  
bResult = PT_OpenScan2Key();  
if (!bResult)  
    AfxMessageBox(_T("PT_OpenScan2Key fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

PT_CloseScan2Key

To Close application program scan.exe.

```
void PT_CloseScan2Key  
{  
}
```

Parameters

None

Returned Values

None.

Example

PT_CloseScan2Key()

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

PT_SetToDefault

To Reset all the scanner settings to default value.

```
int PT_SetToDefault  
{  
}  
}
```

Parameters

None

Returned Values

Returning 1 if the operation is successful, otherwise, return 0.

Example

```
if(!PT_SetToDefault())  
    AfxMessageBox(_T("PT_SetToDefault fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

Scanner Related Functions

PT_EnableScanner

To Enable scanner to scan barcode. This function will also get scanned data from scanning device and store in the system buffer. Application can use the other function, *PT_GetBarcodeData*, to retrieve scanned data from system buffer.

```
int PT_EnableScanner
```

```
{  
}  
}
```

Parameters

None

Returned Values

Returning 0 if the operation is successful, otherwise, return 1.

Example

```
if(!PT_EnableScanner())  
    AfxMessageBox(_T("PT_EnableScanner fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

PT_DisableScanner

To close the scanning device.

```
void PT_DisableScanner  
{  
}
```

Parameters

None

Returned Values

None.

Example

```
PT_DisableScanner();
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

PT_CheckBarcodeData

To Check whether there is scanned barcode data available in system buffer.

```
BOOL PT_CheckBarcodeData  
{  
}  
}
```

Parameters

None

Returned Values

This function returns TRUE if there is scanned barcode data in system buffer and FALSE if there is no scanned barcode data in system buffer.

Example

```
if(PT_CheckBarcodeData())  
    m_strScanData = _T("There are barcode data in system buffer");  
else  
    m_strScanData = _T("There are no barcode data in system buffer");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

PT_GetBarcodeData

To Get Barcode data and barcode type from system buffer.

```
BOOL PT_GetBarcodeData
{
    UINT* uiBarType,
    Char* pBuffer,
    UINT* uiMaxBufferLen
}
```

Parameters

uiBarType

[out] barcode type.

pBuffer

[out] buffer for storing scanned data..

uiMaxBufferLen

[in/out] The maximum buffer size.

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Remarks

If the buffer size is smaller than scanned data, this function will return 0 and the parameter *uiMaxBufferLen* will return the length of the scanned barcode data.

Example

```
if(PT_CheckBarcodeData()){
    if(PT_GetBarcodeData(&uiBarType, pBarData, &uiMaxLen)){
        for(i=0; i<strlen(pBarData); i++)
            m_strScanData += *(pBarData+i);
    }
    else
        m_strScanData = _T("Can't get scan data");
}
else
    m_strScanData = _T("No Scan Data");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

PT_SetDefault

To Reset the scanner settings to default.

```
BOOL PT_SetDefault
```

```
{  
}  
}
```

Parameters

None

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Example

```
if(PT_SetDefault())  
    AfxMessageBox(_T("PT_SetDefault succeed"));  
else  
    AfxMessageBox(_T("PT_SetDefault fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

Scan Key Related Functions

EnableTriggerKey

To Enable or Disable the scan Trigger Key.

```
DWORD EnableTriggerKey
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to Enable(TRUE) or Disable(FALSE) the scan Trigger Key.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If this action fails, the returned values are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

This function is valid only if the scanning device is enabled. A Warm Reset will enable the scan Trigger Key automatically.

Example

```
BOOL bResult;
bResult = EnableTriggerKey(TRUE);
if(bResult)
    AfxMessageBox(_T("EnableTriggerKey Succeed"));
Else
    AfxMessageBox(_T("EnableTriggerKey Fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

GetLibraryVersion

To GetLibrary Version information.

```
int GetLibraryVersion
{
}
```

Parameters

None

Returned Values

The version information, for example, if the returned value is 401, it means that dll version is 4.01

Example

```
int nVersion;
CString strTemp;
nVersion = GetLibraryVersion();
strTemp.Format(_T("Version = %d"), nVersion);
AfxMessageBox(strTemp);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

GefTriggerKeyStatus

To Get scan Trigger Key Status.

```
DWORD GefTriggerKeyStatus  
{  
}  
}
```

Parameters

None.

Returned Values

Returned value 1 indicates that the scan Trigger Key is enabled. Returned value 0 indicates that the scan Trigger Key is disabled.

Example

```
if(GefTriggerKeyStatus())  
    AfxMessageBox(_T("scan key ensable!"));  
else  
    AfxMessageBox(_T("scan key disable!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

PressTriggerKey

To Trigger the Scan Key.

```
DWORD PressTriggerKey
{
    BOOL bPress
}
```

Parameters

bPress

[in] Flag that indicates whether to Press(TRUE) or Release(FALSE) the scan Trigger Key.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If this action fails, the returned value is [E_FUNC_ERROR](#).

Remarks

This function is valid only if the scanning device is enabled.

Example

```
PressTriggerKey(TRUE);
Sleep(1000);
PressTriggerKey(FALSE);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

TriggerStatus

To get the scan Trigger Key status.

```
DWORD TriggerStatus
{
}
```

Parameters

None.

Returned Values

The returned value 1 indicates that the scan Trigger Key is pressed and 0 indicates that scan Trigger Key is released.

Example

```
if(TriggerStatus())
    AfxMessageBox(_T("scan key pressed!"));
else
    AfxMessageBox(_T("scan key release!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90

Image Related Functions

IMG_Open

Open 2D module.

```
BOOL IMG_Open
{
}
```

Parameters

None.

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Example

```
if(IMG_Open())
    AfxMessageBox(_T("IMG_Open Succeed"));
else
    AfxMessageBox(_T("IMG_Open Fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

IMG_Close

Close 2D module.

```
void IMG_Close  
{  
}
```

Parameters

None.

Returned Values

None.

Example

```
IMG_Close();
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

IMG_SetPictureSize

Set the VGA resolution 、width 、height 、margin of the capture picture.

```
DWORD IMG_SetPictureSize
{
    BOOL bVGA,
    int nWidth,
    int nHeight,
    int nMargin
}
```

Parameters

bVGA

[in] set image pixel is VGA resolution(640*480) if *bVGA* is TRUE, else set image pixel is 752*480.

Default is 1.

nWidth

[in] set image width. Range: 0 - 640 when *bVGA* is TRUE, 0 - 752 when *bVGA* is FALSE. Default is 640.

nHeight

[in] set image height. Range: 1 - 480. Default is 480.

nMargin

[in] set cut *nMargin* *2 monad from image margin. Range: 0 - 238. Default is 0.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
if(IMG_SetPictureSize(1,640,480,0) != E_FUNC_SUCCEED)
    AfxMessageBox("Set Picture Size fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

IMG_GetPictureSize

Get the VGA resolution 、width 、height 、margin of the capture picture.

```
DWORD IMG_GetPictureSize
{
    BOOL *bVGA,
    int *nWidth,
    int *nHeight,
    int *nMargin
}
```

Parameters

bVGA

[out] receive image VGA setting value.

nWidth

[out] receive image width setting value.

nHeight

[out] receive image height setting value.

nMargin

[out] receive image margin setting value.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
BOOL bVGA, bResult;
int nWidth, nHeight, nMargin;
if(IMG_GetPictureSize(&bVGA, &nWidth, &nHeight, &nMargin) != E_FUNC_SUCCEED)
    AfxMessageBox("Get Picture Size fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_SetPictureFormat

Set file format of the capture picture.

```
DWORD IMG_SetPictureFormat
{
    int nFormat
}
```

Parameters

nFormat

[in] set file format of the capture picture. The file format can be one of those in the table below:

Value	Format	Capture time
0	TIFF(.tif)	27~31 second
1	JPEG(.jpg)	1~4 second (default)
2	BMP(.bmp)	27~31 second

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
if(IMG_SetPictureFormat(1) != E_FUNC_SUCCEED)
    AfxMessageBox("Set Picture Format fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

IMG_GetPictureFormat

Get file format of the capture picture.

```
DWORD IMG_GetPictureFormat  
{  
    int *nFormat  
}
```

Parameters

nFormat

[out] receive file format setting value.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
int nFormat;  
if(IMG_GetPictureFormat(&nFormat) != E_FUNC_SUCCEED)  
    AfxMessageBox("Set Picture Format fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

IMG_CapturePicture

Capture picture and save to file path.

```
DWORD IMG_CapturePicture
{
    WCHAR*lpFilePath
}
```

Parameters

lpFilePath

[in] file save path after capture picture.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Remarks

If this function use instantly after 2D module trigger, must delay 0.5 second roughly. 2D module store image data need some time.

Example

```
WCHAR szFilePath[256];
wcscpy(szFilePath, L'\\My Documents\\Img_01.jpg');
if(IMG_CapturePicture(szFilePath) != E_FUNC_SUCCEED)
    AfxMessageBox("Capture Picture fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

IMG_SetInvert

Set image invert, include flip 、mirror..

```
DWORD IMG_SetInvert
{
    BOOL bFlip,
    BOOL bMirror
}
```

Parameters

bFlip

[in] set image invert of upside down. Default is 0.

bMirror

[in] set image invert of left to right. Default is 0.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
if(IMG_SetInvert(1,1) != E_FUNC_SUCCEED)
    AfxMessageBox("Set Invert fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

IMG_GetInvert

Get image invert setting value, include flip 、mirror.

```
DWORD IMG_GetInvert
{
    BOOL *bFlip,
    BOOL *bMirror
}
```

Parameters

bFlip

[out] receive image flip setting value.

bMirror

[out] receive image mirror setting value.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEEDED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
BOOL bFlip, bMirror;
if(IMG_GetInvert(&bFlip, &bMirror) != E_FUNC_SUCCEEDED)
    AfxMessageBox("Get Invert fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_SetRotation

Set image rotate of 0°~270°.

```
DWORD IMG_SetRotation  
{  
    int nRotate  
}
```

Parameters

nRotate

[in] set image rotate. The rotate degree can be one of those in the table below:

Value	Rotate degree
0	Rotate 0 degrees to the right (0°) (default)
1	Rotate 90 degrees to the right (90°)
2	Rotate 180 degrees (upside down - 180°)
3	Rotate 90 degrees to the left (270°)

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
if(IMG_SetRotation(0) != E_FUNC_SUCCEED)  
    AfxMessageBox("Set Rotation fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

IMG_GetRotation

Get image rotate setting value.

```
DWORD IMG_GetRotation  
{  
    int *nRotate  
}
```

Parameters

nRotate

[out] receive image rotate setting value.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
int nRotate;  
if(IMG_GetRotation(&nRotate) != E_FUNC_SUCCEED)  
    AfxMessageBox("Get Rotation fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

IMG_SetProperty

Set image basic property, include edge sharpen 、brightness 、contrast.

```
DWORD IMG_SetProperty
{
    int nEdgeSharp,
    int nBright,
    BOOL bContrast
}
```

Parameters

nEdgeSharp

[in] set picture edge sharpen. Range: 0 - 23. Default is 0.

nBright

[in] set picture brighten or darken using gamma correction. A setting of 100 results when *nBright* is 0. The optimal setting is 50. Range: 0 - 255. Default is 0.

bContrast

[in] increases the contrast of the picture. Default is 0.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
if(IMG_SetProperty(0,50,0) != E_FUNC_SUCCEED)
    AfxMessageBox("Set Property fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_GetProperty

Get image basic property setting value, include edge sharpen 、brightness 、contrast.

```
DWORD IMG_GetProperty
{
    int *nEdgeSharp,
    int *nBright,
    BOOL *bContrast
}
```

Parameters

nEdgeSharp

[out] receive image edge sharpen setting value.

nBright

[out] receive image brightness setting value.

bContrast

[out] receive image contrast setting value.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
int nEdgeSharp, nBright;
BOOL bContrast;
if(IMG_GetProperty(&nEdgeSharp, &nBright, &bContrast) != E_FUNC_SUCCEED)
    AfxMessageBox("Get Property fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_SetRevise

Set image revise, include compensation 、 noise reduction (fly spec).

```
DWORD IMG_SetRevise
{
    BOOL bCompensate,
    BOOL bNoiseReduce
}
```

Parameters

bCompensate

[in] flattens the image to account for variations in illumination across the image. Default is 0.

bNoiseReduce

[in] used. to reduce salt and pepper nosis in the image. Default is 0.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
if(IMG_SetRevise(0,0) != E_FUNC_SUCCEED)
    AfxMessageBox("Set Revise fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_GetRevise

Get image revise setting value, include compensation 、 noise reduction (fly spec).

```
DWORD IMG_GetRevise
{
    BOOL *bCompensate,
    BOOL *bNoiseReduce
}
```

Parameters

bCompensate

[out] receive image compensation setting value.

bNoiseReduce

[out] receive image noise reduction setting value.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
BOOL bCompensate, bNoiseReduce
if(IMG_GetRevise(&bCompensate, &bNoiseReduce) != E_FUNC_SUCCEED)
    AfxMessageBox("Get Revise fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_SetQualityJPEG

Set the desired quality when the JPEG image format is selected.

```
DWORD IMG_SetQualityJPEG
{
    int nQuality
}
```

Parameters

nQuality

[in] set JPEG image quality. Only available when select JPEG image format. Smaller numbers result in greater amounts of lossy compression, faster transmission times, and lower quality. Range: 0 - 100. Default is 50.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
if(IMG_SetQualityJPEG(50) != E_FUNC_SUCCEED)
    AfxMessageBox("Set JPEG Quality fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_GetQualityJPEG

Get image JPEG quality setting value.

```
DWORD IMG_GetQualityJPEG  
{  
    int *nQuality  
}
```

Parameters

nQuality

[out] receive image JPEG quality setting value.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
int nQuality;  
if(IMG_GetQualityJPEG(&nQuality) != E_FUNC_SUCCEED)  
    AfxMessageBox("Get JPEG Quality fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanpiax.h

Link Library: scanpiax.lib

Link DLL: scanpiax.dll

Device: PT90-9230

IMG_SetFilter

Set image filter, include distance filter ~ document image filter.

```
DWORD IMG_SetFilter
{
    BOOL bDistance,
    int nDocImage
}
```

Parameters

bDistance

[in] enhances pictures taken from very long distances (greater than 10 feet or 3 m). Default is 0.

nDocImage

[in] sharpens the edges and smoothes the area between the edges of the transmitted text image. This filter should be used with gamma correction, typically provides better JPEG compression than the standard Edge Sharpen. The optimal setting is 26. Range: 0–255. Default is 0.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
if(IMG_SetFilter(0,0) != E_FUNC_SUCCEED)
    AfxMessageBox("Set Filter fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_GetFilter

Get image filter setting value, include distance filter \ document image filter

```
DWORD IMG_GetFilter
{
    BOOL *bDistance,
    int *nDocImage
}
```

Parameters

bDistance

[out] receive distance filter setting value.

nDocImage

[out] receive document image filter: setting value.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
BOOL bDistance;
int nDocImage;
if(IMG_GetFilter(&bDistance, &nDocImage) != E_FUNC_SUCCEED)
    AfxMessageBox("Get Filter fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_SetTransmit

Set image transmit, include pixel ship · histogram ship.

```
DWORD IMG_SetTransmit
{
    int nPixelShip,
    BOOL bHistogram
}
```

Parameters

nPixelShip

[in] can be used to decimate the image by shipping only certain, regularly spaced pixels. The smaller number of pixels shipped, the smaller the image. However, after a certain point, the image becomes unusable. The pixel ship can be one of those in the table below:

Value	Pixel Ship
1	Ship every pixel (default)
2	Ship every 2nd pixel, both horizontally and vertically
3	Ship every 3rd pixel, both horizontally and vertically

bHistogram

[in] a histogram gives a quick picture of the tonal range of an image, or key type. A low-key image has detail concentrated in the shadows; a high-key image has detail concentrated in the highlights; and an average-key image has detail concentrated in the midtones. Default is 0.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
if(IMG_SetTransmit(1,0) != E_FUNC_SUCCEED)
    AfxMessageBox("Set Transmit fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_GetTransmit

Get image transmit setting value, include pixel ship · histogram ship.

```
DWORD IMG_GetTransmit
{
    int *nPixelShip,
    BOOL *bHistogram
}
```

Parameters

nPixelShip

[out] receive image pixel ship setting value.

bHistogram

[out] receive image histogram ship setting value.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
int nPixelShip;
BOOL bHistogram;
if(IMG_GetTransmit(&nPixelShip, &bHistogram) != E_FUNC_SUCCEED)
    AfxMessageBox("Get Transmit fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_SetScanTimeout

Set a time-out of the 2D module's trigger.

```
DWORD IMG_SetScanTimeout
{
    int nTimeout
}
```

Parameters

nTimeout

[in] a time-out of the 2D module's trigger. Range: 0–300 (seconds).

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Remarks

Can call *PressTriggerKey* function or press SCAN key of device to capture image after this function set complete.

This function change timer-out scanner setting, other applied procedure of scanner related will influence.

Example

```
if(IMG_SetScanTimeout(10) != E_FUNC_SUCCEED)
    AfxMessageBox("Set Scan Timeout fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT90-9230

IMG_GetScanTimeout

Get time-out setting value of the 2D module's trigger.

```
DWORD IMG_GetScanTimeout  
{  
    int *nTimeout  
}
```

Parameters

nTimeout

[in] receive time-out setting value.

Returned Values

If this action succeeds, the returned value is [E_FUNC_SUCCEED](#). If this action fails, the returned value is [E_FUNC_ERROR](#), [E_FUNC_SCANNER_NOT_OPEN](#).

Example

```
int nTimeout;  
if(IMG_GetScanTimeout(&nTimeout) != E_FUNC_SUCCEED)  
    AfxMessageBox("Get Scan Timeout fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

IMG_Default

The 2D module reverts to default setting of image component.

```
BOOLIMG_Default
```

```
{  
}  
}
```

Parameters

None

Returned Values

Returning TRUE if the operation is successful, otherwise, return FALSE.

Remarks

This function only reverts to default setting of image component, not revert scan time-out setting.

Example

```
if(IMG_Default())  
    AfxMessageBox("Image Default Succeed");  
else  
    AfxMessageBox("Image Default fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90-9230

Scan Structure

ScannerSetting Structure

This setting file contains information used by [SCAN_BatchSetting](#) and [SCAN_BatchRead](#).

```
Struct ScannerSetting
{
    DWORD cbSize;
    struct GeneralSetting generalsetting;
    struct Code11_Setting Code11;
    struct Code39_Setting Code39;
    struct Code93_Setting Code93;
    struct Code128_Setting Code128;
    struct Codabar_Setting Codabar;
    struct EAN8_Setting EAN8;
    struct EAN13_Setting EAN13;
    struct Industrial25_Setting Indust25;
    struct Interleaved25_Setting Inter25;
    struct MSI_Setting MSIPlessey;
    struct UK_Setting UKPlessey;
    struct Telepen_Setting Telepen;
    struct UPCA_Setting UPCA;
    struct UPCE_Setting UPCE;
    struct Matrix25_Setting Matrix25;
    struct PDF417_Setting PDF417;
    struct Code16K_Setting Code16K;
    struct ChinaPost_Setting ChinaPost;
    struct RSS14_Setting Rss14;
    struct RSSLimited_Setting RssLimited;
    struct RSSExpanded_Setting RssExpanded;
    struct Pharmacode_Setting Pharmacode;
    struct MicroPDF_Setting MicroPDF;
    struct EANUCC_Setting EANUCC;
    struct IATA25_Setting IATA25;
    struct Trioptic_Setting Trioptic;
    struct PosiCode_Setting PosiCode;
    struct CodablockF_Setting CodablockF;
    struct Code49_Setting Code49;
```

```

struct TLC39_Setting TLC39;
struct Planet_Setting Planet;
struct Postnet_Setting Postnet;
struct AustralianPost_Setting AustralianPost;
struct BritishPost_Setting BritishPost;
struct CanadianPost_Setting CanadianPost;
struct JapanesePost_Setting JapanesePost;
struct KIXPost_Setting KIXPost;
struct KoreaPost_Setting KoreaPost;
struct USPS_Setting USPS;
struct UPU_Setting UPU;
struct Aztec_Setting Aztec;
struct DataMatrix_Setting DataMatrix;
struct MaxiCode_Setting MaxiCode;
struct QRCode_Setting QRCode;
struct ChineseSensible_Setting ChineseSensible;
struct OCR_Setting OCR;
}

```

Members

cbSize

This is required. It is the size of structure *ScannerSetting* , in bytes.

generalsetting

[GeneralSetting](#) Structure.

Code11

[Code11_Setting](#) Structure.

Code39

[Code39_Setting](#) Structure.

Code93

[Code93_Setting](#) Structure.

Code128

[Code128_Setting](#) Structure.

Codabar

[Codabar_Setting](#) Structure.

EAN8

[EAN8_Setting](#) Structure.

EAN13

[EAN13_Setting](#) Structure.

Indust25

[Industrial25_Setting](#) Structure.

Inter25

[Interleaved25_Setting](#) Structure.

MSIPlessey

[MSI_Setting](#) Structure.

UKPlessey

[UK_Setting](#) Structure.

Telepen

[Telepen_Setting](#) Structure.

UPCA

[UPCA_Setting](#) Structure.

UPCE

[UPCE_Setting](#) Structure.

Matrix25

[Matrix25_Setting](#) Structure.

PDF417

[PDF417_Setting](#) Structure.

Code16K

[Code16K_Setting](#) Structure.

ChinaPost

[ChinaPost_Setting](#) Structure.

Rss14

[RSS14_Setting](#) Structure.

RssLimited

[RSSLimited_Setting](#) Structure.

RssExpanded

[RSSExpanded_Setting](#) Structure.

Pharmacode

[Pharmacode_Setting](#) Structure.

MicroPDF

[MicroPDF_Setting](#) Structure.

EANUCC

[EANUCC_Setting](#) Structure.

IATA25

[IATA25_Setting](#) Structure.

Trioptic

[Trioptic_Setting](#) Structure.

PosiCode

[PosiCode_Setting](#) Structure.

CodablockF

[CodablockF_Setting](#) Structure.

Code49

[Code49_Setting](#) Structure.

TLC39

[TLC39_Setting](#) Structure.

Planet

[Planet_Setting](#) Structure.

Postnet

[Postnet_Setting](#) Structure.

AustralianPost

[AustralianPost_Setting](#) Structure.

BritishPost

[BritishPost_Setting](#) Structure.

CanadianPost

[CanadianPost_Setting](#) Structure.

JapanesePost

[JapanesePost_Setting](#) Structure.

KIXPost

[KIXPost_Setting](#) Structure.

KoreaPost

[KoreaPost_Setting](#) Structure.

USPS

[USPS_Setting](#) Structure.

UPU

[UPU_Setting](#) Structure.

Aztec

[Aztec_Setting](#) Structure.

DataMatrix

[DataMatrix_Setting](#) Structure.

MaxiCode

[MaxiCode_Setting](#) Structure.

QRCode

[QRCode_Setting](#) Structure.

ChineseSensible

[ChineseSensible_Setting](#) Structure.

OCR

[OCR Setting](#) Structure.

Remarks

The *cbSize* must be the size of Structure *ScannerSetting*, in bytes.

Structure Information

Header: scanapi.h

Device: PT90

GeneralSetting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct GeneralSetting
{
    UINT  m_uiLED;
    UINT  m_uiBeep;
    UINT  m_uiVibrator;
    UINT  m_uiScanResume;
    UINT  m_uiPosition;
    UINT  m_uiTransmit;
    UINT  m_uiAimID;
    UINT  m_uiInverte;
    UINT  m_uiScanTout;
    UINT  m_uiIdleTout;
    UINT  m_uiOrientation
    Unsigned char  m_strPreamble[10];
    Unsigned char  m_strPostamble[10];
}
```

Members

All members

See [Indication](#), [Transmission](#), [Scan](#) and [String setting](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

Code11_Setting Structure

This setting contains information used by *Structure ScannerSetting*.

```
Struct Code11_Setting
{
    UINT  m_uiRead;
    UINT  m_uiChkDig;
    UINT  m_uiMaxLen;
    UINT  m_uiMinLen;
}
```

Members

All members

See [Code11](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

Code39_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Code39_Setting
{
    UINT m_uiRead;
    UINT m_uiChkDig;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
    UINT m_uiFullASCII;
    UINT m_uiXmitStarStop;
    UINT m_uiAppend;
    UINT m_uiCodePage;
}
```

Members

All members

See [Code39](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

Code93_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Code93_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
    UINT m_uiCodePage;
}
```

Members

All members

See [Code93](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

Code128_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Code128_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
    UINT m_uiSBT128;
    UINT m_uiCodePage;
}
```

Members

All members

See [Code128](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

Codabar_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Codabar_Setting
{
    UINT m_uiRead;
    UINT m_uiChkDig;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
    UINT m_uiXmitStarStop;
    UINT m_uiConcat;
}
```

Members

All members

See [Codabar](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

EAN8_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct EAN8_Setting
{
    UINT m_uiRead;
    UINT m_uiXmitChkDig;
    UINT m_ui2DigSupp;
    UINT m_ui5DigSupp;
    UINT m_uiSuppReq;
    UINT m_uiSuppSep;
}
```

Members

All members

See [EAN8](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

EAN13_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct EAN13_Setting
{
    UINT  m_uiRead;
    UINT  m_uiXmitChkDig;
    UINT  m_uiISBNConvert;
    UINT  m_ui2DigSupp;
    UINT  m_ui5DigSupp;
    UINT  m_uiSuppReq;
    UINT  m_uiSuppSep;
}
```

Members

All members

See [EAN13](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

Industrial25_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Industrial25_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [Industrial 2 of 5](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

Interleaved25_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Interleaved25_Setting
{
    UINT m_uiRead;
    UINT m_uiChkDig;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [Interleaved 2 of 5](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

MSL_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct MSL_Setting
{
    UINT m_uiRead;
    UINT m_uiXmitChkDig;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [MSI Plessey](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

UK_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct UK_Setting
{
    UINT m_uiRead;
    UINT m_uiXmitChkDig;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [UKPlessey](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

Telepen_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Telepen_Setting
{
    UINT  m_uiRead;
    UINT  m_uiMaxLen;
    UINT  m_uiMinLen;
    UINT  m_uiFullASCII;
}
```

Members

All members

See [Telepen](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

UPCA_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct UPCA_Setting
{
    UINT m_uiRead;
    UINT m_uiXmitChkDig;
    UINT m_uiCoupon;
    UINT m_uiXmitNumSys;
    UINT m_ui2DigSupp;
    UINT m_ui5DigSupp;
    UINT m_uiSuppReq;
    UINT m_uiSuppSep;
}
```

Members

All members

See [UPCA](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

UPCE_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct UPCE_Setting
{
    UINT  m_uiRead;
    UINT  m_uiXmitChkDig;
    UINT  m_uiUPCEI;
    UINT  m_uiConvertToUPCA;
    UINT  m_uiXmitNumSys;
    UINT  m_ui2DigSupp;
    UINT  m_ui5DigSupp;
    UINT  m_uiSuppReq;
    UINT  m_uiSuppSep;
}
```

Members

All members

See [UPCE](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

Matrix25_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Matrix25_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [Matrix25](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

PDF417_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct PDF417_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [PDF-417](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

Code16K_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Code16K_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [Code-16K](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

ChinaPost_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct ChinaPost_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [ChinaPost](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

RSS14_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct RSS14_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [RSS 14](#) settings in Scan Command Table.

Structure Information

Header: scanpiax.h

Device: PT90

RSSLimited_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct RSSLimited_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [RSS Limited](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

RSSExpanded_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct RSSExpanded_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [RSS Expanded](#) settings in Scan Command Table.

Structure Information

Header: scanpiax.h

Device: PT90

Pharmacode_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Pharmacode_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [Italian Pharmacode 39](#) settings in Scan Command Table .

Structure Information

Header: scanpiax.h

Device: PT90

MicroPDF_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct MicroPDF_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [MicroPDF](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

EANUCC_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct EANUCC_Setting
{
    UINT  m_uiRead;
    UINT  m_uiMaxLen;
    UINT  m_uiMinLen;
    UINT  m_uiUPCEANVer;
}
```

Members

All members

See [EAN.UCC composite](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

IATA25_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct IATA25_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [IATA2 of 5](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

Trioptic_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Trioptic_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [TRI-OPTIC](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

PosiCode_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct PosiCode_Setting
{
    UINT m_uiRead;
    UINT m_uiLimit;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [PosiCode](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

CodablockF_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct CodablockF_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [CodablockF](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

Code49_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Code49_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
```

Members

All members

See [Code49](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

TLC39_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct TLC39_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [TLC39](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

Planet_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Planet_Setting
{
    UINT m_uiRead;
    UINT m_uiXmitChkDig;
}
```

Members

All members

See [Planet Code](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

Postnet_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Postnet_Setting
{
    UINT m_uiRead;
    UINT m_uiXmitChkDig;
}
```

Members

All members

See [Postnet](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

AustralianPost_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct AustralianPost_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [Australian Post](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

BritishPost_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct BritishPost_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [British Post](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

CanadianPost_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct CanadianPost_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [Canadian Post](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

JapanesePost_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct JapanesePost_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [Japanese Post](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

KIXPost_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct KIXPost_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [KIXPost](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

KoreaPost_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct KoreaPost_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [KoreaPost](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

USPS_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct USPS_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [USPS](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

UPU_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct UPU_Setting
{
    UINT m_uiRead;
}
```

Members

All members

See [UPU](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

Aztec_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Aztec_Setting
{
    UINT m_uiRead;
    UINT m_uiRunes;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [Aztec Code](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

DataMatrix_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct DataMatrix_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [Data Matrix](#) settings in Scan Command Table.

Structure Information

Header: scanpiax.h

Device: PT90

MaxiCode_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct MaxiCode_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [MaxiCode](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

QRCode_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct QRCode_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [QR/MicroQR Code](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

ChineseSensible_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct ChineseSensible_Setting
{
    UINT m_uiRead;
    UINT m_uiMaxLen;
    UINT m_uiMinLen;
}
```

Members

All members

See [Chinese Sensible Code](#) settings in Scan Command Table .

Structure Information

Header: scanapi.h

Device: PT90

OCR_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct OCR_Setting
{
    UINT    m_uiRead;
    Unsigned char  m_strTemplate[200];
    Unsigned char  m_strUserDefine_g[50];
    Unsigned char  m_strUserDefine_g[50];
    Unsigned char  m_strCheckChar[50];
}
```

Members

All members

See [OCR](#) settings in Scan Command Table.

Structure Information

Header: scanapi.h

Device: PT90

Scan Command Table

Command1	Command2	Value
5 Indication	2 LED indication	0: Disable 1: Enable *
	3 Buzzer indication	0: Disable 1: Enable *
	4 Vibrator	0: Disable * 1: Enable
	5 Resume System Using ScanKey	0: Disable * 1: Enable
6 Transmission	7 Code ID position	0: Before code data * See Note 1 1: After code data
	8 Code ID transmission	0: Disable * 1: Enable
	12 AIM ID	0: Disable * 1: Enable
7 Scan	8 Inverted image scan	0: Disable * 1: Enable
	17 Scan Timeout (Sec)	0~300 See Note 2 2 *
	18 Idle Timeout (Sec)	0~300 See Note 2 5 *
	19 Working Orientation See Note 9	0: Rotate 0 degree * 1: Rotate 90 degrees 2: Rotate 180 degrees 3: Rotate 270 degrees
8 String setting	3 Preamble characters settings	0x01 ~ 0x7F ASCII code (1-10 characters) 00 *
	4 Postamble characters settings	0x01 ~ 0x7F ASCII code (1-10 characters) 00 *
10 Code 11	1 Read	0: Disable * 1: Enable
	2 Check Digit	1: One digit * 2: Two digits

	4 Max. code length	4~80 80*
	5 Min. code length	4~80 4*
11 Code 39	1 Read	0: Disable 1: Enable*
	2 Check Digit	0: Disable* 1: Validate Only 2: Validate and Transmit
	4 Max. code length	0~48 48*
	5 Min. code length	0~48 0*
	10 Full ASCII	0: Disable* 1: Enable
	13 Transmit Start/Stop Characters	0: Disable* 1: Enable
	16 Append	0: Disable* See Note 3 1: Enable
	18 Code Page	1~3, 81~90 See Note 4 2*
12 Code 93	1 Read	0: Disable 1: Enable*
	4 Max. code length	0~80 80*
	5 Min. code length	0~80 0*
	18 Code Page	1~3, 81~90 See Note 4 2*
13 Code 128	1 Read	0: Disable 1: Enable*
	4 Max. code length	0~80 80*
	5 Min. code length	0~80 0*
	14 ISBT 128	0: Disable* 1: Enable

	18 Code Page	1 ~ 3, 81 ~ 90 See Note 4 2 *
14 Codabar	1 Read	0: Disable * 1: Enable
	2 Check Digit	0: Disable * 1: Validate Only 2: Validate and Transmit
	4 Max. code length	4 ~ 60 60 *
	5 Min. code length	4 ~ 60 4 *
	11 Transmit Start/Stop Characters	0: Disable * 1: Enable
	13 Concatenation	0: Off * See Note 5 1: On 2: Required
15 EAN 8	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	16 Two Digit Supplements	0: Disable * 1: Enable
	17 Five Digit Supplements	0: Disable * 1: Enable
	18 Supplements Required	0: Disable * 1: Enable
	19 Supplements Separator	0: Disable * 1: Enable
16 EAN 13	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	12 ISBN conversion	0: Disable * 1: Enable
	16 Two Digit Supplements	0: Disable * 1: Enable
	17	0: Disable *

	4 Max. code length	1 ~ 60 60 *
	5 Min. code length	1 ~ 60 1 *
	10 Full ASCII	0: Disable * 1: Enable
23 UPCA	1 Read	0: Disable See Note 6 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	13 Coupon	0: Disable * 1: Enable
	15 Transmit Number System	0: Disable 1: Enable *
	16 Two Digit Supplements	0: Disable * 1: Enable
	17 Five Digit Supplements	0: Disable * 1: Enable
	18 Supplements Required	0: Disable * 1: Enable
	19 Supplements Separator	0: Disable * 1: Enable
24 UPCE	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	13 UPCE-1	0: Disable * 1: Enable
	14 Convert to UPC-A	0: Disable * 1: Enable
	15 Transmit Number System	0: Disable 1: Enable *
	16 Two Digit Supplements	0: Disable * 1: Enable
	17 Five Digit Supplements	0: Disable * 1: Enable
	18 Supplements Required	0: Disable * 1: Enable

	19 Supplements Separator	0: Disable * 1: Enable
25 Matrix 25	1 Read	0: Disable * 1: Enable
	4 Max. code length	4~80 80 *
	5 Min. code length	4~80 4 *
26 PDF-417	1 Read	0: Disable 1: Enable *
	4 Max. code length	1~2750 2750 *
	5 Min. code length	1~2750 1 *
27 Code-16K	1 Read	0: Disable * 1: Enable
	4 Max. code length	1~160 160 *
	5 Min. code length	1~160 1 *
28 China Post	1 Read	0: Disable * 1: Enable
	4 Max. code length	4~80 80 *
	5 Min. code length	4~80 4 *
29 RSS 14	1 Read	0: Disable * 1: Enable
30 RSS Limited	1 Read	0: Disable * 1: Enable
31 RSS Expanded	1 Read	0: Disable * 1: Enable
	4 Max. code length	4~74 74 *
	5 Min. code length	4~74 4 *
32	1	0: Disable * See Note 7

Italian Pharmacode 39	Read	1: Enable
33 MicroPDF	1	0: Disable *
	Read	1: Enable
	4	1 ~ 366
	Max. code length	366 *
34 EAN.UCC composite	5	1 ~ 366
	Min. code length	1 *
	1	0: Disable *
	Read	1: Enable
36 IATA2 of 5	4	1 ~ 2435
	Max. code length	2435 *
	5	1 ~ 2435
	Min. code length	1 *
37 TRI-OPTIC	12	0: Disable *
	UPC/EAN Version	1: Enable
	1	0: Disable *
	Read	1: Enable
39 PosiCode	4	4 ~ 48
	Max. code length	48 *
	5	4 ~ 48
	Min. code length	4 *
40 Codablock F	9	0: A & B On (No Limited)
	Limit	1: A & B & Limited A On (Limited B Off)
		2: A & B & Limited B On (Limited A Off) *
	1	0: Disable *
39 PosiCode	Read	1: Enable *
	4	4 ~ 48
	Max. code length	48 *
	5	4 ~ 48
	Min. code length	4 *
40 Codablock F	1	0: Disable *
	Read	1: Enable
	4	1 ~ 2048
	Max. code length	2048 *
39 PosiCode	5	1 ~ 2048
	Min. code length	1 *
	1	0: Disable *
	Read	1: Enable

41 Code 49	1 Read	0: Disable 1: Enable *
	4 Max. code length	1 ~ 81 81 *
	5 Min. code length	1 ~ 81 1 *
42 TLC 39	1 Read	0: Disable * See Note 8 1: Enable
43 Planet Code	1 Read	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
44 Postnet	1 Read	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
45 Australian Post	1 Read	0: Disable * 1: Enable
46 British Post	1 Read	0: Disable * 1: Enable
47 Canadian Post	1 Read	0: Disable * 1: Enable
48 Japanese Post	1 Read	0: Disable * 1: Enable
49 KIX Post	1 Read	0: Disable * 1: Enable
50 Korea Post	1 Read	0: Disable * 1: Enable
	4 Max. code length	4 ~ 48 48 *
	5 Min. code length	4 ~ 48 4 *
51 USPS	1 Read	0: Disable * 1: Enable
52 UPU	1 Read	0: Disable * 1: Enable

53 Aztec Code	1 Read	0: Disable 1: Enable *
	4 Max. code length	1 ~ 3832 3832 *
	5 Min. code length	1 ~ 3832 1 *
	9 Runes	0: Disable * 1: Enable
54 Data Matrix	1 Read	0: Disable 1: Enable *
	4 Max. code length	1 ~ 3116 3116 *
	5 Min. code length	1 ~ 3116 1 *
55 MaxiCode	1 Read	0: Disable 1: Enable *
	4 Max. code length	1 ~ 150 150 *
	5 Min. code length	1 ~ 150 1 *
56 QR/MicroQR Code	1 Read	0: Disable 1: Enable *
	4 Max. code length	1 ~ 7089 7089 *
	5 Min. code length	1 ~ 7089 1 *
57 Chinese Sensible Code	1 Read	0: Disable * 1: Enable
	4 Max. code length	1 ~ 7833 7833 *
	5 Min. code length	1 ~ 7833 1 *
58 OCR See Note 10	1 Read	0: OCR off * 1: OCR-A Font 2: OCR-B Font 3: U.S. Currency Font 4: MCR E13-B Font 5: SEMI Font

	14 Template	0x20~0x7F ASCII code (1-200 characters) ‘ddddddd’ *
	15 User-Defined variable ‘g’	0x20~0x7F ASCII code (1-50 characters) 00 *
	16 User-Defined variable ‘h’	0x20~0x7F ASCII code (1-50 characters) 00 *
	17 Check Character	0x20~0x7F ASCII code (1-50 characters) 00 *

Note 1:

In order to set "Code ID position" values, the "Code ID transmission" must be set.

Barcode	Code ID	Barcode	Code ID
Code 11	h	EAN.UCC composite	y
Code 39	b	IATA 25	f
Code 93	i	Trioptic	=
Code 128	j	PosiCode	W
Codabar	a	CodablockF	q
EAN8	D	Code 49	l
EAN13	d	TLC 39	T
Industrial 2 of 5	f	Planet	L
Interleaved 2 of 5	e	Postnet	P
MSI Plessey	g	Australian Post	A
UK Plessey	n	British Post	B
Telepen	t	Canadian Post	C
UPCA	c	Japanese Post	J
UPCE	E	KIX Post	K
Matrix 25	m	Korea Post	?
PDF 417	r	USPS	M
Code 16k	o	UPU	N
China Post	Q	Aztec	z
RSS 14	y	DataMatrix	w
RSS Limited	y	MaxiCode	x
RSS Expanded	y	QR Code	s
Italian Pharmacode 39	<	Chinese Sensible Code	H
Micro PDF	R	OCR	O

Note 2:

ScanTimeout: the maximum time, in seconds, during which the illumination lights remains on without decoding any barcode.

IdleTimeout: the maximum time, in seconds, during which the scanner remains idle without any action.

Note 3:

This setting allows the imager to append the data from several Code 39 bar codes together before transmitting them to the host.

When this setting is enabled, the imager stores those Code 39 bar codes that start with a space, and does not immediately transmit the data.

The imager stores the data in the order in which the bar codes are read, deleting the first space from each.

The imager transmits the appended data when it reads a Code 39 bar code that starts with a character other than a space.

Note 4:

Code pages define the mapping of character codes to characters.

The Code Page option is available for Code 39, Code 93, and Code 128.

Code Page	Standard	Description
1	CPISO 646	
2 (Default)	ISO 2022	Automatic National Replacement Characters
3	CP Binary	
82	ISO 2022 11 Swe	Swedish Replacement Characters
83	ISO 2022 69 Fra	French/Belgium Replacement Characters
81	ISO 2022 25 Fra	French/Belgium Replacement Characters
84	ISO 2022 11 Ger	German Replacement Characters
85	ISO 2022 11 Ita	Italian Replacement Characters
86	ISO 2022 11 Swi	Swiss Replacement Characters
87	ISO 2022 11 UK	British Replacement Characters
88	ISO 2022 11 Dan	Danish Replacement Characters
89	ISO 2022 11 Nor	Norwegian Replacement Characters
90	ISO 2022 11 Spa	Spanish Replacement Characters

Note 5:

When this setting is enabled, the imager looks for a Codabar symbol having a ‘D’ start character, adjacent to a symbol having a ‘D’ stop character.

In this case the two messages are concatenated into one with the ‘D’ characters omitted.

This selection has no effect on Codabar symbols without Stop/Start D characters.

Note 6:

When this setting is disabled, the imager will auto expand the UPC-A code to the 13 digit, EAN-13 format.

Note 7:

Code 39 must be turned on and Trioptic Code must be turned off while scanning Italian Pharmacode.

Note 8:

Code 39 and MicroPDF417 must be turned on while scanning TLC39.

Note 9:

Some bar codes are direction-sensitive. For example, Kix codes and OCR can misread when scanned sideways or upside down. Use the working orientation settings if your direction-sensitive codes will not usually be presented upright to the scanner.

Note 10:

Use this section to program the imager for optical character recognition (OCR). The PT-9230 reads 6 to 60 point OCR typeface.

Note: OCR is not as secure as bar codes. To enhance security in OCR applications, create an OCR template to match the data, and print an OCR check character.

The PT-9230 will read OCR-A, OCR-B, U.S. Currency Serial Number (Money), MICR E-13B, and SEMI fonts.

You can either select an OCR default, or create your own custom template for the type of OCR format you intend to read. See [OCR](#) settings in Scan Command Table to set font, template and check character.

Note: Setting the template and check character options are essential for OCR reading.

OCR Templates

You can create a custom “template” or character string that defines the length and content of OCR strings that will be read with your imager. There are several choices when creating a custom template for your application. You can create a template for a single format, you can string together several formats, and you can create a template for a user-defined variable. These choices are described in detail below.

Creating an OCR Template

A single template allows you to program the imager to read any combination of characters in the order you specify. Refer to examples that follow the Template Characters table below.

Template Characters Table	
a	represents any alphanumeric character (digit or letter)
c	represents a check character position
d	represents any digit
e	represents any available OCR character
g	represents character from user-defined variable “g”
h	represents character from user-defined variable “h”
l	represents any uppercase letter
t	Marks the start of a new template
r	Multi row indicator
All other characters represent themselves. Spaces can be used. Note: In MICR E-13B templates, TOAD characters (capital letters T,O,A, and D), represent Transit, On Us, Amount, and Dash.	

Example: You need to read any combination of eight digits of OCR-A data.

The setting would be: 58-1-1 and 58-14-“ddddddd”, for example:

37680981

Character Match Sequences

You can input character(s) that you want to match.

Example: You need to read three digit, three specific characters (ABC), three digits of OCR-A data.

The setting would be: Set 58-1-1 and 58-14-“dddABCddd”, for example:

551ABC983

Adding Spaces

You may also need to put spaces in your template.

Example: You need to read three digit, space, three specific characters (ABC), space, three digits of OCR-A data.

The setting would be: Set 58-1-1 and 58-14-“dddABC ddd”, for example:

551 ABC 983

Stringing Together Multiple Formats (Creating “Or” Statements)

You may want to program the imager to accept many OCR formats. To do this, you would string together each format with a ‘|’. This tells the imager to read optical characters that match any one of the formats in the template.

Example: You need to read any combination of eight digits, or a combination of four digits, two uppercase letters, and two digits of OCR-A data.

The setting would be: Set 58-1-1 and 58-14-“dddddddtdddlldd”, for example:

99028650

or

9902XZ50

OCR User-Defined Variables

You can create up to two of your own user variables for an OCR template. These variables will represent any OCR readable characters. The user-defined variables are stored under the letters “g” and “h”. Using command 58-15 and 58-16 to create a user variable “g” and “h”. The letters g and h can then be used in an OCR template to define the variable you specified.

Example: You need a variable to represent the letters “A”, “B”, or “C”. You need to read six digits, and an A, B, or C trailing of OCR-A data.

The setting would be: Set 58-1-1, 58-14-“ddddddggg” and 58-15-“ABC”, for example:

654321ABC

or

654321BAC

or

654321CCC

Reading Multi-Row OCR

Example: You need to read multiple rows of OCR-A data as shown below:

12345678

ABCDEFGH

First, enable the OCR-A font. To read the first row of OCR data, you would program the following template: “ddddddd”.

If you wanted to read the second line of data, you would use the following template: “lllllll”.

To read both lines of OCR at one time, use the variable “r” to indicate the start of a new row. All the other templating variables for the individual rows work the same as previously described. For instance, in the above example, you would use the following template to read both rows: “dddddddrlllllll”.

To read the three rows below, you would use the template: “dddddddrlllllllrlldddd”.

12345678

ABCDEFGH

ABCD1234.

OCR Check Character

You may want to print and verify a check character in order to enhance the security of your OCR application. The PT-9230 can be programmed for almost any type of check character. A number of presets are provided for common check character uses (e.g., modulo 10 and modulo 36).

Example: You need to read any combination of seven digits, with a modulo 10 check character in the eighth position of OCR-A data.

The setting would be: Set 58-1-1, 58-14-“ddddddc” and 58-17-“0123456789”. For example, the following string could be scanned:

01234569

And the output would be: 0123456

OCR Modulo 10 Check Character setting:

58-17-“0123456789”

OCR Modulo 36 Check Character setting:

58-17-“0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ”

OCR User-Defined Check Character

You can customize the check character calculation to suit your application. Each character of the check character alphabet can be programmed in its proper order. The number of characters you enter determines the modulo value for the calculation. By default, the check character computation is unweighted.

Example: To program a modulo 11 check character, you would enter the following 11 characters in order “0123456789X”.

Enable the OCR-A font and set template “ddddddc”, then scan the following string:

6512351X

The imager performs the following check character computation:

$$(6 + 5 + 1 + 2 + 3 + 5 + 1 + X) \text{ modulo } 11 = 0$$

Since the result is zero, the message is considered to be valid, so the output would be: 6512351.

Weighting Options

By default, the check character computation is unweighted. It is possible to use one of two weighted modulo 10 schemes. Weighting is often used to detect if two neighboring characters are transposed, a common error when an operator keys in data.

3-1-3-1 Weighted Modulo 10 Check Character

Starting with the check character and working backward through the message, the imager applies a multiplier of 1, then 3, the 1, then 3, and so on. This is the checking scheme used in many EAN • UCC symbologies, including U.P.C. and Interleaved 2 of 5 (when a check digit is invoked). To apply this weighting scheme, set 58-17-“0123456789x3x1”.

Example: Set 58-1-1, 58-14-“dddddcc” and 58-17-“0123456789x3x1”.

Then scan the string below:

01234565

The imager performs the check character computation below:

$$(0 \times 3 + 1 \times 1 + 2 \times 3 + 3 \times 1 + 4 \times 3 + 5 \times 1 + 6 \times 3 + 5 \times 1) \text{ modulo } 10 = 0$$

Since the result is zero, the message is considered to be valid, so the output would be: 0123456

2-1-2-1 Weighted Modulo 10 Check Character

Starting with the check character and working backward through the message, the imager applies a multiplier of 1, then 2, then 1, then 2, and so on. When the result of the multiplication is greater than 9, add both digits to the running sum. This is often referred to as the LUHN formula. One common application of the LUHN formula is validate credit card numbers. This is the modulo 10 checking scheme used in MSI Code and in Code 32 Pharmaceutical (PARAF). To apply this weighting scheme, set 58-17-“0123456789x2x1”.

Example: Set 58-1-1, 58-14-“dddddcc” and 58-17-“0123456789x2x1”.

Then scan the string below:

0128454

The imager performs the check character computation below:

$$\begin{aligned} & (0 \times 1 + 1 \times 2 + 2 \times 1 + 8 \times 2 + 4 \times 1 + 5 \times 2 + 4 \times 1) \text{ modulo } 10 \\ & = (0 + 2 + 2 + (1 + 6) + 4 + (1 + 0) + 4) \text{ modulo } 10 \\ & = 0 \end{aligned}$$

Since the result is zero, the message is considered to be valid, so the output would be: 012845

OCR ISBN-10 Application Example

One application of OCR is to read the ISBN-10 characters typically encoded using the OCR-A or OCR-B font. This is especially useful when the ISBN-10 number is not encoded in an EAN-13 barcode. The following example shows how to configure the imager to read the ISBN-10 strings on books in Japan. After you have followed the steps below, you will be able to scan the following ISBN-10 number or the additional data below it, depending on the line of text at which the imager is aimed.

I S B N 4 - 5 9 4 - 0 3 0 1 9 - X

C 0 0 9 7 ¥ 8 3 8 E

1. Set 58-1-2.
2. Set 58-15-“0123456789”.
3. In Japan, it is common for two fields to follow the ISBN-10 number, the three digit price field, and the four digit price field. The first field typically starts with a “C” (uppercase c), followed by four digits. The second field typically starts with a “P” or a yen symbol, followed by three or four digits, followed by an “E”. Program the user-defined variable “h” to comprise the “P” and the yen symbol (represented by a backslash). Set 58-16-“P\”.
4. Set up three templates to handle the ISBN-10 number, the three digit price field, and four digit price field.

Template: 58-14-“ISBNggggggggggggCdddd hdddEtCdddd hdddE”

5. Finally, set up the ISBN-10 check digit, which is a special position-weighted modulo 11 checksum. The imager automatically invokes the ISBN-10 check for template rows that are:
 - 1.) at least fourteen characters long,
 - 2.) whose first four characters are the letters “ISBN”,
 - 3.) whose last character is a check character, and
 - 4.) when the modulo 11 check character “0123456789X” is programmed.

Check character: 58-17-“0123456789X”.

Function Return Values

Constant	Value	Description
E_FUNC_SUCCEED	0x00000000	The function returned without error.
E_FUNC_ERROR	0x00000001	The function returned error.
E_FUNC_NULLPTR	0x00000002	A null pointer was passed to the function.
E_FUNC_PAR_ERROR	0x00000003	An invalid parameter was passed to the function.
E_FUNC_SCANNER_NOT_OPEN	0x00000004	The scanning device is not enabled.
E_FUNC_SETTING_FAIL	0x00000005	The function setting failed.
BT_ERR_CREATE_FAIL	0x00001001	BlueTooth module startup fail
BT_ERR_INUSING	0x00001002	BlueTooth module is using by other application
BT_ERR_DEVICE_ERROR	0x00001003	BlueTooth Initial setting fail
BT_ERR_SETTING_FAIL	0x00001004	BlueTooth setup fail
BT_ERR_REG_DEV_FAIL	0x00001005	Register communication port fail
BT_ERR_SPP_COM_FAIL	0x00001006	SPP service com open fail
BT_ERR_INSUFFICIENT	0x00001007	The buffer for receive data is insufficient
BT_ERR_PAIR_FAIL	0x00001008	Pair to device fail
BT_ERR_CHANNEL	0x00001009	SPP channel error
BT_ERR_FTP_SERVER_REJECT	0x00001010	FTP server reject connect request
BT_ERR_DVICE_NOT_CONNECT	0x00001011	FTP service device not connect
BT_ERR_FTP_DIR_FAIL	0x00001012	Search the direction fail
BT_ERR_FTP_EMPTY_FILE	0x00001013	No more file data
BT_ERR_CONNECTED	0x00001014	The device had connected
GPRS_ERR_CREATE_FAIL	0x00003001	GPRS module initial create fail
GPRS_ERR_VERSION_ERROR	0x00003002	HW or FW version error. Device isn't support GPRS
GPRS_ERR_DEVICE_ERROR	0x00003003	GPRS initial setting fail
GPRS_ERR_BUFFER_OVERFLOW	0x00003004	Receive data overruns the buffer boundary
GPRS_ERR_BUFFER_INSUFFICIENT	0x00003005	Define buffer boundary is too small
GPRS_ERR_BUFFER_DATA_LOST	0x00003006	Receive data incomplete, part data lost
GPRS_ERR_RAS_NOT_CONNECT	0x00003007	RAS connection is disconnect or fail
GPRS_ERR_ENTRY_ALREADY_EXIST	0x00003008	RAS entry name already exist
GPRS_ERR_ENTRY_NOT_EXIST	0x00003009	RAS entry name is not exist