



# **PT-90 Mobile Computer SDK Programming Manual**

**DOC NO. UM-PT905-01  
Jan. 2010  
Version 1.0**

© 2010 ARGOX Information Co., Ltd.  
<http://www.argox.com>

---

---

## Table of Contents

|  |           |
|--|-----------|
| <b>OVERVIEW</b> .....                        | <b>2</b>  |
| <b>SDK FUNCTIONS</b> .....                   | <b>3</b>  |
| <b>SYSAPIX.DLL</b> .....                     | <b>4</b>  |
| <b>AUDIO RELATED FUNCTIONS</b> .....         | <b>6</b>  |
| <i>Audio_GetVolume</i> .....                 | <i>6</i>  |
| <i>Audio_SetVolume</i> .....                 | <i>7</i>  |
| <b>BATTERY RELATED FUNCTION</b> .....        | <b>8</b>  |
| <i>GetBatteryStatus</i> .....                | <i>8</i>  |
| <b>DISPLAY RELATED FUNCTIONS</b> .....       | <b>10</b> |
| <i>BacklightOn</i> .....                     | <i>10</i> |
| <i>Display_QueryBacklightIntensity</i> ..... | <i>11</i> |
| <i>GetBacklightStatus</i> .....              | <i>13</i> |
| <i>PowerOnLCD</i> .....                      | <i>14</i> |
| <i>SetBacklightPWM</i> .....                 | <i>15</i> |
| <b>KEYPAD RELATED FUNCTIONS</b> .....        | <b>16</b> |
| <i>EnablePowerButton</i> .....               | <i>16</i> |
| <i>GetKeypadAlphaMode</i> .....              | <i>17</i> |
| <i>SendKbdVisualKey</i> .....                | <i>18</i> |
| <i>SetKeypadAlphaMode</i> .....              | <i>19</i> |
| <b>LED RELATED FUNCTIONS</b> .....           | <b>20</b> |
| <i>GetKeypadLEDStatus</i> .....              | <i>20</i> |
| <i>GoodReadLEDon</i> .....                   | <i>21</i> |
| <i>KeypadLEDon</i> .....                     | <i>22</i> |
| <i>QueryKeypadLEDIntensity</i> .....         | <i>23</i> |
| <i>SetKeypadPWM</i> .....                    | <i>25</i> |
| <b>SYSTEM RELATED FUNCTIONS</b> .....        | <b>26</b> |
| <i>CallSuspend</i> .....                     | <i>26</i> |
| <i>EnableAutoConnect</i> .....               | <i>27</i> |
| <i>RegisterAlphaKeyNotification</i> .....    | <i>28</i> |
| <i>ShowChineseIME</i> .....                  | <i>29</i> |
| <i>ShowDesktop</i> .....                     | <i>30</i> |
| <i>ShowExploreToolbar</i> .....              | <i>31</i> |
| <i>ShowTaskbar</i> .....                     | <i>32</i> |
| <i>UnRegisterAlphaKeyNotification</i> .....  | <i>33</i> |
| <b>VIBRATOR RELATED FUNCTIONS</b> .....      | <b>34</b> |

---

|  |           |
|--|-----------|
| <i>VibratorOn</i> .....                | 34        |
| BLUETOOTHRELATEDFUNCTION.....          | 35        |
| <i>BT_Enable</i> .....                 | 35        |
| <i>BT_Disable</i> .....                | 36        |
| WLANRELATEDFUNCTION.....               | 37        |
| <i>WL_Enable</i> .....                 | 37        |
| <i>WL_Disable</i> .....                | 38        |
| <b>SCANAPIX.DLL.....</b>               | <b>39</b> |
| API_SCANRELATEDFUNCTIONS.....          | 42        |
| <i>API_Register</i> .....              | 42        |
| <i>API_Unregister</i> .....            | 43        |
| <i>API_GetBarData</i> .....            | 44        |
| <i>API_GetBarDataLength</i> .....      | 46        |
| <i>API_GetBarType</i> .....            | 47        |
| <i>API_GetError</i> .....              | 48        |
| <i>API_GetSysError</i> .....           | 50        |
| <i>API_GoodRead</i> .....              | 50        |
| <i>API_LoadSettingsFromFile</i> .....  | 51        |
| <i>API_Reset</i> .....                 | 52        |
| <i>API_ResetBarData</i> .....          | 53        |
| <i>API_SaveSettingsToFile</i> .....    | 53        |
| <i>API_SaveSettingsToScanner</i> ..... | 55        |
| <i>S2K_IsLoad</i> .....                | 56        |
| <i>S2K_Load</i> .....                  | 57        |
| <i>SCAN_QueryStatus</i> .....          | 58        |
| <i>SCAN_SendCommand</i> .....          | 59        |
| <i>SCAN_ResumeSystem</i> .....         | 60        |
| <i>SCAN_BatchSetting</i> .....         | 61        |
| <i>SCAN_BatchRead</i> .....            | 62        |
| SCAN2KEYRELATEDFUNCTIONS.....          | 63        |
| <i>PT_OpenScan2Key</i> .....           | 63        |
| <i>PT_CloseScan2Key</i> .....          | 64        |
| <i>PT_SetToDefault</i> .....           | 65        |
| SCANNERRELATEDFUNCTIONS.....           | 66        |
| <i>PT_EnableScanner</i> .....          | 66        |
| <i>PT_DisableScanner</i> .....         | 67        |
| <i>PT_CheckBarcodeData</i> .....       | 68        |
| <i>PT_GetBarcodeData</i> .....         | 69        |

---

|   |            |
|---|------------|
| <i>PT_SetDefault</i> .....                  | 71         |
| SCANKEY RELATED FUNCTIONS.....              | 72         |
| <i>EnableTriggerKey</i> .....               | 72         |
| <i>GetLibraryVersion</i> .....              | 73         |
| <i>GetTriggerKeyStatus</i> .....            | 74         |
| <i>PressTriggerKey</i> .....                | 75         |
| <i>TriggerStatus</i> .....                  | 76         |
| SCANSTRUCTURE.....                          | 77         |
| <i>ScannerSettingStructure</i> .....        | 77         |
| <i>GeneralSettingStructure</i> .....        | 80         |
| <i>Code11_SettingStructure</i> .....        | 81         |
| <i>Code39_SettingStructure</i> .....        | 82         |
| <i>Code93_SettingStructure</i> .....        | 83         |
| <i>Code128_SettingStructure</i> .....       | 84         |
| <i>Codabar_SettingStructure</i> .....       | 85         |
| <i>EAN8_SettingStructure</i> .....          | 86         |
| <i>EAN13_SettingStructure</i> .....         | 87         |
| <i>Industrial25_SettingStructure</i> .....  | 88         |
| <i>Interleaved25_SettingStructure</i> ..... | 89         |
| <i>MSI_SettingStructure</i> .....           | 90         |
| <i>UK_SettingStructure</i> .....            | 91         |
| <i>Telepen_SettingStructure</i> .....       | 92         |
| <i>UPCA_SettingStructure</i> .....          | 93         |
| <i>UPCE_SettingStructure</i> .....          | 94         |
| <i>Matrix25_SettingStructure</i> .....      | 95         |
| <i>UEGeneral_SettingStructure</i> .....     | 96         |
| <i>IATA25_SettingStructure</i> .....        | 97         |
| <i>Trioptic_SettingStructure</i> .....      | 98         |
| <i>RSS_SettingStructure</i> .....           | 99         |
| <b>SCAN COMMAND TABLE.....</b>              | <b>100</b> |
| <b>FUNCTION RETURN VALUES.....</b>          | <b>108</b> |

---

---

## 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.

### 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.

#### Vibrator Related Functions

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

#### Bluetooth Related Functions

- [BT\\_Enable](#) – ENABLE Bluetooth.
- [BT\\_Disable](#) – DISABLE Bluetooth.

#### WLAN Related Functions

- [WL\\_Enable](#) – ENABLE WLAN.
- [WL\\_Disable](#) – DISABLE WLAN.

---

## 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\\_SUCCEEDED](#). If this action fails, the returned value is [E\\_FUNC\\_ERROR](#).

#### Example

```
DWORD dwResult, dwVolume;
dwResult = Audio_GetVolume(&dwVolume);
if(dwResult != E_FUNC_SUCCEEDED)
    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 returned value can be one of the values in the table below.

| 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\\_SUCCEED](#). 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_SUCCEED)
    AfxMessageBox(_T("BacklightOn fail"));
```

#### Requirements

**OS Versions:** Windows CE 6.0 or beyond.

**Header:** sysapiax.h

**Link Library:** sysapiax.lib

**Link DLL:** sysapiax.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 to turn ON (TRUE) or OFF (FALSE) the LCD power.

### 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

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_SUCCEEDED)
    AfxMessageBox(_T("PowerOnLCD fail"));
Sleep(3000);
dwResult=PowerOnLCD(TRUE);//power on LCD
if(dwResult !=E_FUNC_SUCCEEDED)
    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.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

---

## *SendKbdVirtualKey*

To send a VIRTUALKEY to key buffer.

```
DWORD SendKbdVirtualKey
{
    BYTE Key
}
```

### Parameters

*Key*

[in] Specifies a virtual-key code.

### Returned Values

If this action succeeds, the returned value is [E\\_FUNC\\_SUCCEEDED](#). 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\\_SUCCEED](#). 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_SUCCEED)  
    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\\_SUCCEEDED](#). 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_SUCCEEDED)
    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\\_SUCCEEDED](#). 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_SUCCEEDED)
    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.

```
BOOL EnableAutoConnect  
{  
    BOOL bEnable  
}
```

### 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 message window 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.

```
BOOL ShowDesktop
{
    BOOL bShow
}
```

### 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 ExploreToolbar 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

---

---

## 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

---

---

## Bluetooth Related Function

### *BT\_Enable*

To ENABLE the Bluetooth function.

```
BOOLBT_Enable  
{  
}  
}
```

### Parameters

*None.*

### Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

### Example

```
BOOL bResult;  
bResult = BT_Enable();  
if(bResult == FALSE)  
    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\_Disable*

To DISABLE the Bluetooth function

```
BOOL BT_Disable  
{  
}  
}
```

### Parameters

*None.*

### Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

### Example

```
BOOL bResult;  
bResult = BT_Disable();  
if(bResult == FALSE)  
    AfxMessageBox(_T("BlueTooth disable 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

---

---

## 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 three functional groups can be used to control the scanning device. They are API\_SCAN, Scan2Key, and Scanner related functions. Each functional group can be used to control the scanning device in different ways. These three 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\\_DiableScanner](#) – Disable scanner.
- [PT\\_CheckBarcodeData](#) – Check whether there is scanned data in system buffer.
- [PT\\_GetBarcodeData](#) – Get barcode data and type from system buffer.
- [PT\\_SetDeault](#) – 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 trigger status.

#### 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 Code11 settings.
- [Code39\\_Setting Structure](#) – Information of Code39 settings.
- [Code93\\_Setting Structure](#) – Information of Code93 settings.
- [Code128\\_Setting Structure](#) – Information of Code128 settings.
- [Codabar\\_Setting Structure](#) – Information of Codabar settings.
- [EAN8\\_Setting Structure](#) – Information of EAN8 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.
  - [UEGeneral\\_Setting Structure](#) – Information of UPC/EAN General settings.
  - [IATA25\\_Setting Structure](#) – Information of IATA 2 of 5 settings.
  - [Trioptic\\_Setting Structure](#) – Information of TRI-OPTIC settings.
  - [RSS\\_Setting Structure](#) – Information of RSS 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.

```
BOOL API_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*

### **Return 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.

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

#### **Parameters**

*buffer*

[out] buffer for scanned data string.

*uiLength*

[in/out] buffer size

*uiBarType*

[out] barcode type. This parameter is ignored and should be set to NULL.

#### **Returned Values**

Returning 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

```
UINT API_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. **This function is no longer supported as of PT90.** Always return zero.

```
UINT API_GetBarType  
{  
}  
}
```

### Parameters

*None*

### Returned Values

Always return zero

### 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("APIError 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.

```
BOOL API_LoadSettingsFromFile
{
    LPCWSTR 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**

Returning 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*

#### **ReturnedValues**

None.

#### **Example**

```
API_ResetBarData();
```

#### **Requirements**

**OS Versions:** Windows CE 6.0 or beyond.

**Header:** scanapiax.h

**Link Library:** scanapiax.lib

**Link DLL:** scanapiax.dll

**Device:** PT90

---

## *API\_SaveSettingsToFile*

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

```
BOOL API_SaveSettingsToFile
{
    LPCWSTR 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.

```
BOOL API_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*

### **Returned 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.

```
BOOL SCAN_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.

### ReturnedValues

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:** scanapiax.h

**Link Library:** scanapiax.lib

**Link DLL:** scanapiax.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:** scanapiax.h

**Link Library:** scanapiax.lib

**Link DLL:** scanapiax.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\\_SUCCEEDED](#). 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*

[out] Pointer to [ScannerSetting](#) data structure.

### Returned Values

If this action succeeds, the returned value is [E\\_FUNC\\_SUCCEEDED](#). 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:** scanapiax.h

**Link Library:** scanapiax.lib

**Link DLL:** scanapiax.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:** scanapiax.h

**Link Library:** scanapiax.lib

**Link DLL:** scanapiax.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:** scanapiax.h

**Link Library:** scanapiax.lib

**Link DLL:** scanapiax.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. This parameter is ignored and should be set to NULL.

*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:** scanapi.h

**Link Library:** scanapi.lib

**Link DLL:** scanapi.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:** scanapiax.h

**Link Library:** scanapiax.lib

**Link DLL:** scanapiax.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\\_SUCCEED](#). 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("Enable TriggerKey Succeed"));
Else
    AfxMessageBox(_T("Enable TriggerKey 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 Get Library Version information.

```
int GetLibraryVersion
{
}
```

### **Parameters**

*None*

### **Returned Values**

The version information, for example, if the returned value is 301, it means that dll version is 3.01

### **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

---

## *GetTriggerKeyStatus*

To Get scan Trigger Key Status.

```
DWORD GetTriggerKeyStatus
```

```
{  
}  
}
```

### 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(GetTriggerKeyStatus())  
    AfxMessageBox(_T("scan key enable!"));  
else  
    AfxMessageBox(_T("scan key disable!"));
```

### Requirements

**OS Versions:** Windows CE 6.0 or beyond.

**Header:** scanapi.h

**Link Library:** scanapi.lib

**Link DLL:** scanapi.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:** scanapiax.h

**Link Library:** scanapiax.lib

**Link DLL:** scanapiax.dll

**Device:** PT90

---

---

## 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 Indus25;
    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 UEGeneral_Setting UEGeneral;
    struct IATA25_Setting IATA25;
    struct Trioptic_Setting Trioptic;
    struct Rss_Setting RSS;
}
```

### 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.

*UEGeneral*

[UEGeneral\\_Setting](#) Structure.

*IATA25*

[IATA25\\_Setting](#) Structure.

*Trioptic*

[Trioptic\\_Setting](#) Structure.

*RSS*

[RSS\\_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.Scanner.Setting*.

```
Struct GeneralSetting
{
    UINT  m_uiLED;
    UINT  m_uiBeep;
    UINT  m_uiVibrator;
    UINT  m_uiScanResume;
    UINT  m_uiPosition;
    UINT  m_uiAimID;
    UINT  m_uiGlobalMin;
    UINT  m_uiGlobalLock;
    UINT  m_ui1Type;
    UINT  m_ui1Length;
    UINT  m_ui2Type;
    UINT  m_ui2Length;
    UINT  m_ui3Type;
    UINT  m_ui3Length;
    UINT  m_ui4Type;
    UINT  m_ui4Length;
    UINT  m_ui5Type;
    UINT  m_ui5Length;
    UINT  m_ui6Type;
    UINT  m_ui6Length;
    UINT  m_ui7Type;
    UINT  m_ui7Length;
    UINT  m_uiScanTout;
    UINT  m_uiIdleTout;
    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_uiXmitChkDig;
    UINT m_uiCodeID;
}
```

### 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.Scanner.Setting*.

```
Struct Code39_Setting
{
    UINT m_uiRead;
    UINT m_uiChkDig;
    UINT m_uiXmitChkDig;
    UINT m_uiCodeID;
    UINT m_uiFullASCII;
    UINT m_uiXmitStarStop;
    UINT m_uiPharmacode;
    UINT m_uiPharmacodeOnly;
}
```

### **Members**

*All members*

See [Code39](#) settings in Scan Command Table.

### **Structure Information**

**Header:** scanapiax.h

**Device:** PT90

---

### *Code93\_Setting Structure*

This setting file contains information used by *Structure.Scanner.Setting*.

```
Struct Code93_Setting
{
    UINT m_uiRead;
    UINT m_uiCodeID;
}
```

### **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.Scanner.Setting*.

```
Struct Code128_Setting
{
    UINT  m_uiRead;
    UINT  m_uiCodeID;
    UINT  m_uiUCCEAN128;
    UINT  m_uiISBT128;
    UINT  m_uiISBT128XmitID;
    UINT  m_uiISBT128Concat;
    UINT  m_uiISBT128Form;
}
```

### 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.Scanner.Setting*.

```
Struct Codabar_Setting
{
    UINT m_uiRead;
    UINT m_uiChkDig;
    UINT m_uiXmitChkDig;
    UINT m_uiCodeID;
    UINT m_uiXmitStarStop;
    UINT m_uiDualField;
}
```

### 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.Scanner.Setting*.

```
Struct EAN8_Setting
{
    UINT m_uiRead;
    UINT m_uiXmitChkDig;
    UINT m_uiCodeID;
    UINT m_uiConvertToEAN13;
}
```

### 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.Scanner.Setting*.

```
Struct EAN13_Setting
{
    UINT  m_uiRead;
    UINT  m_uiXmitChkDig;
    UINT  m_uiCodeID;
    UINT  m_uiISBNConvert;
    UINT  m_uiISBNSuppReq;
    UINT  m_uiISMNConvert;
    UINT  m_uiISMNSuppReq;
}
```

### 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.Scanner.Setting*.

```
Struct Industrial25_Setting
{
    UINT m_uiRead;
    UINT m_uiCodeID;
}
```

### 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_uiXmitChkDig;
    UINT m_uiCodeID;
}
```

### Members

*All members*

See [Interleaved 2 of 5](#) settings in Scan Command Table.

### Structure Information

**Header:** scanapi.h

**Device:** PT90

---

## *MSI\_Setting Structure*

This setting file contains information used by *Structure.Scanner.Setting*.

```
Struct MSI_Setting
{
    UINT m_uiRead;
    UINT m_uiChkDig;
    UINT m_uiXmitChkDig;
    UINT m_uiCodeID;
}
```

### 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.Scanner.Setting*.

```
Struct UK_Setting
{
    UINT m_uiRead;
    UINT m_uiXmitChkDig;
    UINT m_uiCodeID;
}
```

### Members

*All members*

See [UK.Plessey](#) settings in Scan Command Table.

### Structure Information

**Header:** scanapi.h

**Device:** PT90

---

## *Telepen\_Setting Structure*

This setting file contains information used by *Structure.Scanner.Setting*.

```
Struct Telepen_Setting
{
    UINT m_uiRead;
    UINT m_uiCodeID;
    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.Scanner.Setting*.

```
Struct UPCA_Setting
{
    UINT  m_uiRead;
    UINT  m_uiXmitChkDig;
    UINT  m_uiCodeID;
    UINT  m_uiConvertToEAN13;
    UINT  m_uiCoupon;
    UINT  m_uiCouponXmitID;
    UINT  m_uiXmitNumSys;
}
```

### 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.Scanner.Setting*.

```
Struct UPCE_Setting
{
    UINT  m_uiRead;
    UINT  m_uiXmitChkDig;
    UINT  m_uiCodeID;
    UINT  m_uiConvertToUPCA;
    UINT  m_uiXmitNumSys;
}
```

### 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.Scanner.Setting*.

```
Struct Matrix25_Setting
{
    UINT m_uiRead;
    UINT m_uiChkDig;
    UINT m_uiXmitChkDig;
    UINT m_uiCodeID;
}
```

### Members

*All members*

See [Matrix25](#) settings in Scan Command Table.

### Structure Information

**Header:** scanapi.h

**Device:** PT90

---

## *UEGeneral\_Setting Structure*

This setting file contains information used by *Structure.Scanner.Setting*.

```
Struct UEGeneral_Setting
{
    UINT m_uiSuppReq;
    UINT m_ui2DigSupp;
    UINT m_ui5DigSupp;
    UINT m_ui2DigRedu;
    UINT m_ui5DigRedu;
    UINT m_uiGTIN;
}
```

### Members

*All members*

See [UPC/EAN General](#) settings in Scan Command Table.

### Structure Information

**Header:** scanapi.h

**Device:** PT90

---

## *IATA25\_Setting Structure*

This setting file contains information used by *Structure.Scanner.Setting*.

```
Struct IATA25_Setting
{
    UINT m_uiRead;
    UINT m_uiCodeID;
}
```

### 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;
    UINT m_uiCodeID;
    UINT m_uiConversion;
}
```

### Members

*All members*

See [TRI-OPTIC](#) settings in Scan Command Table.

### Structure Information

**Header:** scanapi.h

**Device:** PT90

---

## *RSS\_Setting Structure*

This setting file contains information used by *Structure.Scanner.Setting*.

```
Struct RSS_Setting
{
    UINT m_uiRead;
    UINT m_uiXmitAppID;
    UINT m_uiXmitChkDig;
    UINT m_uiXmitSymID;
}
```

### Members

*All members*

See [RSS](#) settings in Scan Command Table.

### Structure Information

**Header:** scanapi.h

**Device:** PT90

## Scan Command Table

| Command1                           | Command2                           | Value                                       |
|------------------------------------|------------------------------------|---|
| 5<br>Indication                    | 2<br>LED indication                | 0: Disable<br>1: Enable *                   |
|                                    | 3<br>Buzzer indication             | 0: Disable<br>1: Enable *                   |
|                                    | 4:<br>Vibrator                     | 0: Disable*<br>1: Enable                    |
|                                    | 5<br>Resume System Using ScanKey   | 0: Disable*<br>1: Enable                    |
| 6<br>Transmission                  | 7<br>Code ID position              | 0: Before code data *<br>1: After code data |
|                                    | 12<br>ALMID                        | 0: Disable*<br>1: Enable                    |
|                                    | 13<br>Code ID                      | 0: Clear<br>1: Proprietary See Note 1       |
| 7<br>Scan                          | 6:<br>Global min. code length      | 0-80(0: Disable)<br>3 *                     |
|                                    | 9:<br>Global lock code length      | 0-80(0: Disable)<br>0 *                     |
|                                    | 10:<br>Configurable code length #1 | See Note 2                                  |
|                                    | 11:<br>Configurable code length #2 | See Note 2                                  |
|                                    | 12:<br>Configurable code length #3 | See Note 2                                  |
|                                    | 13:<br>Configurable code length #4 | See Note 2                                  |
|                                    | 14:<br>Configurable code length #5 | See Note 2                                  |
|                                    | 15:<br>Configurable code length #6 | See Note 2                                  |
| 16:<br>Configurable code length #7 | See Note 2                         |   |

|                     |                                      |  |
|---------------------|--------------------------------------|--|
|                     | 17:<br>Scan Timeout (Sec)            | 1~30 See Note 3<br>2 *                         |
|                     | 18:<br>Idle Timeout (Sec)            | 0~255 (0: Disable) See Note 3<br>5 *           |
| 8<br>String setting | 3<br>Preamble characters settings    | 0x01~0x7F ASCII code (1-10 characters)<br>00 * |
|                     | 4<br>Postamble characters settings   | 0x01~0x7F ASCII code (1-10 characters)<br>00 * |
| 10<br>Code 11       | 1<br>Read                            | 0: Disable *<br>1: Enable                      |
|                     | 2<br>Check Digit                     | 1: One digit *<br>2: Two digits                |
|                     | 3<br>Transmit Check Digit            | 0: Disable *<br>1: Enable                      |
|                     | 8<br>Code ID setting                 | 0: Disable *<br>0x01~0x7F ASCII code (1 bytes) |
| 11<br>Code 39       | 1<br>Read                            | 0: Disable<br>1: Enable *                      |
|                     | 2<br>Check Digit                     | 0: Disable *<br>1: Enable                      |
|                     | 3<br>Transmit Check Digit            | 0: Disable *<br>1: Enable                      |
|                     | 8<br>Code ID setting                 | 0: Disable *<br>0x01~0x7F ASCII code (1 bytes) |
|                     | 10<br>Full ASCII                     | 0: Disable *<br>1: Enable                      |
|                     | 13<br>Transmit Start/Stop Characters | 0: Disable *<br>1: Enable                      |
|                     | 14<br>Italian Pharmacode             | 0: Disable *<br>1: Enable                      |
|                     | 15<br>Italian Pharmacode Only        | 0: Disable<br>1: Enable *                      |
| 12<br>Code 93       | 1<br>Read                            | 0: Disable<br>1: Enable *                      |
|                     | 8<br>Code ID setting                 | 0: Disable *<br>0x01~0x7F ASCII code (1 bytes) |
| 13<br>Code 128      | 1<br>Read                            | 0: Disable<br>1: Enable *                      |

|               |   |  |
|---------------|---|--|
|               | 8<br>Code ID setting                    | 0: Disable*<br>0x01~0x7F ASCII code(1 bytes)                                     |
|               | 10<br>UCC/EAN 128                       | 0: Disable*<br>1: Enable   |
|               | 14<br>ISBT 128                          | 0: Disable*<br>1: Enable   |
|               | 15<br>ISBT 128 Transmit Identifier Data | 0: Disable<br>1: Enable*   |
|               | 16<br>ISBT 128 Concatenation            | 0: Disable<br>1: Enable*   |
|               | 17<br>ISBT 128 Form:                    | 0: =A+=%*<br>1: =A+&<br>2: =A+&!<br>3: =<+=><br>4: =<+&><br>5: &<+=><br>6: &<+&> |
| 14<br>Codabar | 1<br>Read                               | 0: Disable*<br>1: Enable   |
|               | 2<br>Check Digit                        | 0: Disable*<br>1: Enable   |
|               | 3<br>Transmit Check Digit               | 0: Disable*<br>1: Enable   |
|               | 8<br>Code ID setting                    | 0: Disable*<br>0x01~0x7F ASCII code(1 bytes)                                     |
|               | 11<br>Transmit Start/Stop Characters    | 0: Disable*<br>1: Enable   |
|               | 12<br>Dual Field                        | 0: Disable*<br>1: Enable   |
| 15<br>EAN8    | 1<br>Read                               | 0: Disable<br>1: Enable*   |
|               | 3<br>Transmit Check Digit               | 0: Disable<br>1: Enable*   |
|               | 8<br>Code ID setting                    | 0: Disable*<br>0x01~0x7F ASCII code(1 bytes)                                     |
|               | 13<br>Convert to EAN-13                 | 0: Disable*<br>1: Enable   |
| 16            | 1                                       | 0: Disable   |

|                          |                                |   |
|--------------------------|--------------------------------|---|
| EAN 13                   | Read                           | 1: Enable *   |
|                          | 3<br>Transmit Check Digit      | 0: Disable<br>1: Enable *   |
|                          | 8<br>Code ID setting           | 0: Disable *<br>0x01 ~ 0x7F ASCII code(1 bytes)                   |
|                          | 12<br>ISBN conversion          | 0: Disable *<br>1: Enable   |
|                          | 13<br>ISBN Supplement Required | 0: Disable *<br>1: Enable   |
|                          | 14<br>ISMN conversion          | 0: Disable *<br>1: Enable   |
|                          | 15<br>ISMN Supplement Required | 0: Disable *<br>1: Enable   |
| 17<br>Industrial 2 of 5  | 1<br>Read                      | 0: Disable *<br>1: Enable   |
|                          | 8<br>Code ID setting           | 0: Disable *<br>0x01 ~ 0x7F ASCII code(1 bytes)                   |
| 18<br>Interleaved 2 of 5 | 1<br>Read                      | 0: Disable<br>1: Enable *   |
|                          | 2<br>Check Digit               | 0: Disable *<br>1: Enable   |
|                          | 3<br>Transmit Check Digit      | 0: Disable *<br>1: Enable   |
|                          | 8<br>Code ID setting           | 0: Disable *<br>0x01 ~ 0x7F ASCII code(1 bytes)                   |
| 20<br>MSI Plessey        | 1<br>Read                      | 0: Disable *<br>1: Enable   |
|                          | 2<br>Check Digit               | 0: Disable<br>1: One digit (MOD 10) *<br>2: Two digit (MOD 10/10) |
|                          | 3<br>Transmit Check Digit      | 0: Disable *<br>1: Enable   |
|                          | 8<br>Code ID setting           | 0: Disable *<br>0x01 ~ 0x7F ASCII code(1 bytes)                   |
| 21<br>UK Plessey         | 1<br>Read                      | 0: Disable *<br>1: Enable   |
|                          | 3<br>Transmit Check Digit      | 0: Disable *<br>1: Enable   |

|                 |                              |  |
|-----------------|------------------------------|--|
|                 | 8<br>Code ID setting         | 0: Disable*<br>0x01~0x7F ASCII code(1 bytes) |
| 22<br>Telepen   | 1<br>Read                    | 0: Disable*<br>1: Enable                     |
|                 | 8<br>Code ID setting         | 0: Disable*<br>0x01~0x7F ASCII code(1 bytes) |
|                 | 10<br>Full ASCII             | 0: Disable*<br>1: Enable                     |
| 23<br>UPCA      | 1<br>Read                    | 0: Disable<br>1: Enable*                     |
|                 | 3<br>Transmit Check Digit    | 0: Disable<br>1: Enable*                     |
|                 | 8<br>Code ID setting         | 0: Disable*<br>0x01~0x7F ASCII code(1 bytes) |
|                 | 12<br>Convert to EAN-13      | 0: Disable*<br>1: Enable                     |
|                 | 13<br>Coupon                 | 0: Disable*<br>1: Enable See Note 4          |
|                 | 14<br>Coupon Transmit "JC1"  | 0: Disable<br>1: Enable*                     |
|                 | 15<br>Transmit Number System | 0: Disable<br>1: Enable*                     |
| 24<br>UPCE      | 1<br>Read                    | 0: Disable<br>1: Enable*                     |
|                 | 3<br>Transmit Check Digit    | 0: Disable<br>1: Enable*                     |
|                 | 8<br>Code ID setting         | 0: Disable*<br>0x01~0x7F ASCII code(1 bytes) |
|                 | 14<br>Convert to UPC-A       | 0: Disable*<br>1: Enable                     |
|                 | 15<br>Transmit Number System | 0: Disable<br>1: Enable*                     |
| 25<br>Matrix 25 | 1<br>Read                    | 0: Disable*<br>1: Enable                     |
|                 | 2<br>Check Digit             | 0: Disable*<br>1: Enable                     |
|                 | 3<br>Transmit Check Digit    | 0: Disable*<br>1: Enable                     |

|                       |                                  |  |
|-----------------------|----------------------------------|--|
|                       | 8<br>Code ID setting             | 0: Disable*<br>0x01~0x7F ASCII code(1 bytes) |
| 35<br>UPC/EAN General | 1<br>Supplements Required        | 0: Disable*<br>1: Enable                     |
|                       | 2<br>Two Digit Supplements       | 0: Disable*<br>1: Enable                     |
|                       | 3<br>Five Digit Supplements      | 0: Disable*<br>1: Enable                     |
|                       | 4<br>Two Digit Redundancy        | 0: Disable*<br>1: Enable                     |
|                       | 5<br>Five Digit Redundancy       | 0: Disable*<br>1: Enable                     |
|                       | 6<br>GTIN Formatting             | 0: Disable*<br>1: Enable                     |
| 36<br>IATA2 of 5      | 1<br>Read                        | 0: Disable*<br>1: Enable See Note 5          |
|                       | 8<br>Code ID setting             | 0: Disable*<br>0x01~0x7F ASCII code(1 bytes) |
| 37<br>TRI-OPTIC       | 1<br>Read                        | 0: Disable*<br>1: Enable                     |
|                       | 8<br>Code ID setting             | 0: Disable*<br>0x01~0x7F ASCII code(1 bytes) |
|                       | 10<br>Conversion                 | 0: Disable*<br>1: Enable                     |
| 38<br>RSS             | 1<br>Read                        | 0: Disable*<br>1: Enable                     |
|                       | 2<br>Transmit Application ID     | 0: Disable<br>1: Enable*                     |
|                       | 3<br>Transmit Check Digit        | 0: Disable<br>1: Enable*                     |
|                       | 4<br>Transmit Symbology ID "je0" | 0: Disable*<br>1: Enable                     |

---

---

**Note1:**

This command will set all barcode ID settings to proprietary value. It can't be queried.

| Barcode            | value | Barcode     | value |
|--------------------|-------|-------------|-------|
| Code 11            | 100   | MSI Plessey | 110   |
| Code 39            | 101   | UK Plessey  | 111   |
| Code 93            | 102   | Telepen     | 112   |
| Code 128           | 103   | UPCA        | 113   |
| Codabar            | 104   | UPCE        | 114   |
| EAN8               | 105   | Matrix 25   | 115   |
| EAN13              | 106   | IATA 25     | 125   |
| Industrial 2 of 5  | 107   | Trioptic    | 126   |
| Interleaved 2 of 5 | 108   |             |       |

**Note 2:**

There are seven barcode length-lock available. Specific barcode type can be assigned with a length-lock.

Code type:

|                | Hex Value |
|----------------|-----------|
| CODE11         | 64        |
| CODE39         | 65        |
| CODE93         | 66        |
| CODE128        | 67        |
| CODABAR        | 68        |
| INDUSTRIAL_25  | 6B        |
| INTERLEAVED_25 | 6C        |
| MSI_PLESSEY    | 6E        |
| UK_PLESSEY     | 6F        |
| TELEPEN        | 70        |
| MATRIX_25      | 73        |
| TRIOPTIC       | 7E        |

Length:

0-80 (0: Disable)

Default setting:

0000

---

---

Example:

| Cmd1 | Cmd2 | Value |
|------|------|-------|
| 7    | 10   | 670C  |
| 7    | 11   | 6514  |
| 7    | 12   | 6710  |

Code 128: Can only read barcode of 12- or 16-digit.

Code 39: Can only read barcode of 20-digit.

**Note 3:**

ScanTimeout: the maximum time, in seconds, allowed during which the scanning beam remains ON without decoding any barcode.

IdleTimeout: the maximum time, in seconds, allowed during which the scanning device remains idle without any action.

**Note 4:**

When this setting is Enabled, the UCC/EAN 128 barcode can not be read.

**Note 5:**

IATA2 of 5 only support 13- or 15-digit.

---

---

## 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.                     |