日晶科技股份有限公司 / Sunlit System Technology Corp.
μ-Chip System Development

RFID CF Type Reader User's Guide

Model Name: SLC-11100

Version: 1.0

Date: 2007.04.09

Content

Notice:

In order to avoid misuse or any unexpected damage, please read this guide first.

This device complies with Part 15 of FCC Rules. Operation is subject to the following two conditions:
(1)This device may not cause harmful interference, and
(2)This device must accept any interference received, including interference that may cause undesired operation

1. Hardware environment

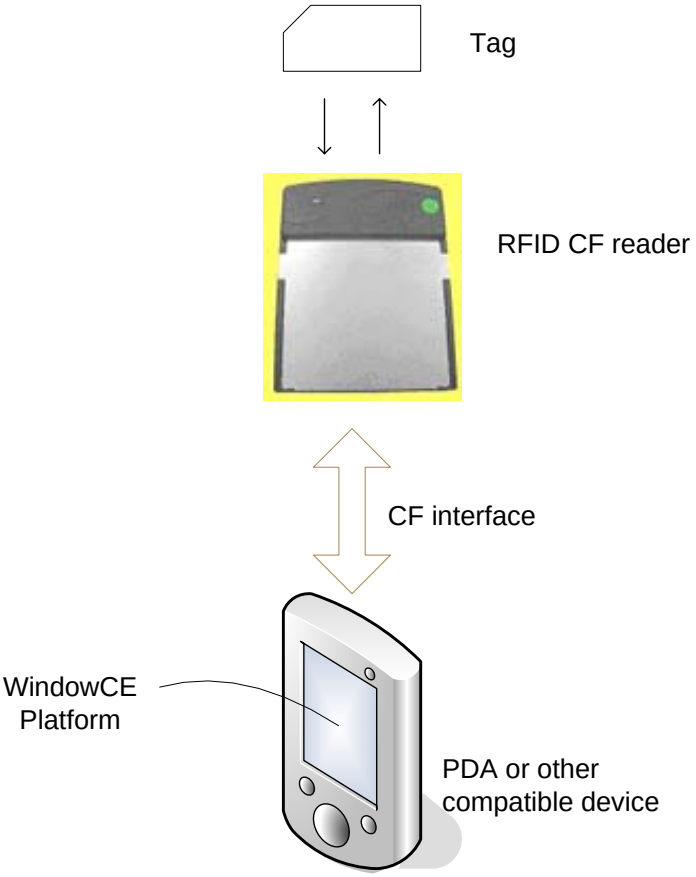
(1) Product introduction

Sunlit's RFID CF type reader is based on **Hitachi μ-solution** to develop.
It was operated at microwave 2.45GHz frequency band.
This is small and light, can be easily install to PDA, tablet PC.

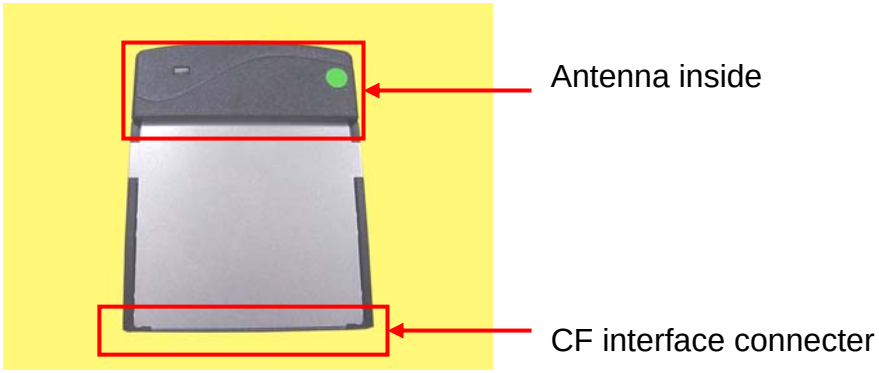
(2) Specification

Power Supply	DC 3V from PDA or PC
Operating Environment	0˚C ~40˚C
Storage Environment	0˚C ~60˚C
Power Consumption	Max. 0.6 W
Dimensions	56 x 43 x 6.1 (mm x mm x mm)
RF Output Power	100 mW (20 dBm)
Frequency Range	2.402GHZ~2.477GHZ
Baud Rate	4800 b/s
Reading Distance	About 5 cm

(3) Hardware structure



(4) Hardware appearance introduction



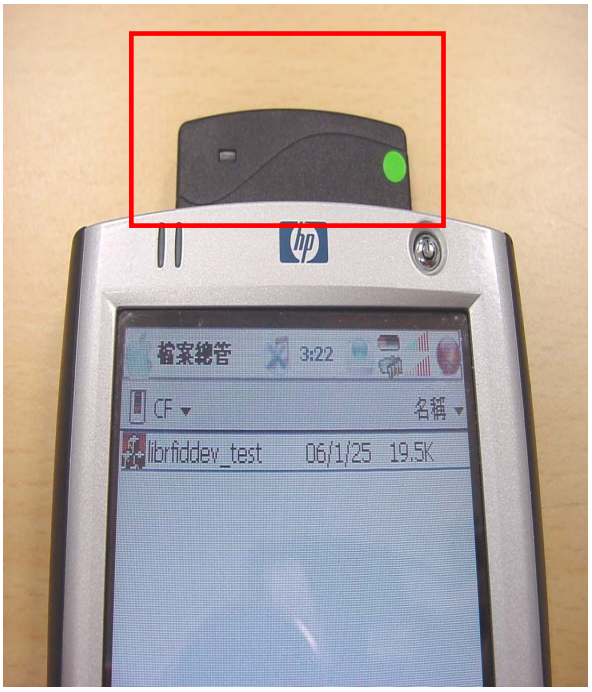
(5) System requirement

Item	Condition	Quantity
Hardware requirement	PDA support CF interface	1
Platform requirement	Windows Pocket PC 2003	1

(6) How to install reader

Face the side with LED up, and insert it into PDA CF slot as shown in picture.

Normally always insert CF reader by put LED side up, but some PDA put CF slot at reverse side, it means user needs to insert CF reader at back side. (LED down)

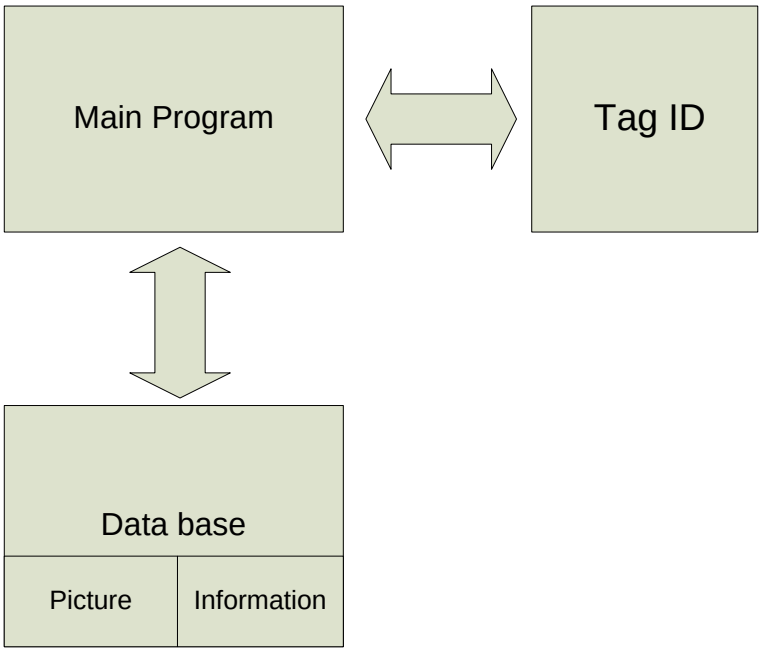


2. Demo software environment

(1) Demo software introduction

This demo program can build a database include TAG ID、TAG information and picture. It can be use to demo ex. material manager、In/out control for people… etc.

(2) Block diagram



(3) Demo program operation

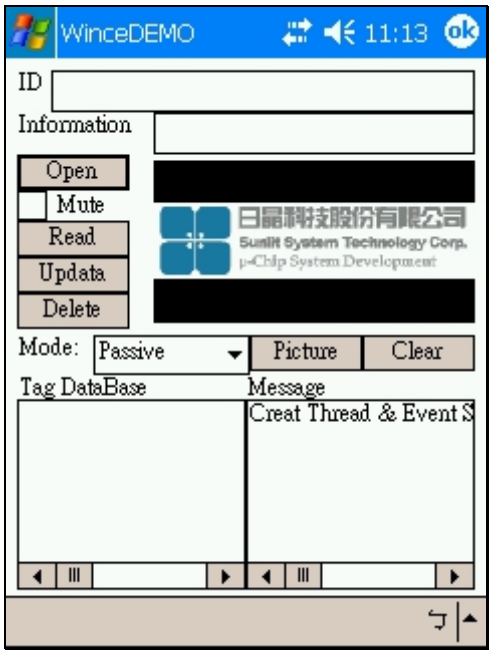
System requirement

Item	Condition	Quantity
PDA	Support CF interface 64MB Flash ROM 64 MB RAM	1
System platform	Windows Pocket PC 2003	1
RFID CF reader		1
Tag	Include Hitachi μ-chip inlet	1

Demo software contents

Item	Description	
Development tool	Microsoft Visual Studio 2005 Traditional Edition	
File contents	WinceDEMO.exe	Main program
	sunlitrfidppc.dll	Dynamic Link Library file
	Database.txt	ID database file after ID saved
	PIC [Folder]	A picture file includes different pictures which correlated with each individual ID

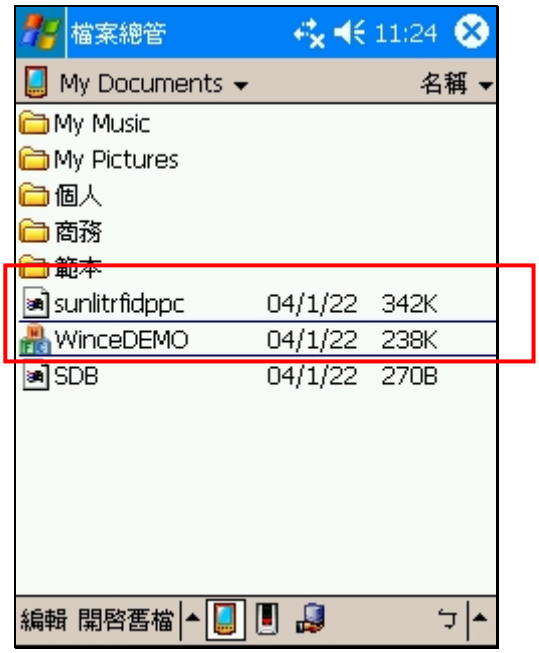
Main Program Window



Function	Description	Function	Description
ID	Display Tag ID when Tag be read.	**Delete**	Delete registered database
Information	Edit information about Tag ID	**Mode**	Select scan mode
Open	Open/Close comport of reader	**Picture**	Select picture correct with ID
Mute	ON/Off reading sound	**Clear**	Clear text of ID & Message window
Read	Trigger reader to scan	**Tag DataBase**	Display registered database
Updata	Save registered database	**Message**	Display status of reader

Demo program operation

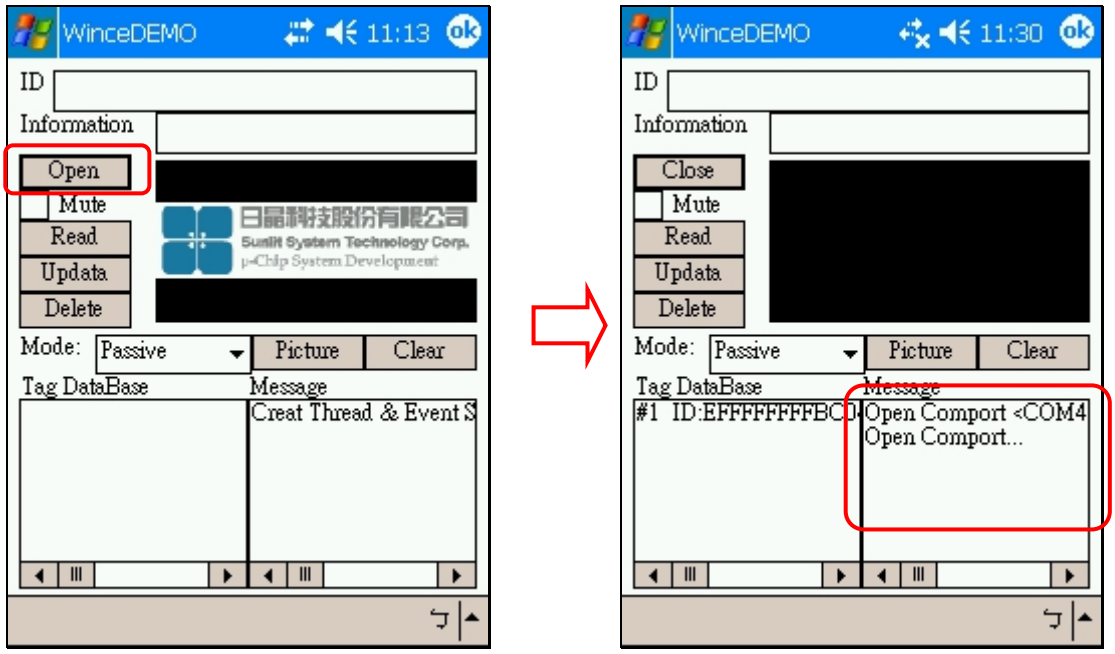
(a) Copy demo program files “**WinceDEMO.exe**” & “**sunlitrfidppc.dll**” to PDA and store at same directory.



※About the file transmissions please refer to PDA user’s manual.

(b) Plug reader in CF slot of PDA

(c) Click file "**WinceDEMO.exe**" and click "**Open**" to open device(Reader)



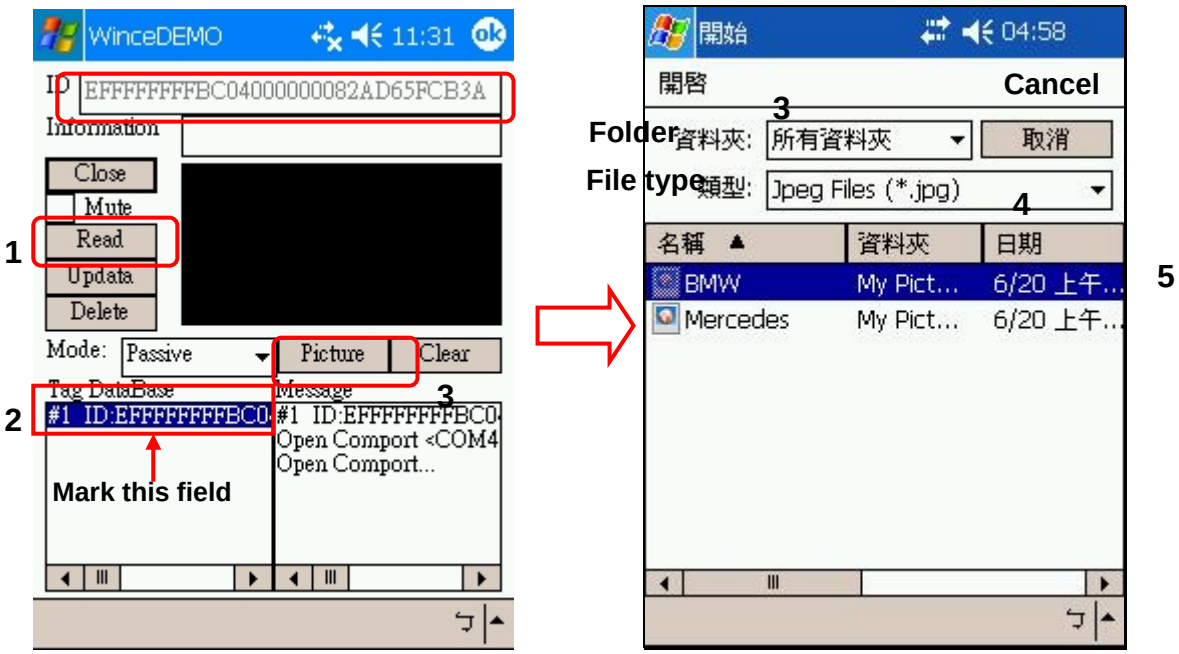
It mean open device successful

(d) Create a ID data base

Be sure the Mode is "**Passive**"

Put the tag on front side of reader, click "**Read**" to scan Tag ID and checking ID number is showing on **ID** window. Mark the ID number in **Tag DataBase** field .

Click "**Picture**"→ "**Folder**" to open location of picture that you want to correct with Tag ID.Click "**Type**" to select file type that you want than click picture file

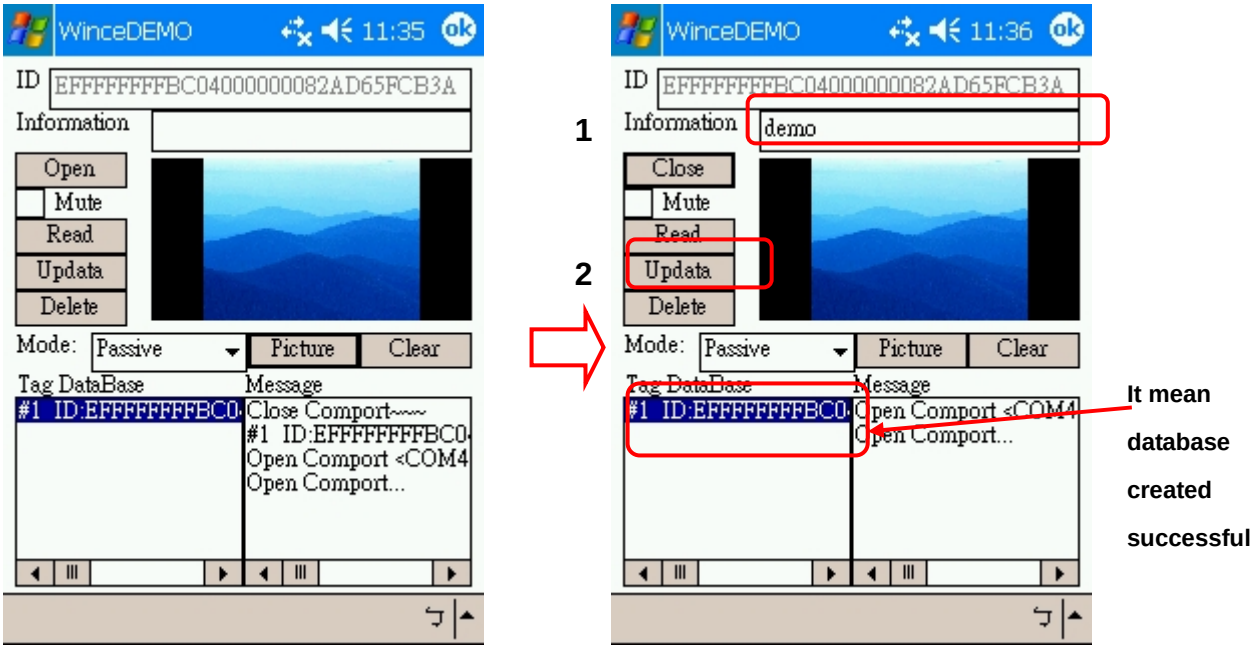


※ **Operation mode**:

Passive: Click **Read** once and reader will scan once.

Active: Click **Read** once and reader will scan continuously.

(e) Editing information about the ID number on “Information” window.
Click “**Updata**” and check “**Tag DataBase**” window to complete the data base created procedure.

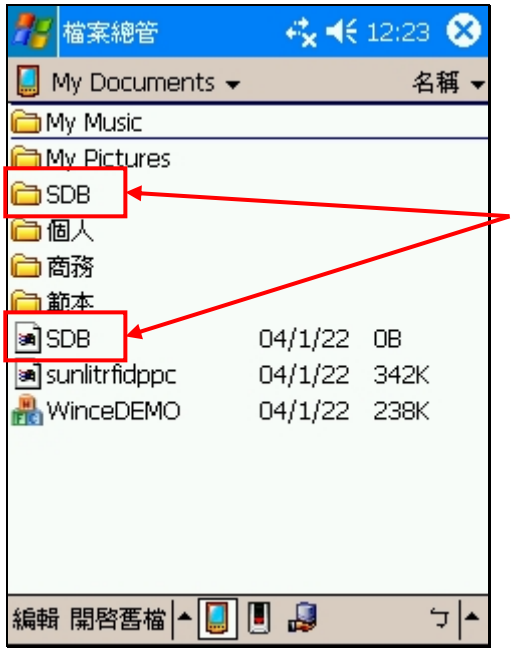


(f) You can follow the step (d) (e) to create some data base that you want.

(g) When you created the data base, the system was created 1 folder and 1 text file:

Item	Description
SDB [text file]	ID database file after ID saved
SDB [Folder]	A picture file storage folder about correlate with ID database after picture saved.

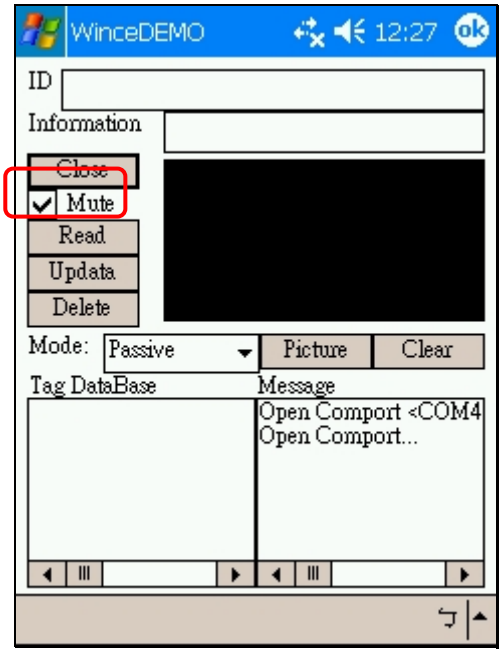
Caution: Every PDA are different resolution so the picture is showed different size. If you want it to be matched, you should modify the source code of the project. And showing picture is Windows API,but it is not stable. Showing picture seveval times may cause the Demo AP crashes.



When you created the data base, the system was created 1 folder and 1 text file

(h) Set Mute function

Follow the figure click the" Mute" function check box, the mute function will enable.

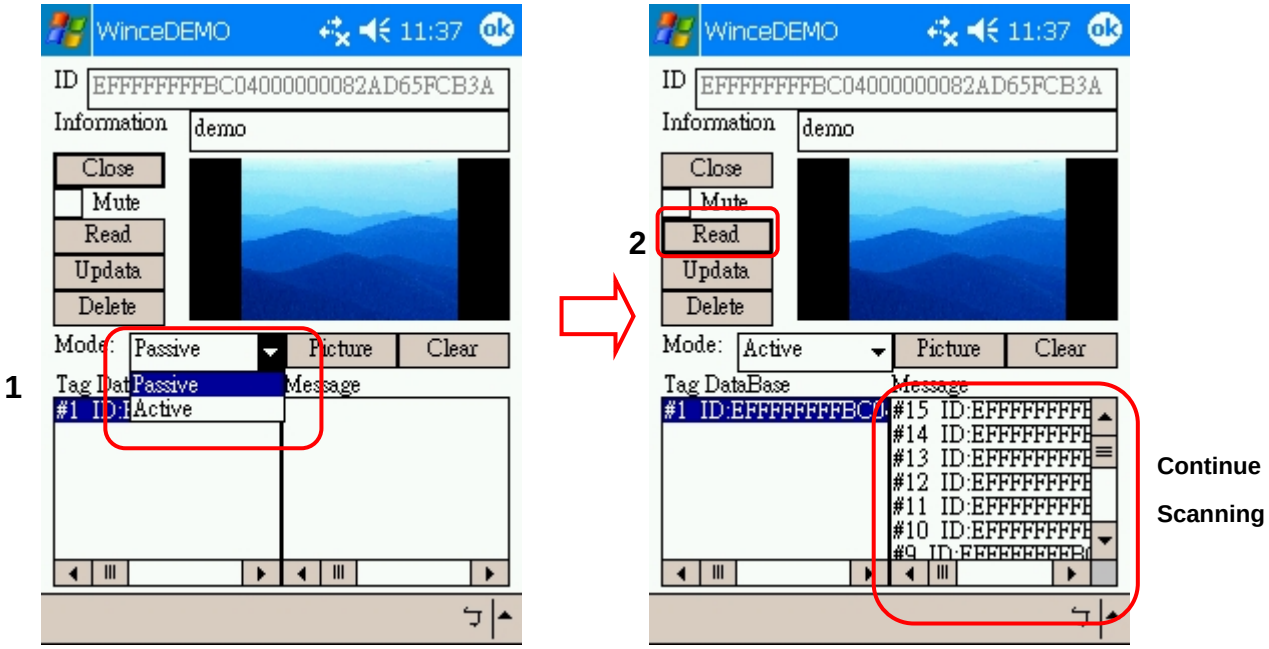


(i) Set Operation mode

Mode: **Active**

Click the "**Mode**" combo box and select "**Active**" item to enable active function

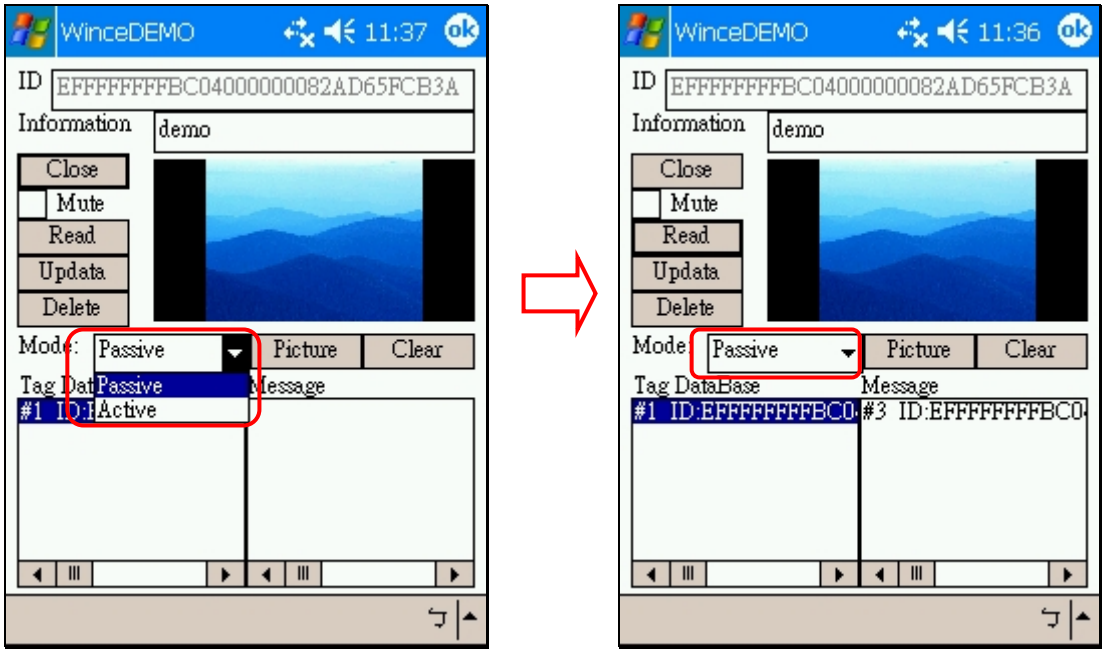
Click "**Read**", if tag scanned by reader and the scan procedures will continuously.



Mode: **Passive**

Click "**Mode**" combo box select "**Passive**" item to enable passive function

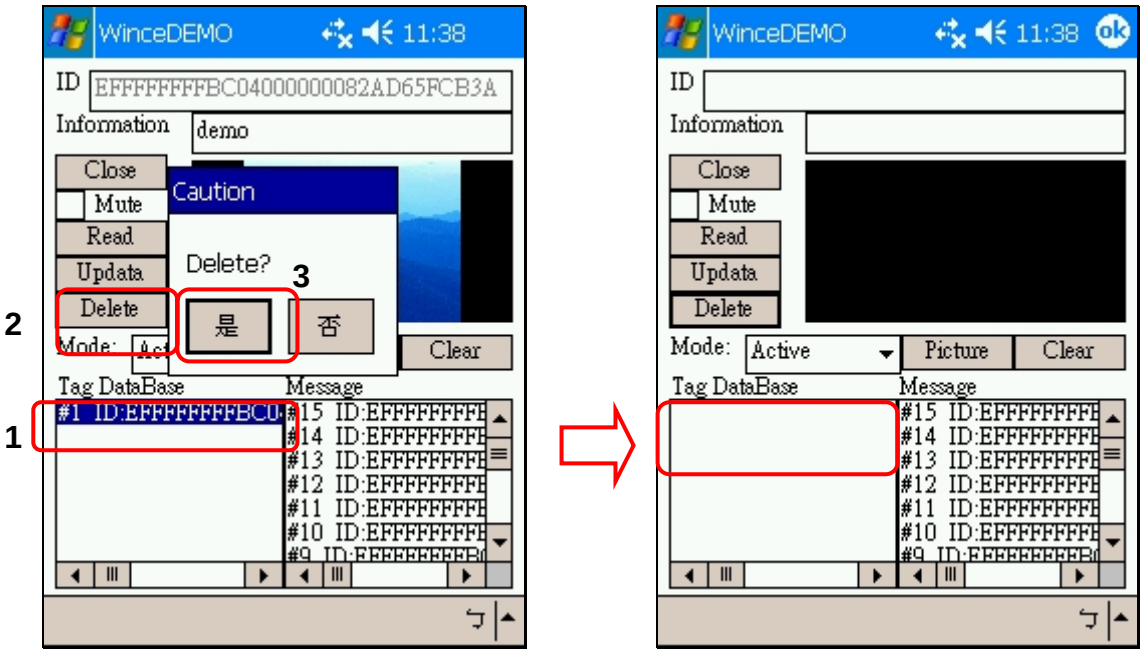
Click "**Read**" and the reader scanning just only once.



(j) Delete data base

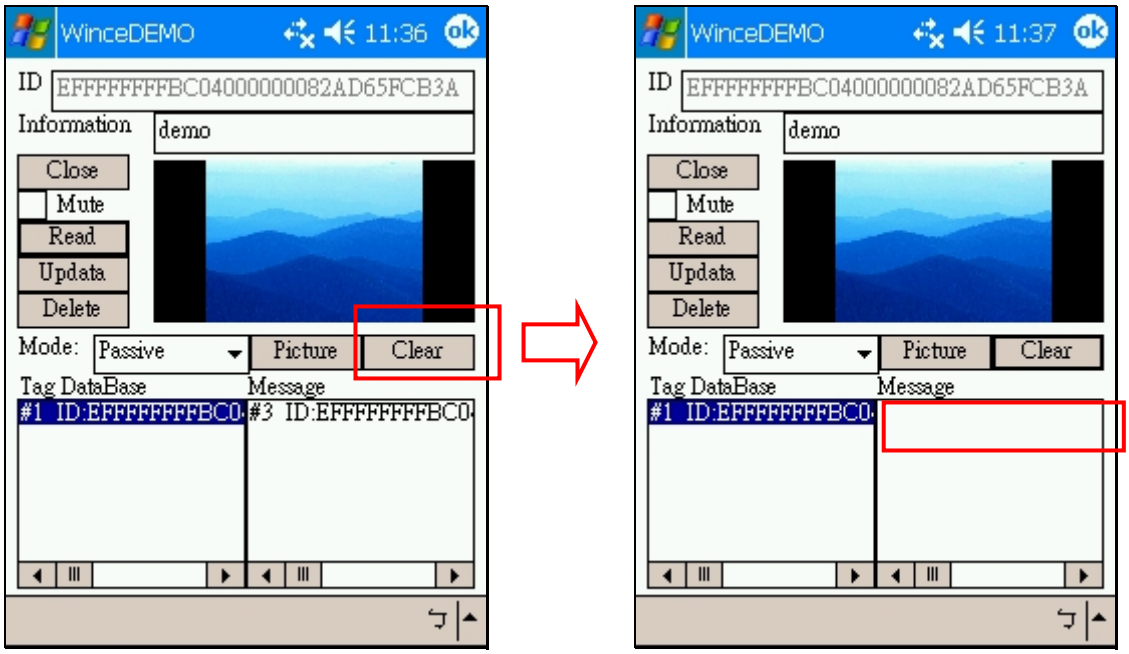
Click data you want to delete on **Tag DataBase** window

Click "**Delete**" and select "**Yes**" to delete data.



(k) Clear function

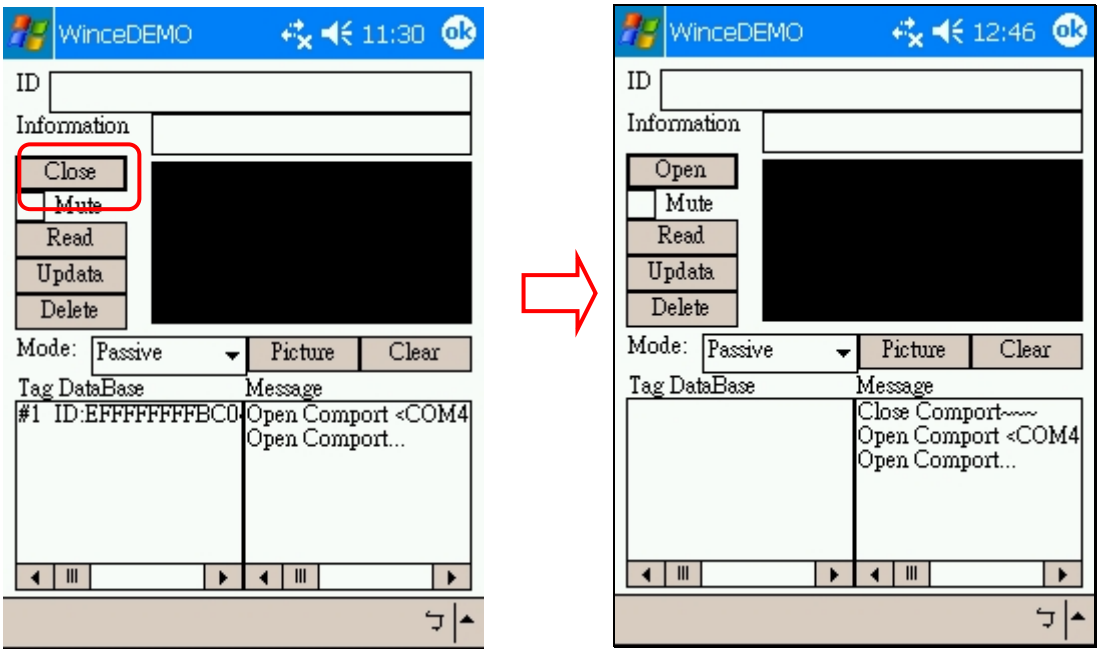
If click "Clear" than the text message on the **Message** window will be clear.



(l) Disable device(Reader)

To confirm device is open.

Click “**Close**” to disable device.



3. API user's guide

ONE：Structure of Sunlit RFID DLL for PPC

TWO：Program Declaration

THREE：FUNCTION Introduction

FOUR：EVENTS Introduction

APPENDIX：Development Environment

ONE、Structure of Sunlit RFID DLL for PPC

The Sunlit RFID DLL has 18 functions. They can separate into 3 groups：Comport operation、Device operation and Device information：

Comport operation functions includes following 5 functions：

SUNLITRFID_CFExist

SUNLITRFID_Open

SUNLITRFID_OpenII

SUNLITRFID_IsOpen

SUNLITRFID_Close.

Most functions of SUNLITRFID_Open and SUNLITRFID_OpenII are the same, except using different parameters.

Device operation functions includes following 9 functions：

SUNLITRFID_Echo

SUNLITRFID_SoftwareReset

SUNLITRFID_HardwareReset ***〈Depend on device〉***

SUNLITRFID_Pause

SUNLITRFID_OpModeSet

SUNLITRFID_OpModeGet

SUNLITRFID_ActiveScanIntervalSet

SUNLITRFID_ActiveScanIntervalGet

SUNLITRFID_ScanTag

Device information functions includes following 4 functions：

SUNLITRFID_ProductNameGet

SUNLITRFID_ModelNameGet

SUNLITRFID_HardwareVersionGet

SUNLITRFID_FirmwareVersionGet

The following table lists all DLL FUNCTION and FUNCTION parameter. For more information, Please see FUNCTION Introduction.

Comport operation functions
SUNLITRFID_CFExist(TCHAR *CFName)
SUNLITRFID_Open(HWND hWnd,LPCWSTR PortName, DWORD BaudRate,SUNLITRFID_ENVIRONMENT *Environment)
SUNLITRFID_OpenII(HWND hWnd,LPCWSTR PortName, DWORD BaudRate,HANDLE *haEvent,SUNLITRFID_TAGID *ActiveTagID)
SUNLITRFID_IsOpen(void)
SUNLITRFID_Close(void)
Device operation functions
SUNLITRFID_Echo(void)
SUNLITRFID_SoftwareReset(void)
SUNLITRFID_HardwareReset(void) ***《Depend on device》***
SUNLITRFID_Pause(void)
SUNLITRFID_OpModeSet(BYTE Data)
SUNLITRFID_OpModeGet(BYTE *Data)
SUNLITRFID_ActiveScanIntervalSet(BYTE Data)
SUNLITRFID_ActiveScanIntervalGet(BYTE *Data)
SUNLITRFID_ScanTag(SUNLITRFID_TAGID *TagID)
Device information functions
SUNLITRFID_ProductNameGet(SUNLITRFID_PRODUCT_NAME *ProductName)
SUNLITRFID_ModelNameGet(SUNLITRFID_MODEL_NAME *ModelName)
SUNLITRFID_HardwareVersionGet(SUNLITRFID_HARDWARE_VER *HardwareVer)
SUNLITRFID_FirmwareVersionGet(SUNLITRFID_FIRMWARE_VER *FirmwareVer)

Currently, there are only 3 different EVENTS in Sunlit RFID DLL：

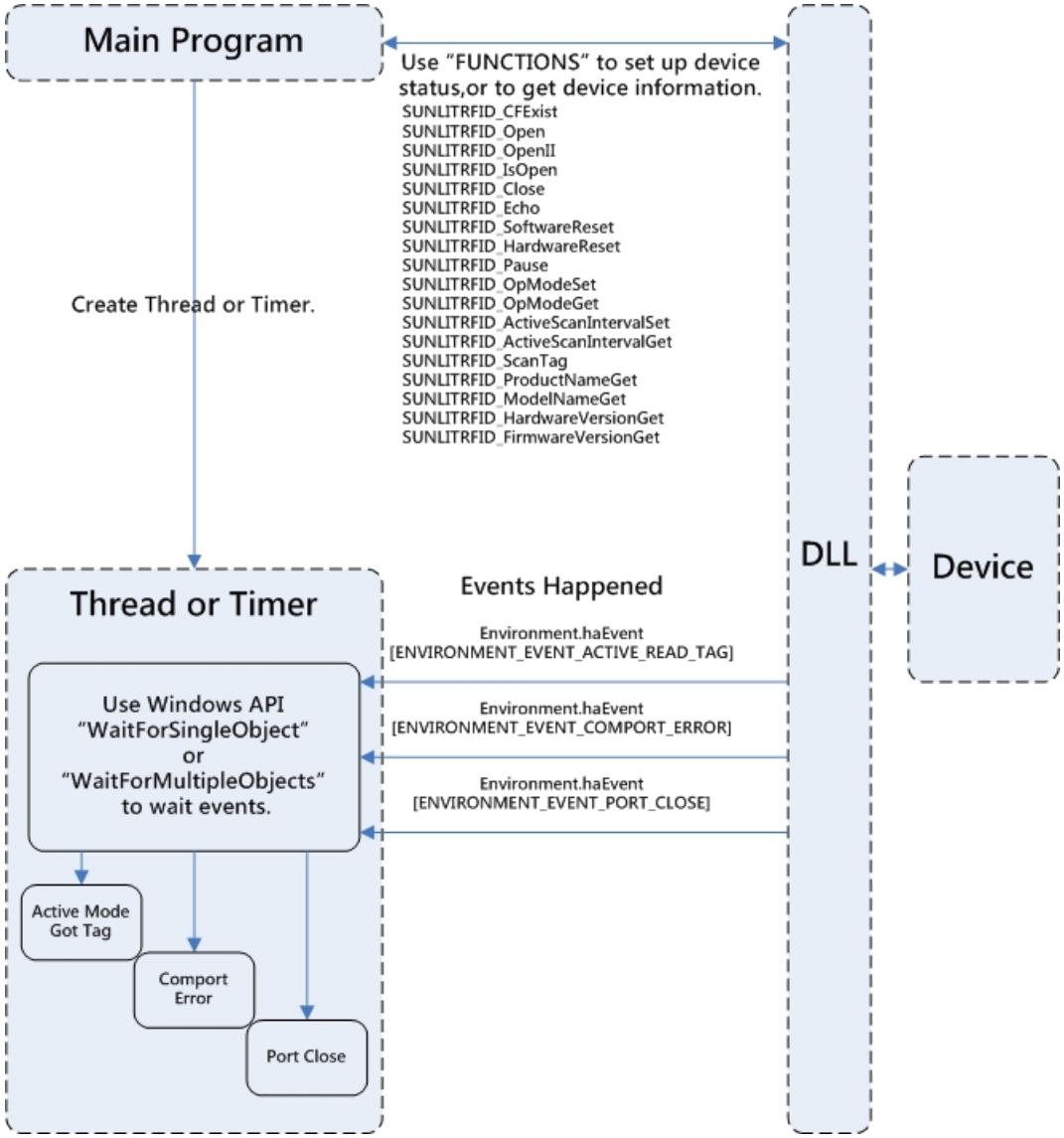
Environment.haEvent[ENVIRONMENT_EVENT_ACTIVE_READ_TAG]

Environment.haEvent[ENVIRONMENT_EVENT_COMPORT_ERROR]

Environment.haEvent[ENVIRONMENT_EVENT_PORT_CLOSE]

For more information, Please see EVENTS Introduction.

The following picture represents how the DLL works with your program.



TWO、Program Declaration

(1) Constant Definition

The following table lists all constant definition that TYPE definition used.

Definition		Introduction
#define PRODUCT_NAME_LENGTH	32	Define the length of product name information. (Unit：byte)
#define MODEL_NAME_LENGTH	16	Define the length of model name information. (Unit：byte)
#define HARDWARE_LENGTH	8	Define the length of hardware version information. (Unit：byte)
#define FIRMWARE_LENGTH	8	Define the length of firmware version information. (Unit：byte)
#define TAG_LENGTH	16	Define the length of Tag ID data. (Unit : byte)
#define ENVIRONMENT_EVENT_NUMBER	3	Define the number of events happened.
#define ENVIRONMENT_EVENT_COMPORT_ERROR	0	For more information, Please see EVENTS Introduction.
#define ENVIRONMENT_EVENT_PORT_CLOSE	1	For more information, Please see EVENTS Introduction.
#define ENVIRONMENT_EVENT_ACTIVE_READ_TAG	2	For more information, Please see EVENTS Introduction.

(2) TYPE Definition

The following table lists all TYPE definition that function used.

Definition	Introduction
typedef struct { **unsigned char data[TAG_LENGTH];** **} SUNLITRFID_TAGID;**	Type definition of tag ID Since Mu-ID data is represented by 128 bits format, so we condense this 128 bit information by byte format. And every 8 bits data is respectively each byte, for example, data[0] represents bit120~bit127; data[1] represents bit112~bit119; data[TAG_LENGTH-1] represents bit0~bit7.
typedef struct { **HANDLE haEvent[ENVIRONMENT_EVENT_NUMBER];** **SUNLITRFID_TAGID ActiveTagID;** **} SUNLITRFID_ENVIRONMENT;**	Type definition of enviroment status. The type has two elements: haEvent and ActiveTagID. haEvent is HANDLE array that use for wait events. For more information, Please see EVENTS Introduction. ActiveTagID was introduced at above definition.
typedef struct { **char data[PRODUCT_NAME _LENGTH+1];** **} SUNLITRFID_PRODUCT_NAME;**	Type definition of product name information. After function executed successfully. The char array will be added the end of string character '0x00'
typedef struct { **char data[MODEL_NAME_LENGTH+1];** **} SUNLITRFID_MODEL_NAME;**	Type definition of Model Name After function executed successfully. The char array will be added the end of string character '0x00'
typedef struct { **char data[HARDWARE_LENGTH+1];** **} SUNLITRFID_HARDWARE_VER;**	Type definition of hardware version After function executed successfully. The char array will be added the end of string character '0x00'
typedef struct { **char data[FIRMWARE_LENGTH+1];** **} SUNLITRFID_FIRMWARE_VER;**	Type definition of firmware version After function executed successfully. The char array will be added the end of string character '0x00'

(3) MESSAGE Definition

The following table lists all DLL function message and how to solve it.

No.	Name	Value	Introduction
1	HANDLE_SUCCESS	0x00	This message will happen when the function you called was executed successfully.
2	HANDLE_ERR_TAG_PASSIVE	0x20	This message will happen when you call “SUNLITRFID_ScanTag” in passive mode,and device found a tag.
3	HANDLE_ERR_TAG_NOT_FOUND	0x22	This message will happen when you call “SUNLITRFID_ScanTag” in passive mode,but device does’t found a tag.
4	HANDLE_ERR_NO_RESPONSE	0x80	This message will happen when you call function,but device has no response. To solve this problem, please check hardware power or connection.
5	HANDLE_ERR_COMPORT	0x81	This message will happen when you call “SUNLITRFID_ScanTag” in passive mode,but comport failed with some error. To solve this problem, please check hardware power or connection.
6	HANDLE_ERR_CREATE_EVENT	0x88	This message will happen when you call “SUNLITRFID_Open” or “SUNLITRFID_OpenII”, but DLL create event fail. To solve this problem, please try again.
7	HANDLE_ERR_CREATE_THREAD	0x89	This message will happen when you call “SUNLITRFID_Open” or “SUNLITRFID_OpenII”, but DLL create thread fail. To solve this problem, please try again.
8	HANDLE_ERR_ILLEGAL_RETURN_DATA	0x8b	This message will happen when you call function,but device return incorrect data. To solve this problem, please try again the function you called.
9	HANDLE_ERR_UNSUPPORTED_FUNCTION	0x8c	This message will happen when device does’t support this function.
10	HANDLE_ERR_ILLEGAL_DATA	0xfd	This message will happen when the parameter of function that you called was illegal. To solve this problem, please send legal data to function.

11	HANDLE_ERR_UNKNOW	0xff	Undefined error. Please record how this message happened and contact with us 〈www.sunlitcorp.com〉.
12	Others	?	Unknow message. Please record how this message happened and contact with us 〈www.sunlitcorp.com〉.

THREE、FUNCTION Introduction

1. bool SUNLITRFID_CFExist (

TCHAR *CFName

);

SUNLITRFID_ComportNumGet is used to find the CF card in your PPC.

Parameters：

CFName：The parameter is shown as TCHAR array format, when this function is executed successfully, the array will show each COMPORT name that inserted CF card. For example, your PPC has 2 different CF card, insert into COM 1 and COM 4, then CFName will show as “COM1\0x00COM4\0x00”.

Return Values：

We need to judge whether we find COMPORT or not, so we define Return Value “TRUE” for successfully find, “False” for failed.

For example：

```
TCHAR CFName[1024];

if(SUNLITRFID_CFExist(CFName))
{}//Found CF card successfully
else    //Not found CF card
{}
```

2. BYTE SUNLITRFID_Open (

HWND hWnd,

LPCWSTR PortName,

DWORD BaudRate,

SUNLITRFID_ENVIRONMENT *Environment

);

SUNLITRFID_Open is used to open the comport

Parameters：

hWnd：The program's HANDLE as a parameter that send thru the Function. If you don't know what's your program's HANDLE, use NULL.

PortName：The comport name you want to open.

BaudRate：The comport baud rate that you want to open with.

Environment：Please see 「TYPE」 definition.

Return Values：

The function will return byte value that indicate the execution result.Please see 「MESSAGE」 definition for byte value.

For example：

```
SUNLITRFID_ENVIRONMENT Environment;
WCHAR wcaPortName[]={'C','O','M','1',0x00};

if(SUNLITRFID_Open(NULL,wcaPortName,19200,&Environment)==HANDLE_SUCCESS)
{}//Open comport successfully
else  //Open comport failed
{}
```

3. BYTE SUNLITRFID_OpenII(

HWND hWnd,

LPCWSTR PortName,

DWORD BaudRate,

HANDLE *haEvent,

SUNLITRFID_TAGID *ActiveTagID

);

SUNLITRFID_OpenII is used to open the comport.

Parameters：

hWnd：The program's HANDLE as a parameter that send thru the Function. If you don't know what's your program's HANDLE, use NULL.

PortName：The comport name you want to open.

BaudRate：The comport baud rate that you want to open with.

haEvent：Please see 「TYPE」 definition.

ActiveTagID：Please see 「TYPE」 definition.

Return Values：

The function will return byte value that indicate the execution result.Please see 「MESSAGE」 definition for byte value.

For example：

```
HANDLE haEvent[ENVIRONMENT_EVENT_NUMBER];
SUNLITRFID_TAGID ActiveTagID;
WCHAR wcaPortName[]={'C','O','M','1',0x00};

if(SUNLITRFID_Open(NULL,wcaPortName,19200,haEvent, &ActiveTagID)==HANDLE_SUCCESS)
{}//Open comport successfully
else//Open comport failed
{}
```

4. bool SUNLITRFID_IsOpen(void);

SUNLITRFID_IsOpen can check the comport status that is opened or not.

Parameters：

No parameter needed.

Return Values：

The function will return a boolean value. If comport was opened,it will return true,otherwise false.

For example：

```
if(SUNLITRFID_IsOpen())
{} //Comport is open
else    //Comport is closed
{}
```

5. bool SUNLITRFID_Close(void);

SUNLITRFID_Close is used for close comport.

Parameters：

No parameter needed.

Return Values：

The function will return a boolean value,If the function executed successfully, it will return true,otherwise false.

For example：

```
if(SUNLITRFID_Close())
{} //Comport was closed successfully
else    //Close comport fail
{}
```

6. BYTE SUNLITRFID_Echo(void);

SUNLITRFID_Echo request device to reply message.

Parameters：

No parameter needed.

Return Values：

The function will return byte value that indicate the execution result.Please see 「MESSAGE」 definition for byte value

For example：

```
if(SUNLITRFID_Echo()==HANDLE_SUCCESS)
{} //The device was reply message successfully
else
{}
```

7. BYTE SUNLITRFID_SoftwareReset(void);

SUNLITRFID_SoftwareReset reset device by software.

Parameters：

No parameter needed.

Return Values：

The function will return byte value that indicate the execution result.Please see 「MESSAGE」 definition for byte value

For example：

```
if(SUNLITRFID_SoftwareReset()==HANDLE_SUCCESS)
{} //The device was reset by software successfully
else
{}
```

8. BYTE SUNLITRFID_HardwareReset(void);

SUNLITRFID_HardwareReset reset device by hardware.〈**Depend on device**〉

Parameters：

No parameter needed.

Return Values：

The function will return byte value that indicate the execution result.Please see 「MESSAGE」 definition for byte value

For example：

```
if(SUNLITRFID_HardwareReset()==HANDLE_SUCCESS)
{} //The device was reset by hardware successfully
else
{}
```

9. BYTE SUNLITRFID_Pause(void);

SUNLITRFID_Pause pauses the device from any operation mode.

Parameters：

No parameter needed.

Return Values：

The function will return byte value that indicate the execution result.Please see 「MESSAGE」 definition for byte value

For example：

```
if(SUNLITRFID_Pause()==HANDLE_SUCCESS)
{} //The device was paused successfully
else
{}
```

10. BYTE SUNLITRFID_OpModeSet(

BYTE Data

);

SUNLITRFID_OpModeSet is used to set dvice's operation mode, "Active Mode" or "Passive Mode".

Parameters：

Data：The value 0x00 represents passive mode,other values represent active mode

Return Values：

The function will return byte value that indicate the execution result, please see 「MESSAGE」 definition for byte value

For example：

```
BYTE byData;
byData=0x00;//passive mode

if(SUNLITRFID_OpModeSet(byData)==HANDLE_SUCCESS)
{} //The device was set in passive mode successfully
else
{}
```

11. BYTE SUNLITRFID_OpModeGet(

BYTE *Data

);

SUNLITRFID_OpModeGet is used to get dvice's operation mode.

Parameters：

Data：After executing the function ,The value 0x00 represents that the device was in passive mode,other values represent that the device was in active mode.

Return Values：

The function will return byte value that indicate the execution result, please see 「MESSAGE」 definition for byte value.

For example：

```
BYTE byData;

if(SUNLITRFID_OpModeGet(&byData)==HANDLE_SUCCESS)
{
    if(byData==0x00){} //The device was in passive mode
    else {} //The device was in active mode

}
else
{}
```

12. BYTE SUNLITRFID_ActiveScanIntervalSet(

BYTE Data

);

SUNLITRFID_ActiveScanIntervalSet is used to set the scan interval time for active mode.

Parameters：

Data：The value represents scan interval time,bwtween 5 ~ 255 (Uint：10mS)

Return Values：

The function will return byte value that indicate the execution result, please see 「MESSAGE」 definition for byte value.

For example：

```
BYTE byData;
byData=10;//scan interval=100ms

if(SUNLITRFID_ ActiveScanIntervalSet(byData)==HANDLE_SUCCESS)
{} //Set scan interval successfully
else
{}
```

13. BYTE SUNLITRFID_ActiveScanIntervalGet(

BYTE *Data

);

SUNLITRFID_ActiveScanIntervalGet is used to get the current scan interval time for active mode.

Parameters：

Data：After executing the function ,The value represents the scan interval time that in active mode.

(Uint：10mS)

Return Values：

The function will return byte value that indicate the execution result, please see 「MESSAGE」 definition for byte value

For example：

```
BYTE byData;

if(SUNLITRFID_ ActiveScanIntervalGet(&byData)==HANDLE_SUCCESS)
{} //Get scan interval successfully
else
{}
```

14. BYTE SUNLITRFID_ScanTag(

SUNLITRFID_TAGID *TagID

);

SUNLITRFID_ScanTag for scan tag. When it passive mode, reader will read tag once, when it is active mode, reader will start to read tag continuously.

Parameters：

TagID：Please see 「TYPE」 definition.

Return Values：

The function will return byte value that indicate the execution result, please see 「MESSAGE」 definition for byte value.

For example：

```
SUNLITRFID_TAGID TagID;

switch(SUNLITRFID_ SUNLITRFID_ScanTag(&TagID))
{
    case HANDLE_SUCCESS: //Start scan tag in active mode
        break;
    case HANDLE_ERR_TAG_PASSIVE://Found Tag in passive mode
        break;
    case HANDLE_ERR_TAG_NOT_FOUND://Not Found Tag in passive mode
        break;
    dfault://Please see「MESSAGE」 definition
        break;
}
```

15. BYTE SUNLITRFID_ProductNameGet(

SUNLITRFID_PRODUCT_NAME *ProductName

);

SUNLITRFID_ProductNameGet is used to get the product name information form the device.

Parameters：

ProductName：Please see 「TYPE」 definition.

Return Values：

The function will return byte value that indicate the execution result, please see 「MESSAGE」 definition for byte value.

For example：

```
SUNLITRFID_PRODUCT_NAME ProductName;

if(SUNLITRFID_ProductNameGet(&ProductName)==HANDLE_SUCCESS)
{} //Get product name information successfully
else
{}
```

16. BYTE SUNLITRFID_ModelNameGet(

SUNLITRFID_MODEL_NAME *ModelName

);

SUNLITRFID_ModelNameGet is used to get the model name information form the device.

Parameters：

ModelName：Please see 「TYPE」 definition.

Return Values：

The function will return byte value that indicate the execution result, please see 「MESSAGE」 definition for byte value.

For example：

```
SUNLITRFID_MODEL_NAME ModelName;

if(SUNLITRFID_ModelNameGet(&ModelName)==HANDLE_SUCCESS)
{} //Get model name information successfully
else
{}
```

17. BYTE SUNLITRFID_HardwareVersionGet(

SUNLITRFID_HARDWARE_VER *HardwareVer

);

SUNLITRFID_HardwareVersionGet is used to get the hardware version from the device.

Parameters：

HardwareVer：Please see 「TYPE」 definition.

Return Values：

The function will return byte value that indicate the execution result, please see 「MESSAGE」 definition for byte value.

For example：

```
SUNLITRFID_HARDWARE_VER HardwareVer;

if(SUNLITRFID_HardwareVersionGet(&HardwareVer)==HANDLE_SUCCESS)
{} //Get hardware version successfully
else
{}
```

18. BYTE SUNLITRFID_FirmwareVersionGet(

SUNLITRFID_FIRMWARE_VER *FirmwareVer

);

SUNLITRFID_FirmwareVersionGet is used to get the firmware version from the device.

Parameters：

FirmwareVer：Please see 「TYPE」 definition.

Return Values：

The function will return byte value that indicate the execution result, please see 「MESSAGE」 definition for byte value.

For example：

```
SUNLITRFID_FIRMWARE_VER FirmwareVer;

if(SUNLITRFID_FirmwareVersionGet(&FirmwareVer)==HANDLE_SUCCESS)
{} //Get firmware version successfully
else
{}
```

FOUR、EVENTS Introduction

1. Environment.haEvent[ENVIRONMENT_EVENT_ACTIVE_READ_TAG]

The event happened when the device read a tag in active mode.

2. Environment.haEvent[ENVIRONMENT_EVENT_COMPORT_ERROR]

The event happened when comport or hardware error that DLL can't communicate with device.

3. Environment.haEvent[ENVIRONMENT_EVENT_PORT_CLOSE]

The event happened when comport closed. The event's purpose is let your thread or timer to escape form the waiting function "WaitForSingleObject" or "WaitForMultipleObjects".

You may need to use Windows API function "WaitForSingleObject" or "WaitForMultipleObjects" to wait events. For more information, Please see Microsoft MSDN Library.

If you use function "WaitForSingleObject" or "WaitForMultipleObjects" to wait events in **timer**. We recommend that the 2nd parameter of function "WaitForSingleObject" or the 4th parameter of function "WaitForMultipleObjects" must be a limited value. For example:50.

Hardware Environment：

Product Name：RFID CF Type Reader V3.0

Model Name：SLC-10200

Firmware Version：V1.0 or later

Hardware Version：V1.0 or later

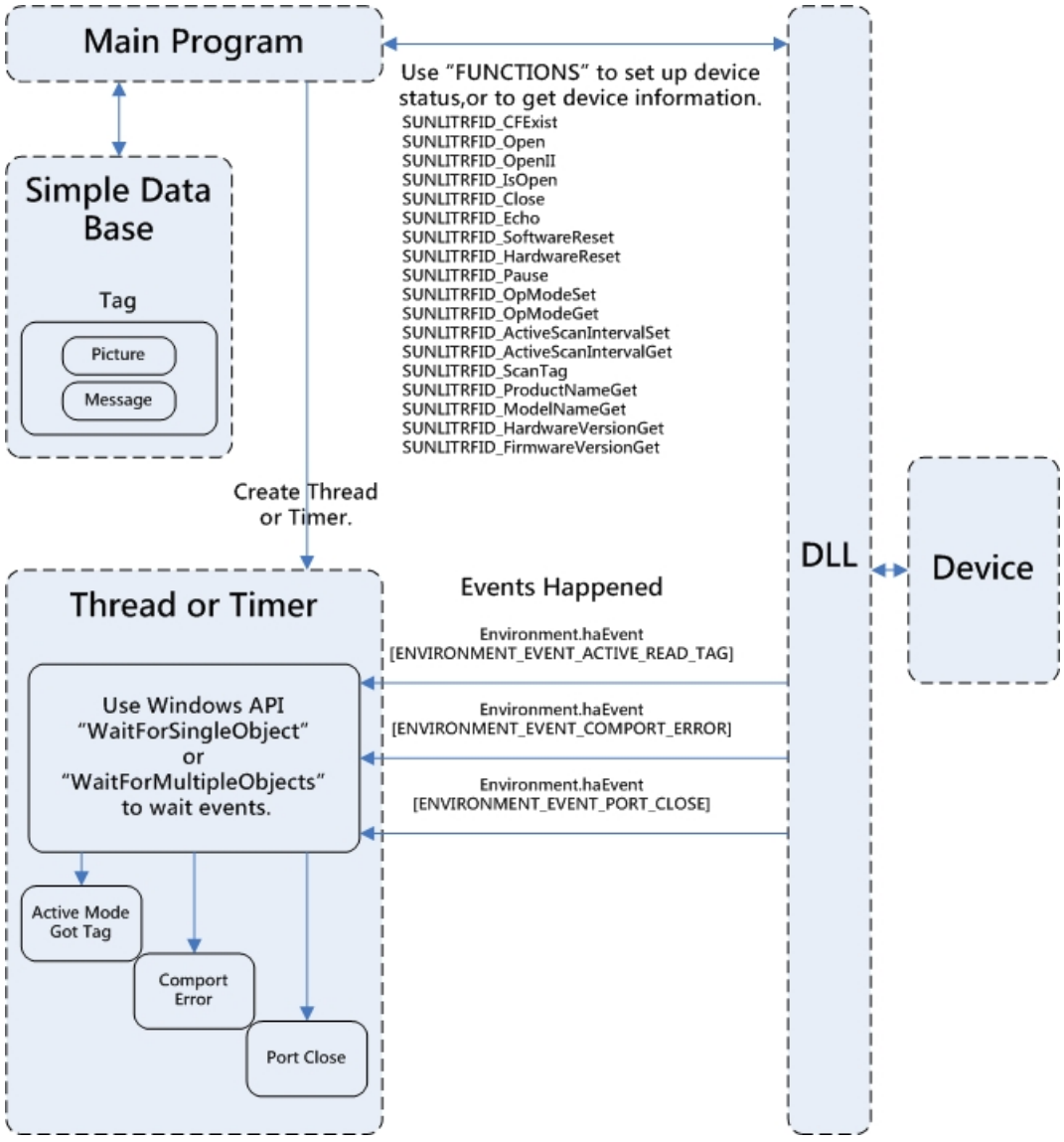
Software Environment：

Development Environment：Microsoft Visual C++ 2005 Professional〈Chinese Traditional〉

4. Demo program source code

ONE、The Main Framework of the Demo Program

The following picture represent**s the main framework of the demo program.**



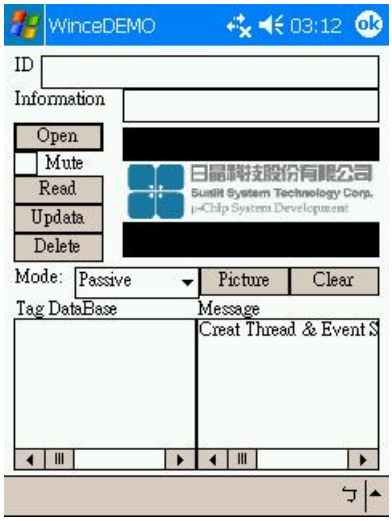
TWO、Program Code Explanation

※There are many subroutines, following are the source code of to subroutines.

The explanation is in accordance with user interface.

(1) Open Comport

Click “Open” button to open comport.



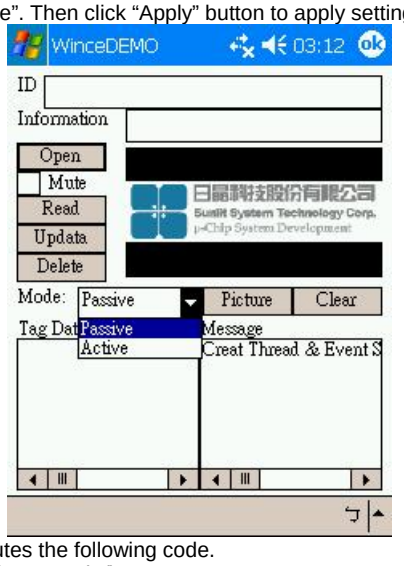
The “Open” button executes the following code.

```
void CWinceDEMODlg::OnBnClickedButtonOpen()
{
	// TODO: 在此加入控制項告知處理常式程式碼
	CString sTemp;
	TCHAR CFName[1024];
	ZeroMemory(CFName, sizeof(CFName));

	ButtonOpen.EnableWindow(false);

	if(!SUNLITRFID_IsOpen())
	{
		ListMsg.ResetContent();

		if(SUNLITRFID_CFExist(CFName))
		{
			ListMsg.InsertString(0,CString("Open Comport..."));
			ListMsg.UpdateWindow ();
```

```
                if(SUNLITRFID_Open(this->m_hWnd ,CFName,4800,&Environment)==HANDLE_SUCCESS)
                {
                        Sleep(10);
                        SUNLITRFID_Pause();

                        ButtonOpen.SetWindowTextW(CString("Close"));
                        DrawEnvironment.ClearPicture();
                        sTemp.Format(CString("Open Comport <%s> Successed!!!"),CFName);
                        ListMsg.InsertString(0,sTemp);

                        if(ButtonMute.GetCheck()==BST_UNCHECKED)
                                MessageBeep(MB_ICONASTERISK);
                }
                else
                {
                        sTemp.Format(CString("Open Comport <%s> Fail..."),CFName);
                        ListMsg.InsertString(0,sTemp);
                }
        }
        else    ListMsg.InsertString(0,CString("Can't Find CF Card..."));
    }
    else
    {
        SUNLITRFID_Close();
        DrawEnvironment.ClearPicture();
        ListMsg.InsertString(0,CString("Close Comport~~~"));
        ButtonOpen.SetWindowTextW(CString("Open"));

        if(ButtonMute.GetCheck()==BST_UNCHECKED)
                MessageBeep(MB_ICONHAND);
        if(ComboBoxOPMode.GetCount()>0)
                ComboBoxOPMode.SetCurSel(0);
    }

SetEvent(hEventFlip);
ListMsg.UpdateWindow();
ButtonOpen.EnableWindow(true);
}
```

(2) Select Operation Mode

"Operation Mode:" selects "passive" or "active". Then click "Apply" button to apply setting.

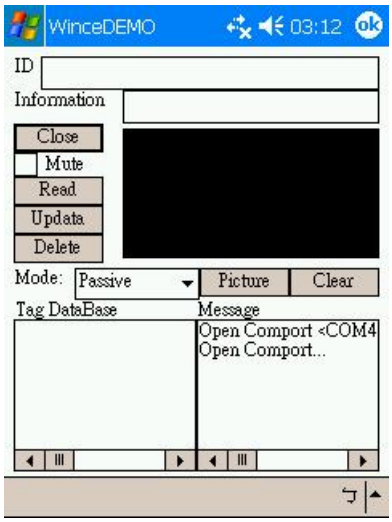


"Mode:" selects "passive" or "active", it executes the following code.

```
void CWinceDEMODlg::OnCbnSelchangeComboOpmode()
{
	// TODO: 在此加入控制項告知處理常式程式碼
	CString sTemp,sTemp2;
	BYTE byTemp=0;

	if(SUNLITRFID_IsOpen())
	{
	  SUNLITRFID_Pause();
	  ListMsg.InsertString(0,CString("Pause Device!!!")) ;
	  SUNLITRFID_ActiveScanIntervalSet(DEFAULT_SCAN_INTERVAL_10MS);
	  //-----------------------------------------------------------------------------
	  //Set Operation Mode
	  if( ComboBoxOPMode.GetCurSel()==0)    byTemp=0;//Passive Mode
	  else byTemp=0xff;//Active Mode

	  if(SUNLITRFID_OpModeSet(byTemp)==HANDLE_SUCCESS)
		ListMsg.InsertString(0,CString("Set Operation Mode Successed!!!")) ;

	  else ListMsg.InsertString(0,CString("Set Operation Mode Fail...")) ;
	  //-----------------------------------------------------------------------------
	  if(ButtonMute.GetCheck()==BST_UNCHECKED)
		MessageBeep(MB_ICONASTERISK);
	}
	else
	{
	  ListMsg.InsertString(0,CString("Comport Error...")) ;
	  if(ButtonMute.GetCheck()==BST_UNCHECKED)
		MessageBeep(MB_OK);
	}
}
```

(3) Read Tag

Click “Read” button to read tag. When it passive mode, reader will read tag once, when it is active mode, reader will start to read tag continuously.



The “Read” button executes the following code.

```
void CWinceDEMODlg::OnBnClickedButtonRead()
{
	// TODO:  在此加入控制項告知處理常式程式碼
	CString sTemp,sTemp2;
	BYTE byTemp=0;
	SUNLITRFID_TAGID TagID;

	ButtonRead.EnableWindow(false);

	SUNLITRFID_Pause();

	byTemp=SUNLITRFID_ScanTag(&TagID);

	DrawEnvironment.ClearPicture();

	switch(byTemp)
	{
	  case HANDLE_SUCCESS://Only in Active Mode
			ListMsg.InsertString(0,CString("(A)Read~~~"));
			break;

	  case HANDLE_ERR_TAG_PASSIVE://Read Tag in Passive Mode

			if(SDB.IsExists(&TagID)==-1)
			{
				if(ButtonMute.GetCheck()==BST_UNCHECKED)
					MessageBeep(MB_ICONASTERISK);

				AddNewTagToDB(&TagID);//Add New Tag Dialog
			}
			else   if(ButtonMute.GetCheck()==BST_UNCHECKED)
				MessageBeep(MB_OK);

			ShowTagInfo(&TagID,true);

			break;

	  case HANDLE_ERR_TAG_NOT_FOUND://Did't Read Tag in Passive Mode
			ListMsg.InsertString(0,CString("(P)Not Found Tag..."));
			break;

	  case HANDLE_ERR_COMPORT:
			ListMsg.InsertString(0,CString("Comport Error..."));
```

```
                break;

        case HANDLE_ERR_NO_RESPONSE:
                ListMsg.InsertString(0,CString("Device has no response???"));
                break;

        default:
                sTemp.Format(CString("Undefined value:%d..."),byTemp);
                ListMsg.InsertString(0,sTemp);
                break;
    }
    SetEvent(hEventFlip);
    ButtonRead.EnableWindow(true);
}
```

APPENDIX：Development Environment

Hardware Environment：

Product Name：RFID CF Type Reader V3.0

Model Name：SLC-10200

Firmware Version：V1.0 or later

Hardware Version：V1.0 or later

Software Environment：

Development Environment：Microsoft Visual C++ 2005 Professional〈Chinese Traditional〉

5. Disk information

Folder	File	Description
sunlitrfidppc.dll.(V1.1.0.1)2006-12-22		Include API
	sunlitrfidppc.h	Header file
	sunlitrfidppc.dll	Dynamic Link Library file
	sunlitrfidppc.lib	Library file
Demo software(V1.1.0.1)		Demo Software
	WinceDEMO.exe	Main program file
	sunlitrfidppc.dll	Dynamic Link Library file
	CFR_SLC-10100_ User Guide_V1_ENG.pdf	PDF file of user guide

6. Package list

Item	Quantity
CF type reader	1
Application CD	1

7. How to contact us

For further information or in case of difficulties please contact

Sunlit System Technology Corp.
www.sunlitcorp.com

8F1, NO19, LANE.120, SEC.1, Neihu Rd., Neihu Chiu Taipei Taiwan 114 R.O.C.
webmaster@sunlitcorp.com

TEL: 886-2-6600-6351
FAX: 886-2-6600-6765

Warning: In order to avoid misuse or any unexpected damage, please read this guide first.

FCC Statement

Regulatory Approvals

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation.

This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

Reorient or relocate the receiving antenna.

Increase the separation between the equipment and receiver.

Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

Consult the dealer or an experienced radio/TV technician for help.

To assure continued compliance, any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment. (Example - use only shielded interface cables when connecting to computer or peripheral devices).

FCC Radiation Exposure Statement

SAR compliance has been established in PDAs configurations with CompactFlash slot on the top, and can be used in PDA(s) with substantially similar physical dimensions, construction, and electrical and RF characteristics. SAR compliance for body-worn operations with PDA(s) is restricted to belt-clips, holsters, and accessories that have no metallic component in the assembly and which provide at least 1.5 cm separation between the device, including its antenna, and the users body. The antenna(s) used for this transmitter must not be co-located or operating in conjunction with any other antenna or transmitter

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference, and

(2) This device must accept any interference received, including interference that may cause undesired operation.

This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

The antennas used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter.