Programmer's Reference

SPT-3000

TABLE of Contents

OPOS DRIVER....4

SCOPE....4
CONCEPT....4
SUPPORTED DEVICES LIST....5
USING SAM4S OPOS SETUP UTILITY....6
MAIN Window....6
How to add a new device on a system....7
How to modify the registered device settings....8
How to remove the device from a registry....9
How to add a new LDN....9
How to remove the registered LDN....9
Testing the Sam4s OPOS Driver by the sample code program....10

VFD CONTROL LIBRARY....11

VFD_DLL.DLL....11
EXPORTED FUNCTIONS OF DLL....12
PortOpen....12
PortClose....13
ClearText....14
DisplayText....15
DisplayTextAt....16
AddTextAt....18
ClearDescriptors....20
SetDescriptors....21
SetCursorOnOff....23
SetOverwriteMode....24
SetVertiScrlMode....25
SetHoriScrlMode....26

DRAWER CONTROL LIBRARY (VISUAL C++)....27

VC_DRAWERDLL.DLL....27
EXPORTED FUNCTIONS OF DLL....28

DrawerInit 28
SetVoltage 29
SetDrawer 30
DrawerOpen 31
ReadStart 32

DRAWER CONTROL LIBRARY (VISUAL BASIC) 34

VB_DRAWERDLL.DLL 34
EXPORTED FUNCTIONS OF DLL 35
DrawerInit 35
SetVoltage 36
SetDrawer 37
DrawerOpen 38
ReadStart 39

OPOS Driver

Scope

This document explains how to setup POS devices by the SAM4S OPOS Driver Setup Application and OPOS itself as well.

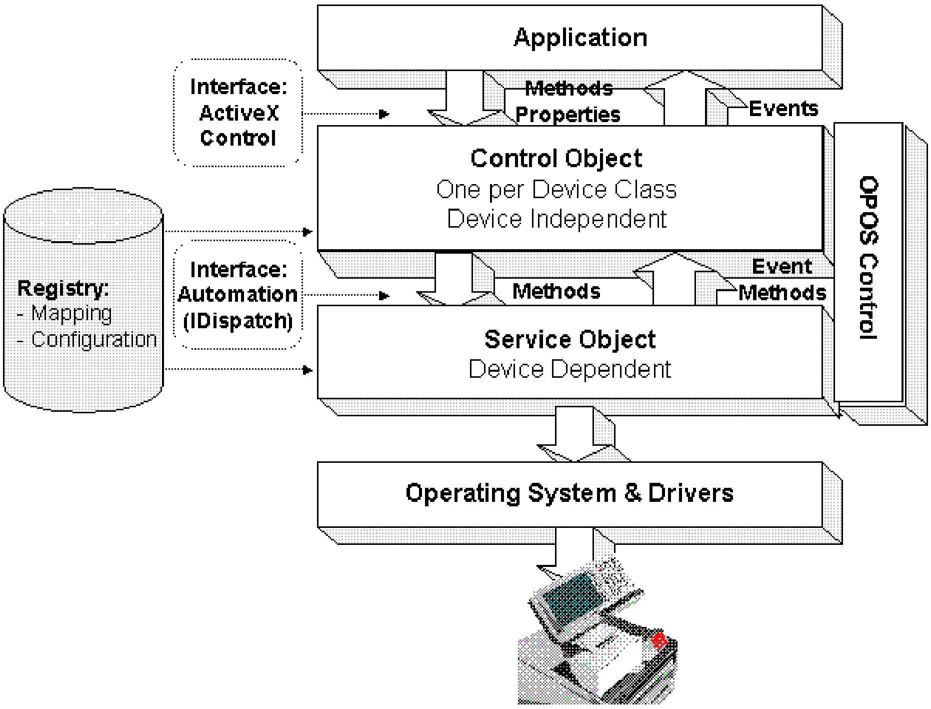
You can do following jobs by the SAM4S OPOS Driver Setup Application.

- Add or Remove POS Devices in a system registry.
- Modify setting of the connection between a device and an OPOS driver.
- Add or Remove a logical device name (LDN) of a registered pos device.

Concept

OPOS(OLE for Retail POS) was the first widely-adopted POS device standard. It was initiated by Microsoft, NCR, Epson, and Fujitsu-ICL to help integrate POS hardware into applications for the Windows™ family of operating systems. OPOS uses COM technology, and is therefore language independent.

The OLE for Retail POS software is implemented using the layers shown in the following diagram:



Supported Devices List

POS Printer Class :

- Sam4s ELLIX10S
- Sam4s ELLIX10P (Parallel)
- Sam4s ELLIX20S
- Sam4s ELLIX20P (Parallel)

Cash Drawer Class:

- PCDSTD (For Sam4s ELLIX Series)
- QCDSTD (For QBIG POS Terminal)
- QCDUSB (For SPT-3000 POS Terminal)

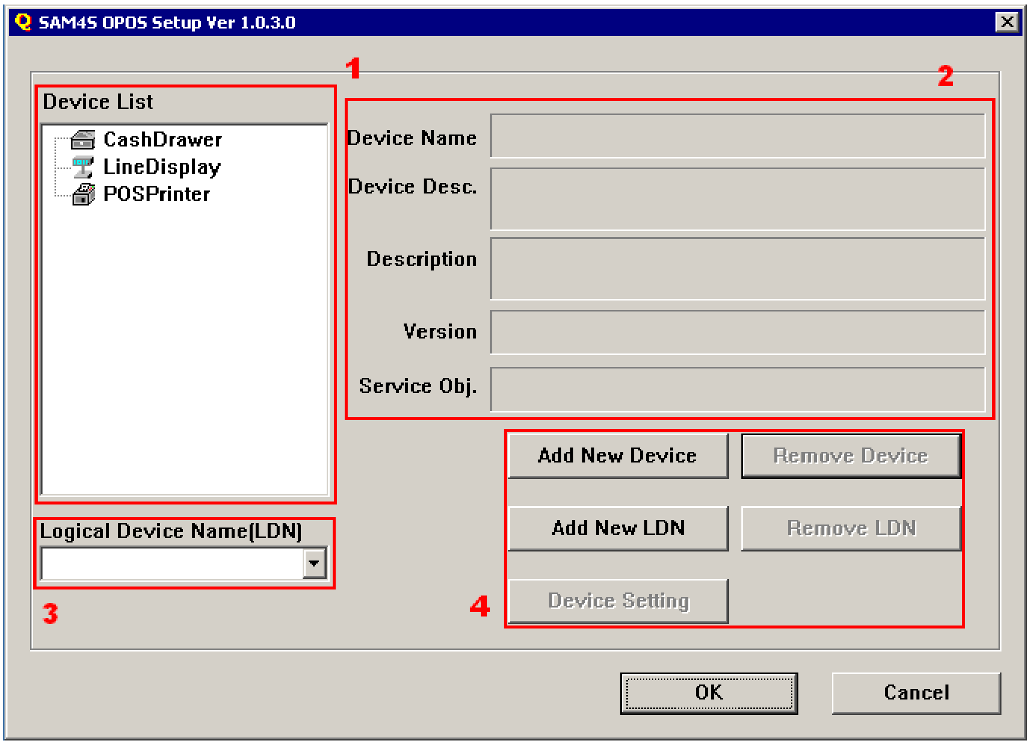
Line Display Class:

- QLD202C (For QBIG and SPT-3000)

Notes

For more detail information about OPOS or UnifiedPOS (UPOS) architecture, please visit the UPOS committee web site. (**http://monroecs.com/**)

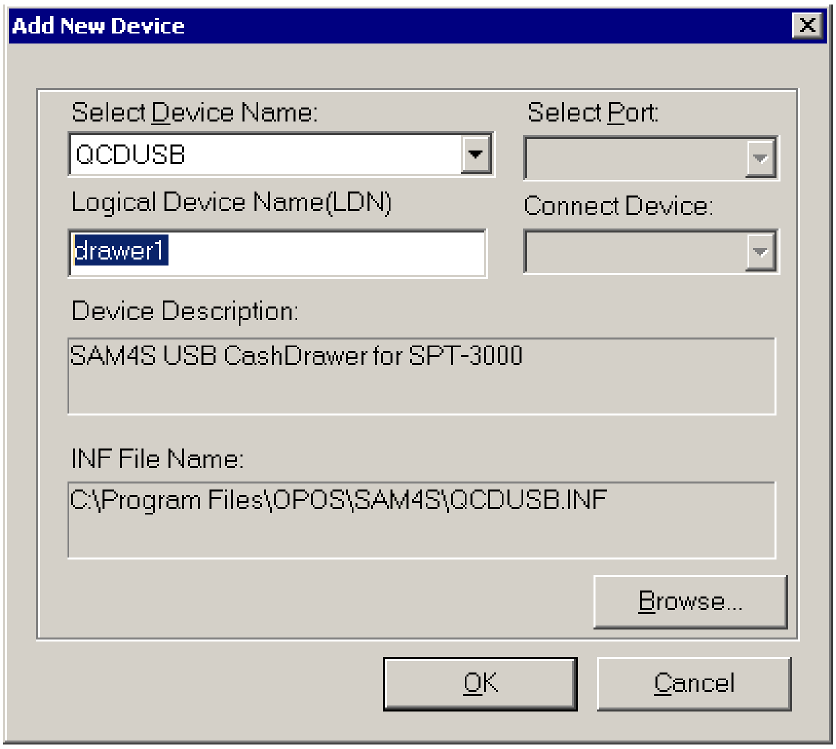
MAIN Window



1. POS devices list, which are registered on a system.
2. Detail information about a control object which is selected on devices list.
3. List of LDN (Logical Device Name) about a selected device.
4. Register, modify or delete devices.

How to add a new device on a system

1. Click “Add New Device” on the main window.



Select Device Name: QCDUSB

Select Port:

Logical Device Name(LDN) drawer1

Connect Device:

Device Description: SAM4S USB CashDrawer for SPT-3000

INF File Name: C:\Program Files\OPOS\SAM4S\QCDUSB.INF

2. Select a device class name from “Select Device Name” combo box.
3. Select a port type from “Select Port” combo box if there are multiple port types of the selected device.
4. Write a name on the edit box if you want to make **a Logical Device Name** (LDN)[1] for the selected device.
5. You can see the device description and check the path and the name of device’s INF file.
6. Click “OK” to save settings to a registry with closing this (“Add New Device”) dialog.

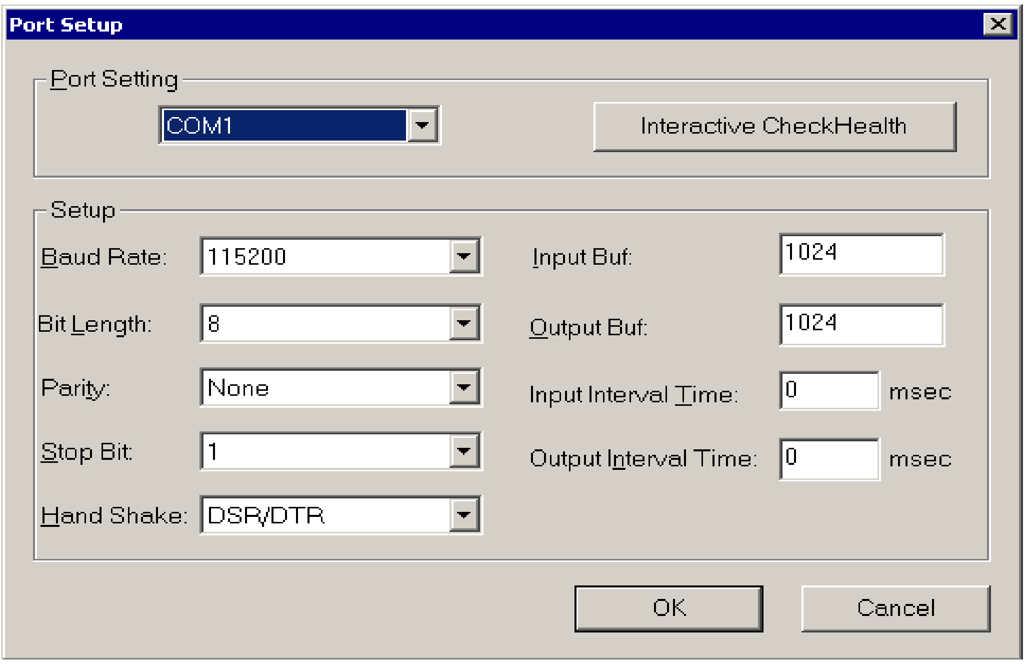
---

[1] **A Logical Device Name(LDN)** : An application may open a Control by passing the Device Name key to the **Open** method. In many cases, however, the application will want a level of isolation where the application specifies a “Logical Device Name” that is translated into a Device name. *–From UnifiedPOS Specification*

How to modify the registered device settings

1. Select a device on “Device List” which is on the left side of main window.
2. Click “Device Setting” to change device settings.

   “Port Setup” dialog is popped if a device using a serial or a parallel port is selected in the device list.



Port Setting: COM1

Setup

Baud Rate: 115200

Bit Length: 8

Parity: None

Stop Bit: 1

Hand Shake: DSR/DTR

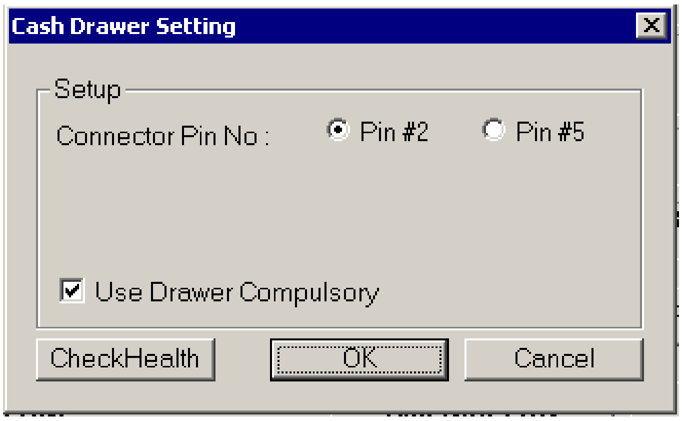
Input Buf: 1024

Output Buf: 1024

Input Interval Time: 0 msec

Output Interval Time: 0 msec

“Cash Drawer Setting” dialog is popped if a cash drawer device is selected.



Setup

Connector Pin No : ● Pin #2 ○ Pin #5

☑ Use Drawer Compulsory

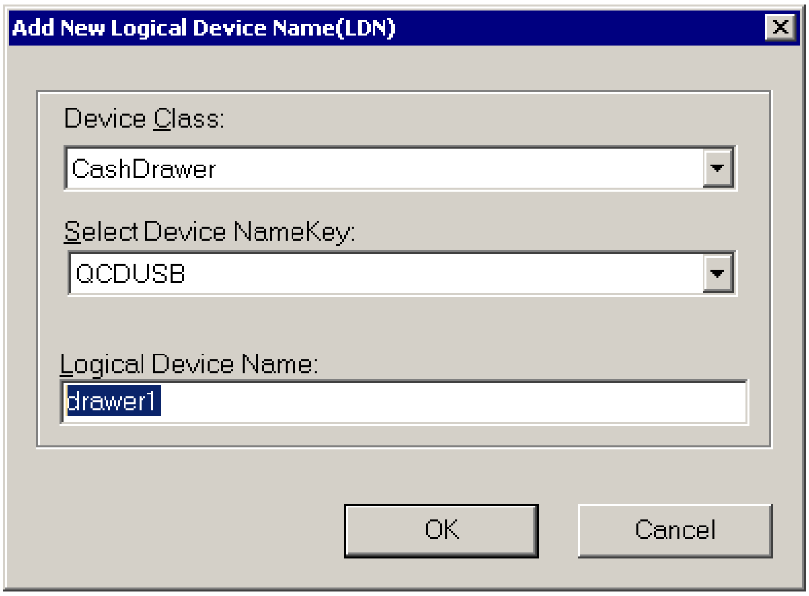
3. Click the “CheckHealth” to start CheckHealth method.

How to remove the device from a registry

1. Select a device on “Device List” which is on the left side of main window.
2. Click “Remove Device”.

How to add a new LDN

1. Click “Add New LDN”.



Device Class: CashDrawer

Select Device NameKey: QCDUSB

Logical Device Name: drawer1

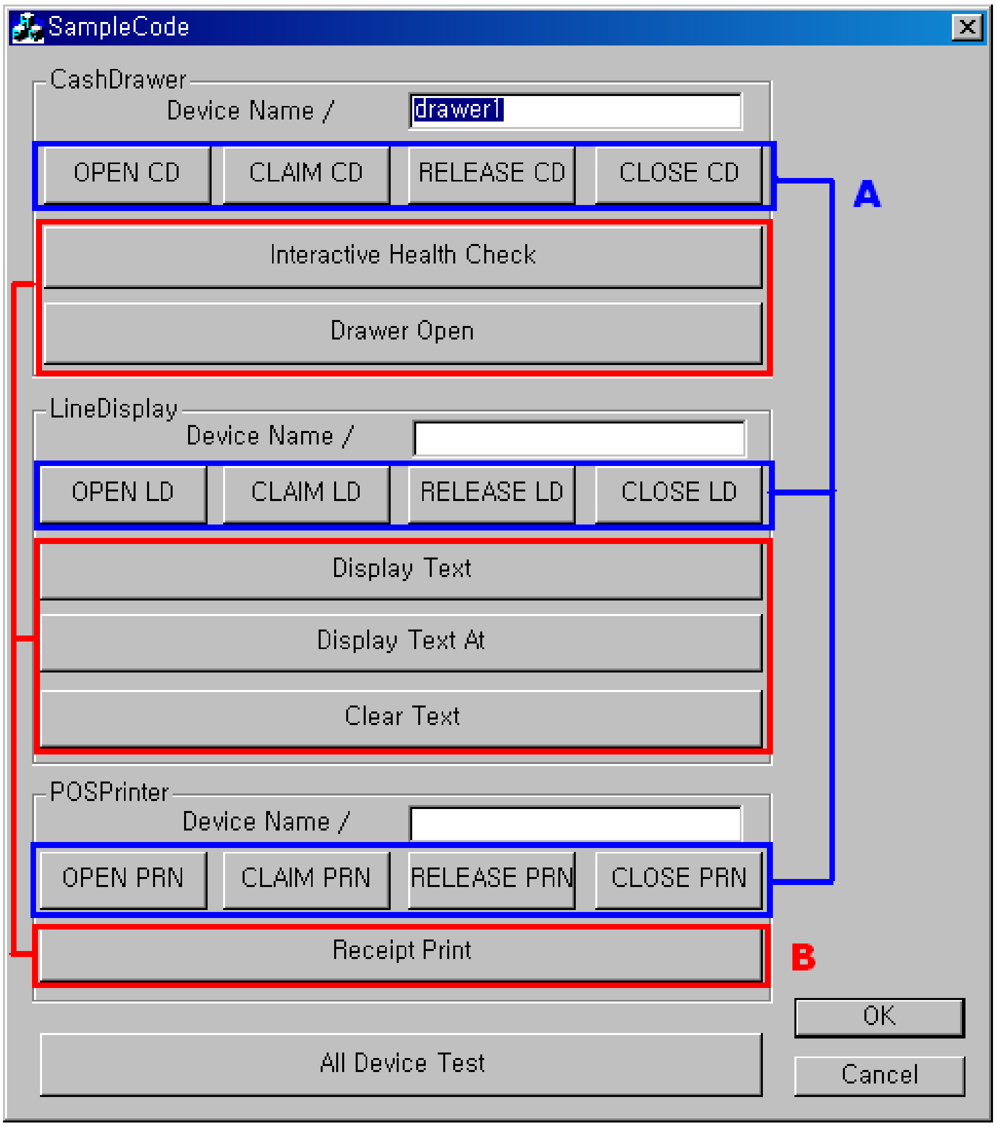
2. Select a device class and a device name key properly.
3. Write a LDN and click “OK”.

How to remove the registered LDN

1. Select a device on “Device List” which is on the left side of main window.
2. Select the LDN which will be removed from a registry.
3. Click “Remove LDN”.

Testing the Sam4s OPOS Driver by the sample code program

1. Run “SampleCode.exe” which is located in “<Program Files> \ OPOS \ Sam4s \ SampleCode \ VC”.



2. This program supports three types of POS devices class.
3. Write the device name or LDN into an edit box of which a device is tested.
4. The OPOS driver test sequence (A & B) is like as below. (XX is device class. E.g. CD is CashDrawer.)
   Open device (OPEN XX) → Claim device (CLAIM XX) →
   Device operation (B) → Release device (RELEASE XX) → Close device (CLOSE XX)

VFD Control Library

VFD_DLL.dll

You can control SPT-3000's VFD module (2x20) from this file. SPT-3000's VFD is character type.

In VFD_DLL.dll, setting values for COM5 are like below metrics and they are fixed.

COM5 is port for VFD of SPT-3000.

Initial setting value	
Port	COM5
Baud Rate	9600 bps
Data Bit	8
Parity Bit	1
Stop Bit	1

You can load or unload dll library dynamically, and refer the way how to as below.

```
//how to load dll
hDll = LoadLibrary("VFD_DLL.dll");

//how to release dll
FreeLibrary(hDll);
```

Exported functions of DLL

VFD_DLL.dll has several interfaces. ALL OF THEM CAN BE CALLED IN THE CONDITION OF DLL LOADED.

PortOpen

long PortOpen();

Return Value

TRUE

Open port is succeeded.

FALSE

Open port is failed. Port cannot be opened or is already opened.

Parameters

None

Remarks

We can set port open with this method. VFD can be set features; text clear (ClearText), overwriting mode (setOverwriteMode), and cursor hiding (setCursorOnOff) in PortOpen.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*PortOpenFunc)();

//how to get function pointer pointed to function of dll
PortOpenFunc lpPortOpenFunc;
lpPortOpenFunc = (PortOpenFunc)GetProcAddress(hDll,"PortOpen");

//how to call function of dll
lpPortOpenFunc();
```

PortClose

long PortClose();

Return Value

TRUE

Close Port is succeeded.

FALSE

Close port is failed. Port cannot be closed or is not yet opened.

Parameters

None

Remarks

We can set port close with this method.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*PortCloseFunc)();

//how to get function pointer pointed to function of dll
PortCloseFunc lpPortCloseFunc;
lpPortCloseFunc = (PortCloseFunc)GetProcAddress(hDll,"PortClose");

//how to call function of dll
lpPortCloseFunc();
```

ClearText

long ClearText();

Return Value

TRUE

Clear text is succeeded.

FALSE

Clear text is failed. This is occurred when port is not opened.

Parameters

None

Remarks

ClearText clears the current window to blanks, sets CursorRow and CursorColumn to zero, and resynchronizes the beginning of the window with the start of the viewport.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*clearTextFunc)();

//how to get function pointer pointed to function of dll
clearTextFunc lpclearTextFunc;
lpclearTextFunc = (clearTextFunc)GetProcAddress(hDll,"clearText");

//how to call function of dll
lpclearTextFunc();
```

DisplayText

long DisplayText(char str[]);

Return Value

TRUE

Display text is succeeded.

FALSE

Display text is failed. This is occurred when port is not opened.

Parameters

str

The string of characters to display.

Remarks

The characters in str are processed beginning at the location specified by CursorRow and CursorColumn, and continue in succeeding character positions. Any previous data in a character position is overwritten, including character.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*displayTextFunc)(char str[]);

//how to get function pointer pointed to function of dll
displayTextFunc lpdisplayTextFunc;
lpdisplayTextFunc = (displayTextFunc)GetProcAddress(hDll,"displayText");

//how to call function of dll
char* Temp = (char*)(const char*)str;
lpdisplayTextFunc(Temp);
```

DisplayTextAt

long displayTextAt(**long** posX, **long** posY, **char** str[]);

Return Value

TRUE

Function call is succeeded.

FALSE

Function call is failed. This is occurred when port is not opened.

Parameters

posX

The start row for the text. (Range 1~20)

posY

The start column for the text. (Range 1~2)

str

The string of characters to display.

Remarks

The characters in str are processed beginning at the window location specified by the posX and posY parameters, and continuing in succeeding columns. The line indicated by posY parameters will be cleared before displayTextAt method. The operational characteristics of the displayTextAt method are the same as the displayText method.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*displayTextAtFunc)(long X , long Y, char str[]);

//how to get function pointer pointed to function of dll
displayTextAtFunc lpdisplayTextAtFunc;
```

```
lpdisplayTextAtFunc = (displayTextAtFunc)GetProcAddress(hDll,"displayTextAt");

//how to call function of dll
int Temp1 = _ttoi(numX);
int Temp2 = _ttoi(numY);
char* Temp3 = (char*)(const char*)str;
lpdisplayTextAtFunc(Temp1, Temp2, Temp3);
```

AddTextAt

long AddTextAt(long posX, **long** posY, **char** str[]);

Return Value

TRUE	Function call is succeeded.
FALSE	Function call is failed. This is occurred when port is not opened.

Parameters

posX	The start row for the text. (Range 1~20)
posy	The start column for the text. (Range 1~2)
str	The string of characters to display.

Remarks

The characters in str are added beginning at the window location specified by the posX and posY parameters, and continuing in succeeding columns. The line indicated by posY parameters will not be cleared before displayTextAt method.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*AddTextAtFunc)(long X , long Y, char str[]);

//how to get function pointer pointed to function of dll
AddTextAtFunc lpAddTextAtFunc;
lpAddTextAtFunc = (AddTextAtFunc)GetProcAddress(hDll,"AddTextAt");

//how to call function of dll
int Temp1 = _ttoi(numX);
int Temp2 = _ttoi(numY);
```

```
char* Temp3 = (char*)(const char*)str;
lpAddTextAtFunc(Temp1, Temp2, Temp3);
```

ClearDescriptors

long clearDescriptors();

Return Value

TRUE

Function call is succeeded.

FALSE

Function call is failed. This is occurred when port is not opened.

Parameters

None

Remarks

This function is used for turning off all descriptors.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*ClsDescriptorFunc)();

//how to get function pointer pointed to function of dll
ClsDescriptorFunc lpClsDescriptorFunc;
lpClsDescriptorFunc = (ClsDescriptorFunc)GetProcAddress(hDll,"clearDescriptors");

//how to call function of dll
lpClsDescriptorFunc();
```

SetDescriptors

long setDescriptors(**long** data, **long** attribute);

Return Value

TRUE

Function call is succeeded.

FALSE

Function call is failed. This is occurred when port is not opened.

Parameters

Data

The row of the descriptor. The data parameter indicates which descriptor to change. (range 1~20)

Attribute

Function can turn the descriptor on/off (range 0~1) with attributes.

If attribute parameter is 0, the descriptor is off.

If attribute parameter is 1, the descriptor is on.

Remarks

Set the state of one of the descriptors, which are small indicators with a fixed label.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*SetDescriptorFunc)(long data,long attribute);

//how to get function pointer pointed to function of dll
SetDescriptorFunc lpSetDescriptorFunc;
lpSetDescriptorFunc = (SetDescriptorFunc)GetProcAddress(hDll,"setDescriptor");

//how to call function of dll
```

```
int Temp1 = _ttoi(num);
lpSetDescriptorFunc(Temp1,DISP_SD_ON);
```

SetCursorOnOff

long setCursorOnOff(**long** attribute);

Return Value

TRUE

Function call is succeeded.

FALSE

Function call is failed. This is occurred when port is not opened.

Parameters

Attribute

If attribute parameter is 0, the cursor is off.

If attribute parameter is 1, the cursor is on.

Remarks

Set the state of cursor.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*SetcuronFunc)(long attribute);

//how to get function pointer pointed to function of dll
SetcuronFunc lpSetcuronFunc;
lpSetcuronFunc = (SetcuronFunc)GetProcAddress(hDll,"setCursorOnOff");

//how to call function of dll
lpSetcuronFunc(1);
```

SetOverwriteMode

long setOverwriteMode();

Return Value

TRUE

Function call is succeeded.

FALSE

Function call is failed. This is occurred when port is not opened. Overwriting mode is set currently.

Parameters

None

Remarks

Selects overwrite mode as the screen display mode. In overwrite mode, entering a character code moves the cursor to the left end of the lower line when the cursor is at the right end of the upper line, and to the left end of the upper line when the cursor is at the right end of the lower line.

This mode is selected when the power is turned on. Selecting overwrite mode cancels horizontal or vertical scroll mode. Except when the cursor is at the right end, entering a character code moves the cursor one character to the right after displaying the character.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*OverwritemodeFunc)();

//how to get function pointer pointed to function of dll
OverwritemodeFunc lpOverwritemodeFunc;
lpOverwritemodeFunc = (OverwritemodeFunc)GetProcAddress(hDll,"setOverwriteMode");

//how to call function of dll
lpOverwritemodeFunc();
```

SetVertiScrlMode

long setVertiScrlMode();

Return Value

TRUE

Function call is succeeded.

FALSE

Function call is failed. This is occurred when port is not opened. Current mode is Vertical scroll.

Parameters

None

Remarks

Selects vertical scroll mode as the screen display mode. In vertical scroll mode, entering a character code moves the cursor to the left end of the lower line when the cursor is at the right end of the upper line, scrolls the characters displayed on the lower line to the upper line, and clears the lower line when the cursor is at the right end of the lower line. At this time, the cursor is moved to the left end of the lower line.

Selecting vertical scroll mode cancels overwrite or horizontal scroll mode. Except when the cursor is at the right end, entering a character code moves the cursor one character to the right after displaying the character.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*VertiscrlmodeFunc)();

//how to get function pointer pointed to function of dll
VertiscrlmodeFunc lpVertiscrlmodeFunc;
lpVertiscrlmodeFunc = (VertiscrlmodeFunc)GetProcAddress(hDll,"setVertiScrlMode");

//how to call function of dll
lpVertiscrlmodeFunc();
```

SetHoriScrlMode

long setHoriScrlMode();

Return Value

TRUE

Function call is succeeded.

FALSE

Function call is failed. This is occurred when port is not opened. Current mode is Horizontal scroll.

Parameters

None

Remarks

Selects horizontal scroll mode as the screen display mode. In horizontal scroll mode, entering a character code scrolls all displayed characters (including commas and periods) one character to the left, then displays the new character at the right end (when the cursor is at the right end of either line.)
Selecting horizontal scroll mode cancels overwrite or vertical scroll mode. Except when the cursor is at the right end, entering a character code moves the cursor one character to the right after displaying the character.

Example (Pseudo codes)

```
//how to define function pointer type
typedef long (*HoriscrlmodeFunc)();

//how to get function pointer pointed to function of dll
HoriscrlmodeFunc lpHoriscrlmodeFunc;
lpHoriscrlmodeFunc = (HoriscrlmodeFunc)GetProcAddress(hDll,"setHoriScrlMode");

//how to call function of dll
lpHoriscrlmodeFunc();
```

Drawer Control Library (Visual C++)

vc_DrawerDll.dll

You can control SPT-3000's Drawer module from this file. You can load or de-load dll library dynamically, and refer the way how to as below.

```
//how to load dll
hDll = LoadLibrary("vc_DrawerDll.dll");

//how to release dll
FreeLibrary(hDll);
```

Exported functions of DLL

vc_DrawerDll.dll has several interfaces. ALL OF THEM CAN BE CALLED IN THE CONDITION OF DLL LOADED.

DrawerInit

bool DrawerInit();

Return Value

TRUE

When a drawer device is founded in system

FALSE

When a drawer device is not founded in system

Parameters

none

Remarks

Notice that a drawer device is founded or not in system.

Example (Pseudo codes)

```
typedef int (*DrawerInitFunc)();

DrawerInitFunc lpDrawerInitFunc;

lpDrawerInitFunc = (DrawerInitFunc)GetProcAddress(hDll,"DrawerInit");

if (lpDrawerInitFunc() == FALSE)
{
    AfxMessageBox("Drawer device is not founded.");
}
```

SetVoltage

```
void SetVoltage(int m_Voltage);
```

Return Value

none

Parameters

m_Voltage

This means a drawer's voltage setting value. A number '12' means '12 Voltage', and a number '24' means '24 Voltage'.

Remarks

Set a drawer's voltage value.

Example (Pseudo codes)

```
typedef int (*SetVoltageFunc)(int);

SetVoltageFunc lpSetVoltageFunc;

lpSetVoltageFunc = (SetVoltageFunc)GetProcAddress(hDll,"SetVoltage");

lpSetVoltageFunc(12);

GetDlgItem(IDC_STAT_SVOL)->SetWindowText("12 V");
```

SetDrawer

void SetDrawer(int m_DrawerNo)

Return Value

none

Parameters

m_DrawerNo

This means a drawer's number. A number '1' means 'Drawer 1', and a number '2' means 'Drawer 2'.

Remarks

Set a drawer's number.

Example (Pseudo codes)

```
typedef int (*SetDrawerFunc)(int);

SetDrawerFunc lpSetDrawerFunc;

lpSetDrawerFunc = (SetDrawerFunc)GetProcAddress(hDll,"SetDrawer");

lpSetDrawerFunc(m_DrawerNo);

GetDlgItem(IDC_STAT_DRANO)->SetWindowText("Drawer 1");
```

DrawerOpen

```
void DrawerOpen(int m_DrawerNo)
```

Return Value

none

Parameters

m_DrawerNo

This means a drawer's number. A number '1' means 'Drawer 1', and a number '2' means 'Drawer 2'.

Remarks

Open a drawer determined by 'm_DrawerNo' value.

Example (Pseudo codes)

```
typedef int (*DrawerOpenFunc)(int);

DrawerOpenFunc lpDrawerOpenFunc;

lpDrawerOpenFunc = (DrawerOpenFunc)GetProcAddress(hDll,"DrawerOpen");

lpDrawerOpenFunc(m_DrawerNo);
```

ReadStart

Int ReadStart()

Return Value

1

OPENDED-Ready : Drawer is opened.

2

CLOSED-Charging: Charging during a drawer is closed. Charging time usually takes about 4.5 sec.

3

OPENDED- Charging: Charging during a drawer is opened. Charging time usually takes about 4.5 sec.

4

CLOSED- Ready: Drawer is closed.

Parameters

none

Remarks

Read a current drawer status.

Example (Pseudo codes)

```
typedef int (*ReadStartFunc)();

ReadStartFunc lpReadStartFunc;

lpReadStartFunc = (ReadStartFunc)GetProcAddress(hDll,"ReadStart");

int result = lpReadStartFunc();

if (result == 1)
{
        GetDlgItem(IDC_STAT_DRASTAT)->SetWindowText("OPENED-Ready");
}
```

```
else if (result == 2)
{
    GetDlgItem(IDC_STAT_DRASTAT)->SetWindowText("CLOSED-Charging");
}
else if (result == 3)
{
    GetDlgItem(IDC_STAT_DRASTAT)->SetWindowText("OPENED-Charging");
}
else if (result == 4)
{
        GetDlgItem(IDC_STAT_DRASTAT)->SetWindowText("CLOSED-Ready");
}
```

Drawer Control Library (Visual BASIC)

vb_DrawerDll.dll

You can control a SPT-3000’s drawer module from this file. You can load or de-load dll library dynamically, and refer the way how to as below.

```
//using Dll
Private Declare Function DrawerOpen Lib "c:\windows\system32\vb_DrawerDll.dll" () As Long
```

‘DrawerOpen’ is a function name called from a dll file. “c:\windows\system32\vb_DrawerDll.dll" is a location which is a dll file is located. You should use functions below a function list, and a dll location can be modified by users. You have to distinguish an upper character and a lower character when using dll function.

Exported functions of DLL

‘vb_DrawerDll.dll’ has several interfaces. ALL OF THEM CAN BE CALLED IN THE CONDITION OF DLL LOADED.

DrawerInit

Boolean DrawerInit();

Return Value

TRUE

When a drawer device is founded in system

FALSE

When a drawer device is not founded in system

Parameters

none

Remarks

Notice that a drawer device is founded or not in system.

Example (Pseudo codes)

```
If (DrawerInit() = False) Then
    MsgBox (“Drawer device is not founded.”)
End If
```

SetVoltage

void SetVoltage(m_Voltage As Integer)

Return Value

none

Parameters

m_Voltage

This means a drawer's voltage setting value. A number '12' means '12 Voltage', and a number '24' means '24 Voltage'.

Remarks

Set a drawer's voltage value.

Example (Pseudo codes)

```
Call SetVoltage(m_Voltage)
```

SetDrawer

void SetDrawer(m_DrawerNo As Integer)

Return Value

none

Parameters

m_DrawerNo

This means a drawer's number. A number '1' means 'Drawer 1', and a number '2' means 'Drawer 2'.

Remarks

Set a drawer's number.

Example (Pseudo codes)

```
Call SetDrawer(m_DrawerNo)
```

DrawerOpen

```
void DrawerOpen(m_DrawerNo As Integer)
```

Return Value

none

Parameters

m_DrawerNo

This means a drawer's number. A number '1' means 'Drawer 1', and a number '2' means 'Drawer 2'.

Remarks

Open a drawer determined by 'm_DrawerNo' value.

Example (Pseudo codes)

```
Call DrawerOpen(m_DrawerNo)
```

ReadStart

Int ReadStart()

Return Value

1

OPENDED-Ready : Drawer is opened.

2

CLOSED-Charging: Charging during a drawer is closed. Charging time usually takes about 4.5 sec.

3

OPENDED- Charging: Charging during a drawer is opened. Charging time usually takes about 4.5 sec.

4

CLOSED- Ready: Drawer is closed.

Parameters

none

Remarks

Read a current drawer status.

Example (Pseudo codes)

```
Dim result As Integer
result = ReadStart()

If (result = 1) Then
        IDC_STAT_DRASTAT.Caption = "OPENED-Ready"
ElseIf (result = 2) Then
        IDC_STAT_DRASTAT.Caption = " CLOSED-Charging"
ElseIf (result = 3) Then
        IDC_STAT_DRASTAT.Caption = " OPENED-Charging"
ElseIf (result = 4) Then
        IDC_STAT_DRASTAT.Caption = " CLOSED-Ready"
End If
```