FP-2000 Series Printer
POS Printer, Cash Drawer
Application Programmer's Guide
of
OPOS.NET Class Library
for
Serial/ USB/ LAN Interface

Version 1.0

FUJITSU ISOTEC LIMITED.

Index

1. Outline

FP POS Printer OPOS.NET Class Library and Drawer OPOS.NET Class Library that control Fujitsu Isotec Limited FP POS Printer (FP-2200/FP-2100/FP-2000/FP-2000L) and Drawer connected to the printer are OPOS.NET Class Library conforming to OPOS 1.13 POS Printer Devise and Drawer Device. When using this class library, refer to "UnifiedPOS Retail Peripheral Architecture”, too.
In this guide, "FP" means same as "FP-2200", "FP-2100", "FP-2000" or "FP-2000L".

1.1. Subject Scope of this document

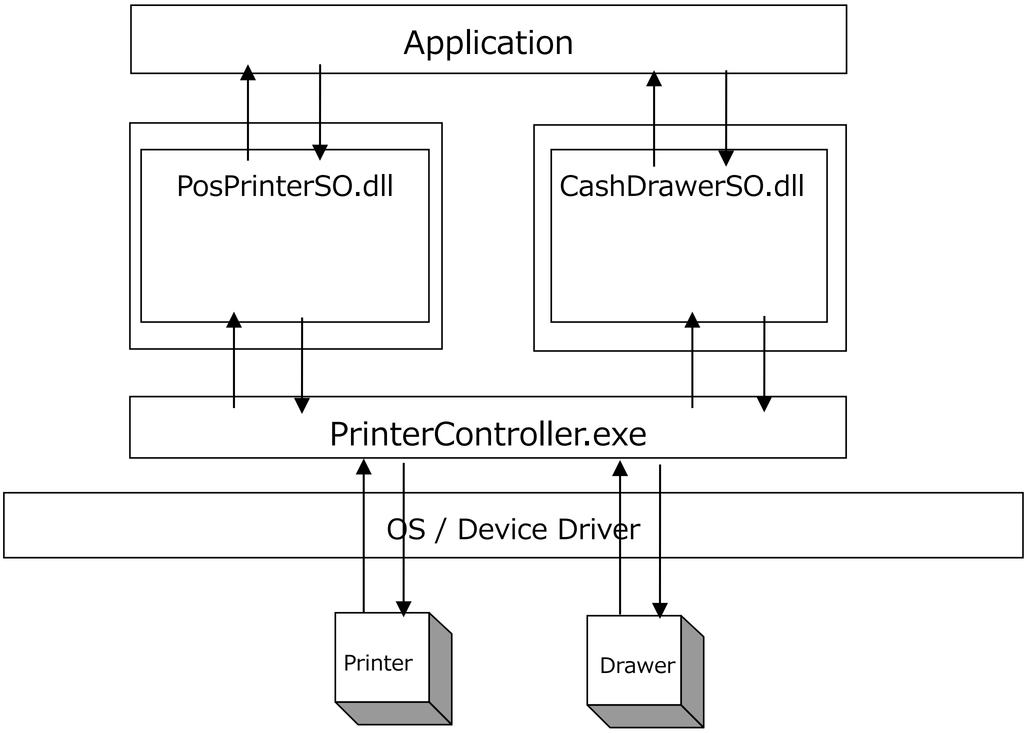
These instructions (Application Programmer's Guide) aim for the main reference of programmers who develop the application for the use of OPOS.NET Class Library, and describe the following contents necessary for that.

- Installation way of OPOS.NET Class Library
- Usage of OPOS.NET Class Library
- Restrictions of OPOS.NET Class Library
- Interface (Property/Method/Event) Remarks of OPOS.NET Class Library
- Item Setting Remarks of OPOS.NET Class Library

1.2. OPOS Control Outline

1) OPOS Control Configuration Drawing

This class library is created in .NET Framework 4.0. Provides properties, methods, and events to the application. Applications receive processing results through method return values and parameters, properties, and events.



*This class library supports the control of Serial/USB/LAN Interface Printer.

*Multiple numbers of interface and printers can be set to the driver. For details, refer to Chapter 8 "Using Multiple Printers."

*This class library works with Thread Model of STA(Single, Thread or Apartment). To work with several processes, the setting value of "Apartment" in PosExplorer should be set as "1".

*Regarding LAN interface, Network Connection corresponds to 10Base-T, 100Base-TX.

2) Terminology

a. Service Object (Service Object; SO)

A service object is a class that implements the interface of a device class defined by POS for .NET. Exposes properties and methods called from the application.

1.3. Restrictions

Followings are restrictions.

1) POS Printer

[Restrictions on OPOS specifications]

All the interfaces of OPOS POS Printer Device are provided, but there are the following restrictions.

a. It does not support property setting concerning journal printing and journal.
b. It does not support property setting concerning slip printing and slip.
c. It does not support functions of Italic, custom color, shading printing, and cartridge.
d. It does not support change of receipt printing character font. (Printing font change)
e. The following methods always return Illegal(106) after enabling.
    PrintTwoNormal Method
    BeginInsertion Method
    EndInsertion Method
    BeginRemoval Method
    EndRemoval Method
    ChangePrintSide Method
    MarkFeed Method
    ResetStatistics Method
    RetrieveStatistics Method
    UpdateStatistics Method

(Limitation of Cable disconnection and connection)

There are conditions for disconnection and connection of LAN cable under printer "enable" status:

1)When LAN cable is disconnected from PC, printer cannot support any actions. When the cable is disconnected during "enable" status, this class library may fail to be "enable" even though the application restarts.

2)When LAN cable is disconnected from Printer, and while this class library and printer is connecting, printer cannot support any actions even if "disable" is sent. If "disable" is sent in above situation, even though the application is restarted, this class library may fail to enable.

When the LAN cable is disconnected, connect the LAN cable again, and switch off and on the power of Printer.

2) Drawer

[Restrictions of OPOS specifications OPOS]

All the interfaces of OPOS Drawer Device are provided, but there are the following restrictions.

a. **PowerNotify** Property (Power source notifying function setting)

Setting is only for Disabled(0) (Impossible to notify)and unchangeable.

b. **PowerState** Property (Power source state)

Only Unknown(2000)(Unclear) is set.

c. **DirectIO** Method (Particular-to-Device function)

It is not supported. After enabling, it always returns Illegal(106).

d. **WaitForDrawerClose** Method (Waiting for the drawer to close)

It is not supported. After enabling, it always returns Illegal(106).

e. **DirectIOEvent** Event (Particular- to-Device event)

It is not supported.

f. **DrawerOpened** Property, **StatusUpdateEvent** Event

Status notification of the Drawer is available only when **CapStatus** is TRUE and Printer class library is enabled (**DeviceEnabled**=TRUE) for the printer connected to the drawer. In case these conditions are not met, the status of the drawer is not notified.

[Restriction of Drawer Hardware Specifications]

It does not support notifying function of drawer power source condition.

3) Common Restrictions on POS Printer and Drawer

This class library is not thread-safe. When the method or property is accessed from the different thread, unexpected result may occur. In the multithread environment, implementation of exclusive processing for the critical sections is required for accessing the property and executing the method in order to avoid the method and property are executed at the same time.

4) Restriction when Windows driver and OPOS driver are installed in the same system

Problems such as failure to print correctly from the OPOS driver may occur if the Windows driver and OPOS driver are both installed in the same system.

In this case, it is recommended that you uninstall the driver that is not being used.

5) Setting of Apartment ="0" with connecting LAN interface cable

Apartment is set as "0" in PosExplorer setting, one process can control several printers. But the several processes do "Open" this class library (LAN Interface) at the same time, unexpected results may happen.

6) Setting of Apartment ="1" with connecting LAN interface cable

Apartment is set as "0" in PosExplorer setting, several processes can control printers.

In this case, the number of printers (LAN interface) should be guaranteed that one process should be less than one unit. If more than 2 printers per one process were used, unexpected results might occur. The following chart shows the examples of the case that Apartment ="1" can work, and the case that Apartment ="1" cannot work. (This class library uses LAN interface.)

7) This OPOS driver does not support the following printer functions.

- "Retry at Error"
- "Plug and Play"

* The above settings are changed to "Disable" when setting with Setup Tool.

8) Precautions when using linerless printer. (FP-2000L)

When the paper sensor setting is enabled.

1. AsyncMode = false and Transaction Printing is recommended.
2. If the next print is performed while the printer is waiting for the paper taken, cancel the waiting period for the paper taken (remove the paper) within two minutes.

   If you do not paper is removed for more than two minutes, the following print result is Timeout.

   (After that, when you remove the paper, the next print occurs.)

   If you want to extend the Timeout time, please change the following PosExplorer.

   Name : SendTimeout (Specify in msec units.)

9) This driver does not support concurrent usage with the OPOS driver (OCX).

10) This driver does not support operation by sleep or hibernation the PC in the device open state.

1.4. Connection Way to POS Printer

Set the POS Printer to the following settings (in gray highlight). Rest of the values can be set in the PosExplorer or the setting program attached with installer.

Memory Swith 1

No.	**Setting Item**	**Setting Contents**
1	Power On Status	*Set form the registry
2	Receive Buffer	64 KB
3	Busy Condition	Bufferfull
4	Receive Error	? Print
5	Auto LF	Disable
6	DSR (#6) RESET	Disable
7	INIT (#25) RESET	Disable
8	USB Soft Reset	Enable

Memory Swith 2

No.	**Setting Item**	**Setting Contents**
1	Cover Open Error	Auto Recovery
2	Error	Recovery by CMND
3	Batch (COM IF)	Disable
4	Batch (BT IF)	Disable
5	Batch (Other IF)	Disable
6	Serial Number	Enable
7	ASB	Enable
8	Font-B *2	Mode1
9	Font *2	*Set from the setting program

*2 : Depending on the language, this is not displayed.

Print

No.	**Setting Item**	**Setting Contents**
1	DPI	*Set from the setting program
2	Paper Width	*Set from the setting program
3	Max Speed	*Set from the setting program
4	Print Density	*Set from the setting program
5	Retry at Error	Disable
7	Language Selection	*Set from the setting program

Hardware

No.	Setting Item	Setting Contents
1	User NV Memory	192KB
2	Graphic Memory	384KB (FP-2000) 896KB (FP-2200 / FP-2100 / FP-2000L)
3	Cut at CoverClose	*Set from the setting program
4	Cutter Mode *3	*Set from the setting program
5	PNE Detect *3	*Set from the setting program
6	Paper Out Senser *3	*Set from the setting program
7	Cleaning Message	Enable

*3 : Depending on the model, this is not displayed.

Buzzer

No.	Setting Item	Setting Contents
1	Error Alert	*Set from the setting program
2	Buzzer Interval	*Set from the setting program
3	Buzzer Repetition	*Set from the setting program
4	Buzzer after Cut *3	*Set from the setting program
5	Paper Out Buzzer *3	*Set from the setting program

*3 : Depending on the model, this is not displayed.

Interface

No.	Setting Item	Setting Contents
1	Protocol	XON/XOFF
2	USB	Printer
3	Plug and Play	Disable

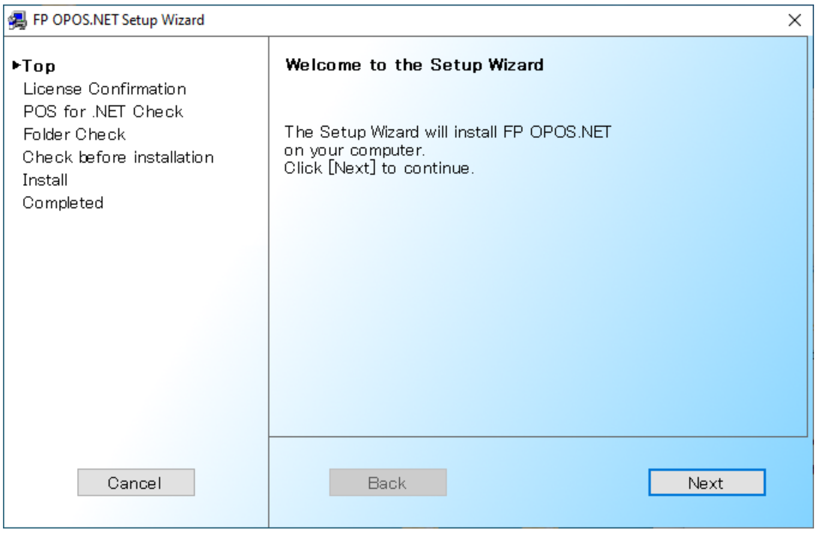
2. Installation

To install this class library, follow the procedure below.

2.1. Installation Procedure

1. In the OPOS.NET folder, double-click Setup.exe.
   If the [User Account Control] dialog appears, click [Yes].

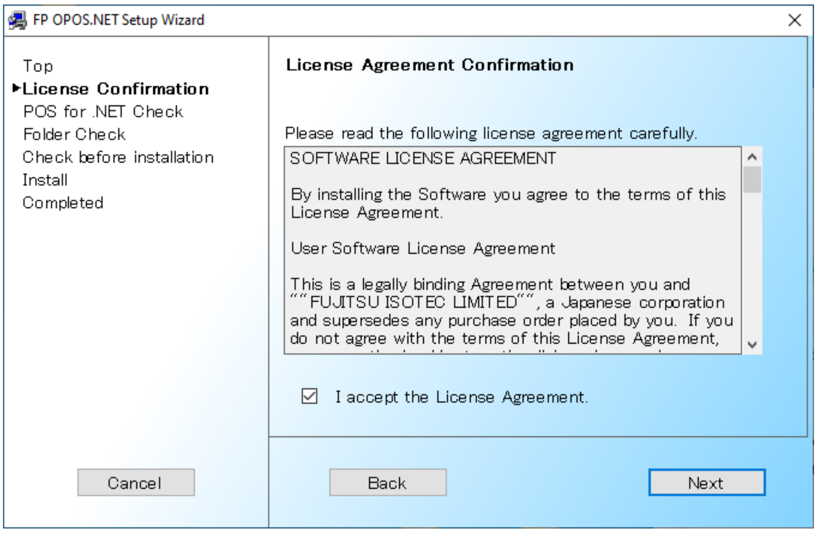
2. The Setup Wizard screen appears. Click [Next].



Welcome to the Setup Wizard

The Setup Wizard will install FP OPOS.NET
on your computer.
Click [Next] to continue.

3. Read the license agreement and check “I accept the License Agreement ”.
   Click [Next].



License Agreement Confirmation

Please read the following license agreement carefully.

SOFTWARE LICENSE AGREEMENT

By installing the Software you agree to the terms of this License Agreement.

User Software License Agreement

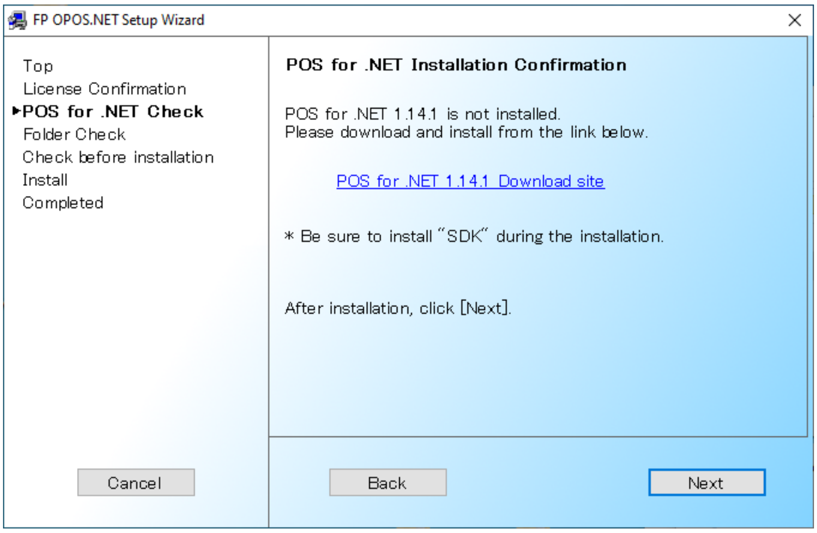
This is a legally binding Agreement between you and ““FUJITSU ISOTEC LIMITED””, a Japanese corporation and supersedes any purchase order placed by you. If you do not agree with the terms of this License Agreement,

☑ I accept the License Agreement.

4. If POS for .NET 1.14.1 is not installed, click the link on the screen below to download and install POS for .NET 1.14.1.

   When the installation is complete, click [Next].

   * If POS for .NET 1.14.1 is already installed, this screen does not appear, so go to step 5.



POS for .NET Installation Confirmation

POS for .NET 1.14.1 is not installed.
Please download and install from the link below.

POS for .NET 1.14.1 Download site

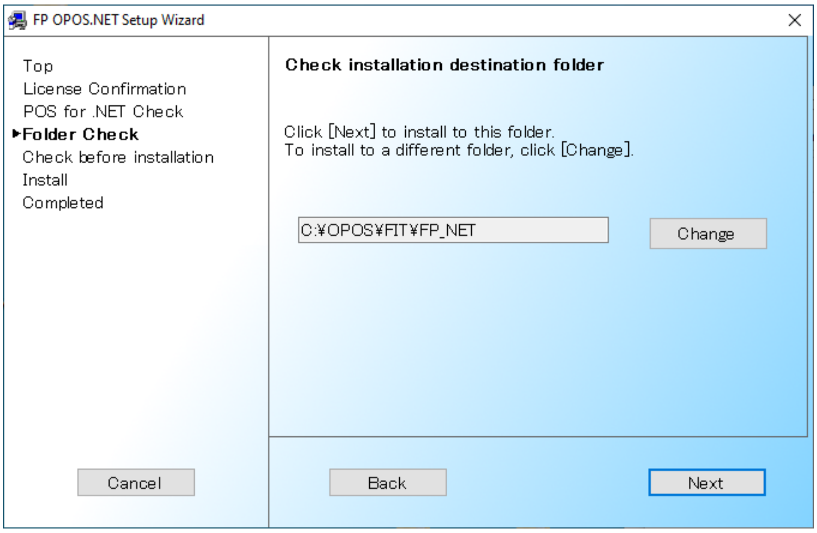
* Be sure to install "SDK" during the installation.

After installation, click [Next].

5. Specify the folder where you want to install the OPOS.NET driver.

   Change the folder if necessary, then click [Next].

   *It cannot be installed directly on a drive or on a server.

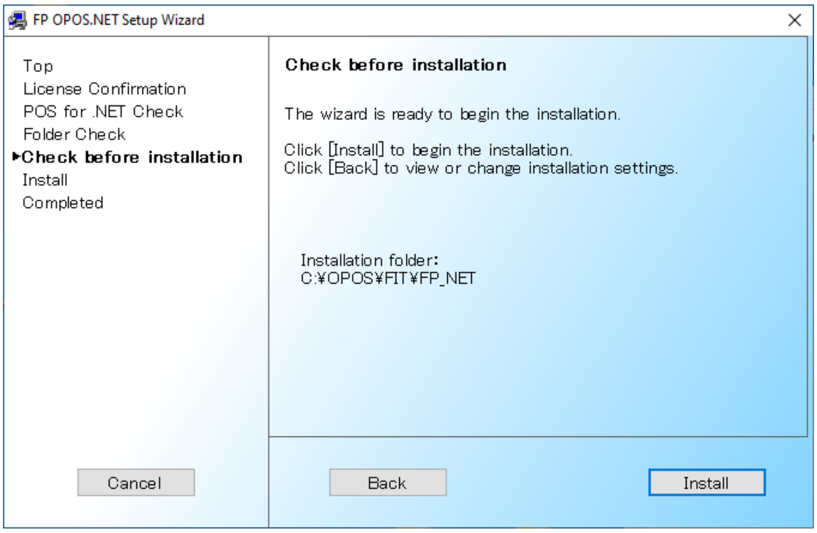


Check installation destination folder

Click [Next] to install to this folder.
To install to a different folder, click [Change].

C:¥OPOS¥FIT¥FP_NET

6. Review what you want to install, and then click [Install].



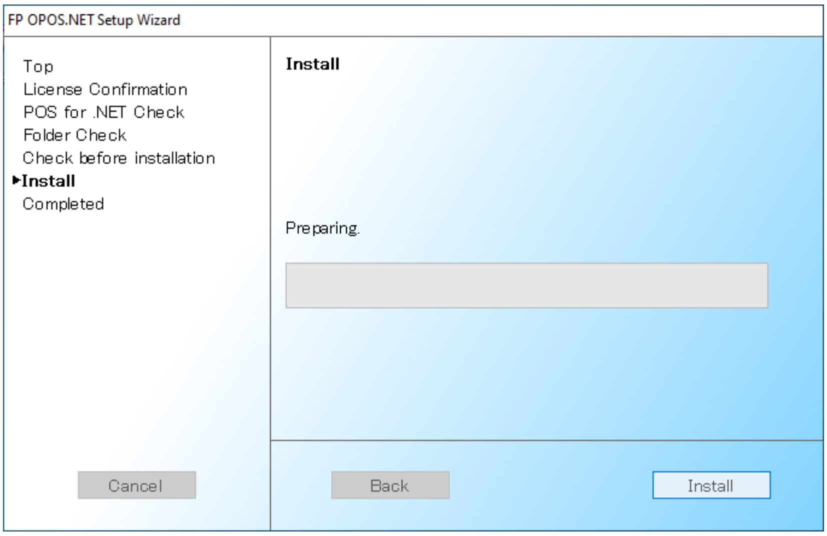
Check before installation

The wizard is ready to begin the installation.

Click [Install] to begin the installation.
Click [Back] to view or change installation settings.

Installation folder:
C:¥OPOS¥FIT¥FP_NET

7. The following screen will be displayed during installation.

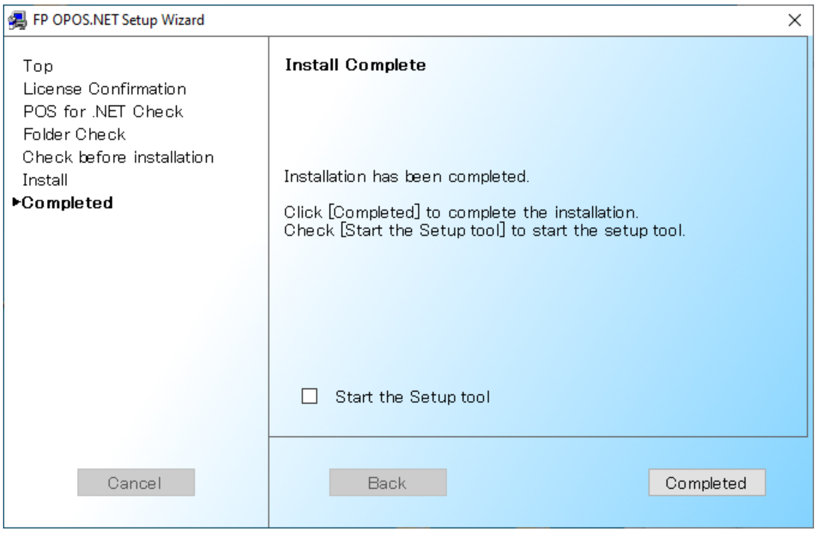


Install

Preparing.

8. Installation complete.

Check "Start the Setup tool" if necessary, then click [Completed] to exit the installer.



Install Complete

Installation has been completed.

Click [Completed] to complete the installation.
Check [Start the Setup tool] to start the setup tool.

☐ Start the Setup tool

2.2. Uninstallation Procedure

1. Follow the steps blow to display the uninstalling screen.

   【For Windows 10(ver1703~) / Windows Server 2019】

   ・Click the Windows logo button in the lower-left corner of the desktop screen.

   ・Click [Windows System] → [Control Panel] →[Program] → [Program and features]

   【For Windows 10(~ver1607) / Windows Server 2016】

   ・Right click the Windows logo button in the lower-left corner of the desktop screen and then click [Program and features].

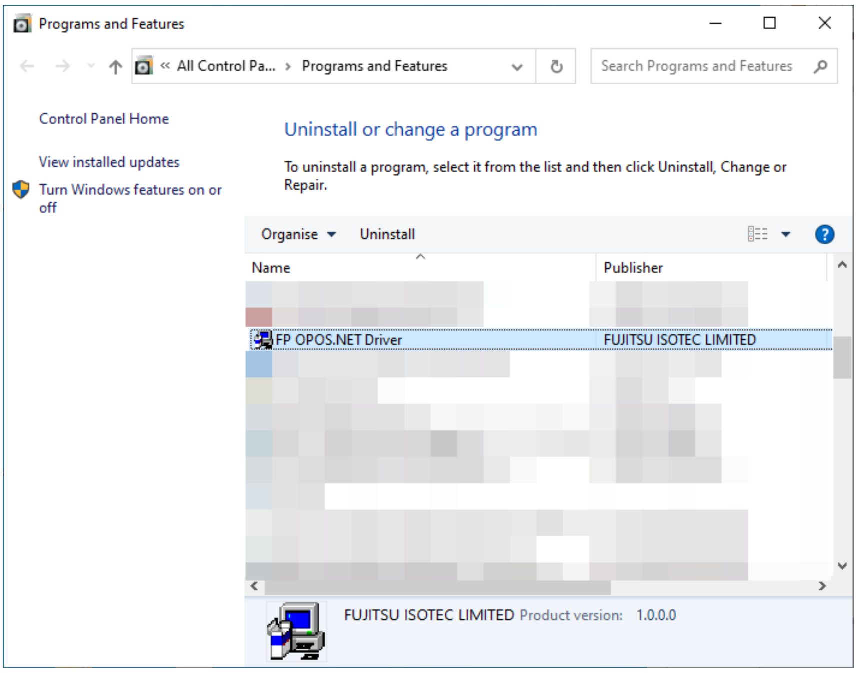
   【For Windows 8.1 / Windows Server 2012R2】

   ・Press the Windows logo key + X key, shortcut menu is displayed bottom left of the screen, and click [Programs and Features].

   【For Windows 7】

   ・Click the button indicated with the Windows logo at the left bottom of the desktop.

   ・Go to [Control Panel]→[Uninstall program] or [Uninstall a program].


Programs and Features
« All Control Pa... › Programs and Features
Search Programs and Features
Control Panel Home
View installed updates
Turn Windows features on or off
Organise
Uninstall
Name
Publisher

Uninstall or change a program

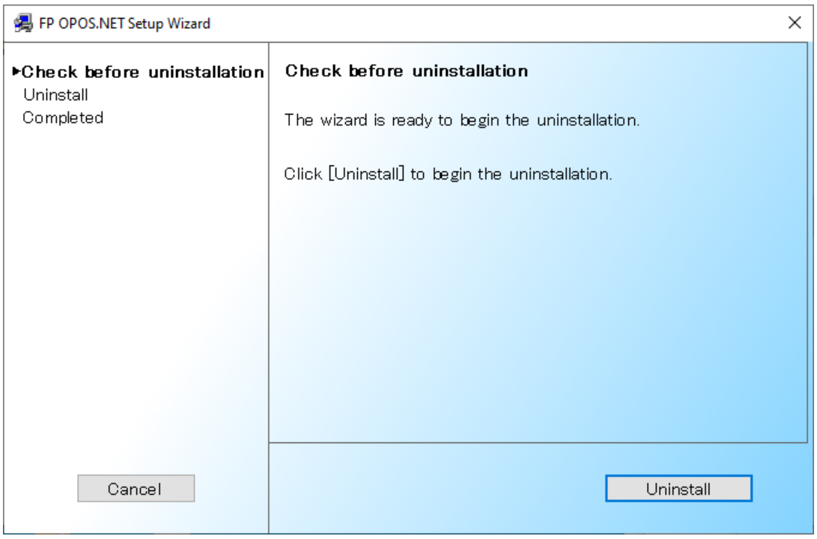
To uninstall a program, select it from the list and then click Uninstall, Change or Repair.

Name	Publisher
FP OPOS.NET Driver	FUJITSU ISOTEC LIMITED

FUJITSU ISOTEC LIMITED Product version: 1.0.0.0

2. Select "FP OPOS.NET Driver", then click "Uninstall".

3. The uninstall confirmation screen appears. Click [Uninstall].



FP OPOS.NET Setup Wizard

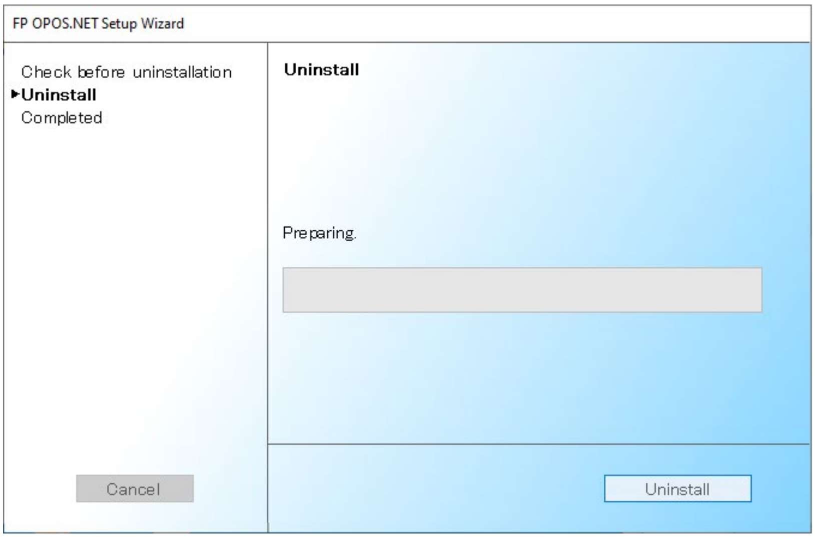
▸**Check before uninstallation**
Uninstall
Completed

Check before uninstallation

The wizard is ready to begin the uninstallation.

Click [Uninstall] to begin the uninstallation.

4. The uninstall is performed.



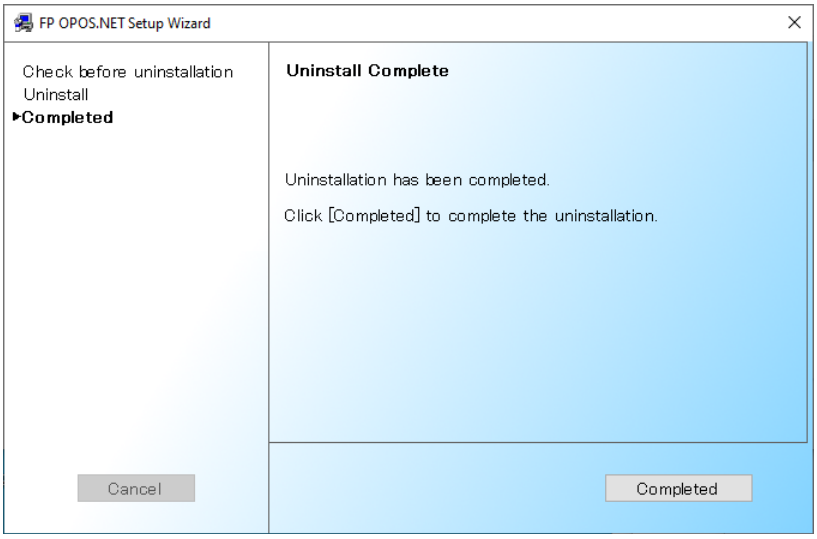
FP OPOS.NET Setup Wizard

Check before uninstallation
▸**Uninstall**
Completed

Uninstall

Preparing.

5. Uninstallation is complete.
   Click [Completed] to exit the uninstaller.



Uninstall Complete

Uninstallation has been completed.

Click [Completed] to complete the uninstallation.

6. Sometimes there are some files that could not be deleted by uninstaller in Installed folder ([System Drive]:\OPOS\FIT\FP_NET), so please delete manually.

That is all of uninstallation.

2.3. Installation File List

File groups of this class library are arranged just like the following.

[System Drive]:\OPOS\FIT\FP_NET

BitImage.dll	PrinterSO-dependent files. for bitmap printing.
BitImage_win32.dll	
CashDrawer.dll	Drawer Service Objects
NLog.dll	Object file for logger file library
PosPrinter.dll	POS Printer Service Objects
PrintController.exe	Printer control Module
SharedObject.dll	Object file for communication
Setup.exe Setup.xml UIData.xml	Installer for this class library

[System Drive]:\OPOS\FIT\FP_NET\ftfpcp
Win32\ftfpcp.dll
WIn64\ftfpcp.dll

[System Drive]:\OPOS\FIT\FP_NET\tdbm
Win32\tdbm.dll
WIn64\tdbm.dll

[System Drive]:\OPOS\FIT\FP_NET\APG
This APG.

[System Drive]:\OPOS\FIT\FP_NET\SampleProgram
Object : Sample program executable
Source : Sample program source

[System Drive]:\OPOS\FIT\FP_NET\SetupTool
Setup tool for this class library

2.4. Setting Program Usage

Register the printer information to use this driver and set up the printer.
When setting, be sure to connect only the printer you want to set. (This setting is not available when multiple printers are connected.)

1. Execution

If the [User Account Control] dialog box is displayed, click [Yes].

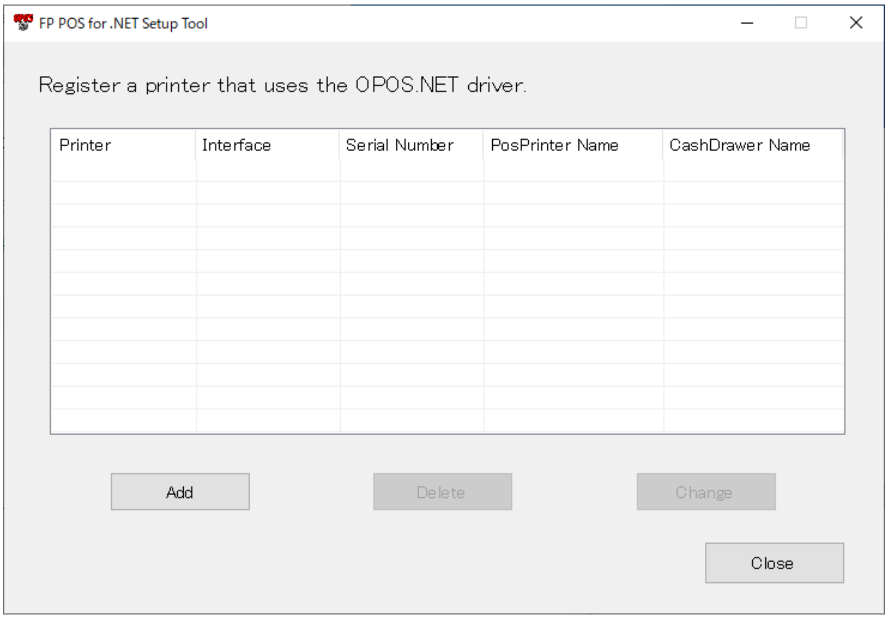
【For Windows 10 (ver1607~)/Windows Server 2016 and later】
1) Click the button indicated with the Windows logo ⊞ at the left bottom of the desktop.
2) Go to [FIT FP Series Printer]→[OPOS.NET Setup Tool].

【For Windows 10 (~ver1511)】
1) Click the button indicated with the Windows logo ⊞ at the left bottom of the desktop.
2) Go to [All apps]→[FIT FP Series Printer]→[OPOS.NET Setup Tool].

【For Windows 8.1 / Windows Server 2012 R2】
1) Move the mouse cursor in the Start screen, and click ⬇.
2) Click the [OPOS.NET Setup Tool] tile in the Apps view.

【For Windows 7】
1) Click the button indicated with the Windows logo ⊞ at the left bottom of the desktop.
2) Go to [All programs] →
[FIT FP Series Printer]→[OPOS.NET]→[OPOS.NET Setup Tool].

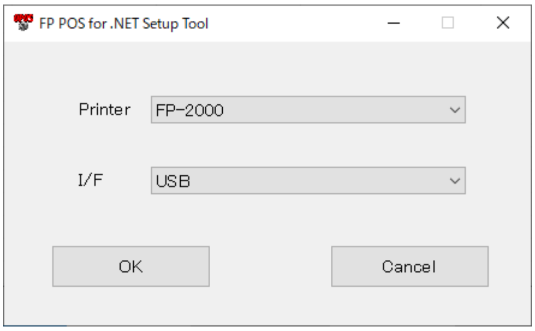
2. Checking Registered Printer Information



Register a printer that uses the OPOS.NET driver.

To add new printer information, click [Add].
To delete printer information, select the printer you want to delete from the registration list and click [Delete].
To change the printer information, select the printer you want to change from the registration list and click [Change].

3. Printer/Port Selection (Add)



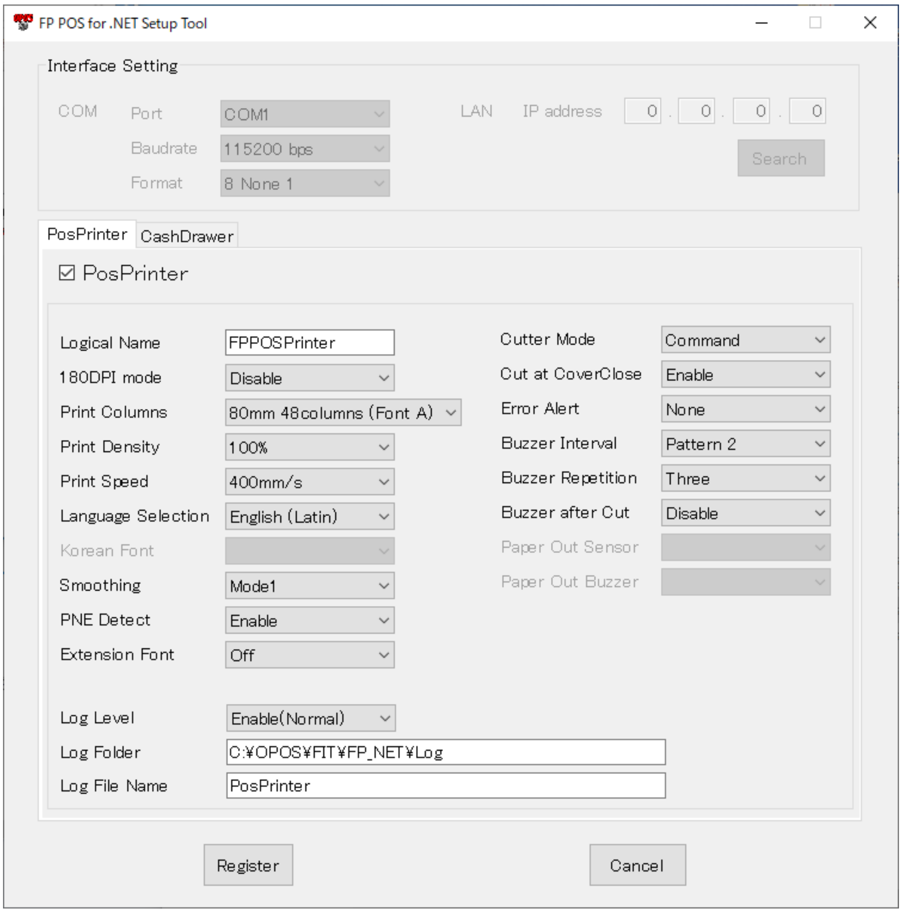
Printer: FP-2000
I/F: USB

Select the printer and interface you want to add, and then click [OK].
The number of devices that can be registered varies depending on the interface.

USB/COM: Up to 2
LAN: Up to 255

4. Advanced Setting(PosPrinter/CashDrawer)

(The screen is an example of FP-2200.)



FP POS for .NET Setup Tool

Interface Setting

COM

Port: COM1

Baudrate: 115200 bps

Format: 8 None 1

LAN

IP address: 0 . 0 . 0 . 0

☑ PosPrinter

Logical Name: FPPOSPrinter

180DPI mode: Disable

Print Columns: 80mm 48columns (Font A)

Print Density: 100%

Print Speed: 400mm/s

Language Selection: English (Latin)

Korean Font:

Smoothing: Mode1

PNE Detect: Enable

Extension Font: Off

Cutter Mode: Command

Cut at CoverClose: Enable

Error Alert: None

Buzzer Interval: Pattern 2

Buzzer Repetition: Three

Buzzer after Cut: Disable

Paper Out Sensor:

Paper Out Buzzer:

Log Level: Enable(Normal)

Log Folder: C:¥OPOS¥FIT¥FP_NET¥Log

Log File Name: PosPrinter

Set the interface to use and the PosPrinter/CashDrawer settings.

Check the setting to be added (PosPrinter/CashDrawer) and change the printer setting.

<An error factor>

- A printer other than the one you want to change is connected.
- A cable is not connected.
- The printer is not powered on.
- The cover is open.
- There is no paper.
- The port is already in use by other.
- The communication conditions do not accord with a printer. (Serial connection)
- The IP address is incorrect. (LAN connection)

List of settings

*The items that can be set differ depending on the printer.

Items			Description
Interface Setting			Configure the interface you want to use.
	COM	Port	Specifies the port number.
		Baudrate	Specifies the communication speed. It must be set to the same baud rate as the printer settings.
		Format	Specifies the communication format. Must be set to the same format as the printer settings.
	LAN	IP address	Set the IP address of the printer. You can also search for and specify a printer from [Search].
PosPrinter			Check to register a PosPrinter.
	Logical Name		Specifies the logical device name of the PosPrinter.
	180DPI mode		Sets the DPI mode. Select Enable to enable the 180 DPI mode.
	Print Columns		Specifies the width of the paper and the number of characters that can be printed on a single line.
	Print Density		Specifies the print density.
	Print Speed		Specifies the print speed. Printing speed varies depending on the printer model.
	Language Selection		Specifies the language specification of the printer to print.
	Korean Font		Specifies the typeface for Korean fonts.
	Smoothing		Specifies the smoothing function. Smoothing is applied when printing fonts and the RecLetterQuality property is set to TRUE.
	PNE Detect		Specifies a paper near end notification.
	Extension Font		Specifies the extended font.
	Cutter Mode		Specifies the paper cut mode.
	Cut at Cover Close		Specifies the cover and the cutting behavior when closed.
	Error Alert		Specifies the buzzer sound when an error occurs.
	Buzzer Interval		Specifies the buzzer interval pattern.
	Buzzer Repetition		Specifies the number of times the buzzer sounds.
	Buzzer after Cut		Specifies the buzzer sound after cutting the paper.
	Paper Out Sensor		Specifies the paper out sensor function.
	Paper Out Buzzer		Specifies the buzzer to sound when the printer is waiting for paper removal.
	Log Level Log Folder Log File Name		Specifies the logging function of this driver. See "7. Log Files" for details.

Items		Description
CashDrawer		Check to register CashDrawer.
	Logical Name	Specifies the logical device name of the CashDrawer.
	Drawer No	Specifies the drawer number.
	Drawer Status	Specifies the drawer status. Set when the event of the drawer open close is reversed.
	Log Level Log Folder Log File Name	Specifies the logging function of this driver. See "7. Log Files" for details.

3. Using OPOS.NET Class Lirary

3.1. Common

The application uses the this class library in the steps as follows:

1. **Open** method: Called to link the control object to the service object.
2. **Claim** method: Called to enable exclusive access to the device. For the device of exclusive use, this method is required, and foe the device of sharable use, it is optional.
3. **DeviceEnabled** property: Set to **TRUE** to operate the device.
4. Use the device. (Each property, method, event)
5. **DeviceEnabled** property: Set to **FALSE** to disable the device.
6. **Release** method: Called to clear exclusive access to the device.
7. **Close** method: Called to release the service object from the control object.

For details of other usage, refer to "OPOS-APG V1.13" document.

3.2. POS Printer

The POS printer supports only "Receipt." For the methods and properties of other than that (Journal or Slip), interface is supplied but behavior is not supported.
According to the general output model, synchronous and asynchronous output is available for the POS printer.
The POS printer is the device to be used exclusively.

3.3. Drawer

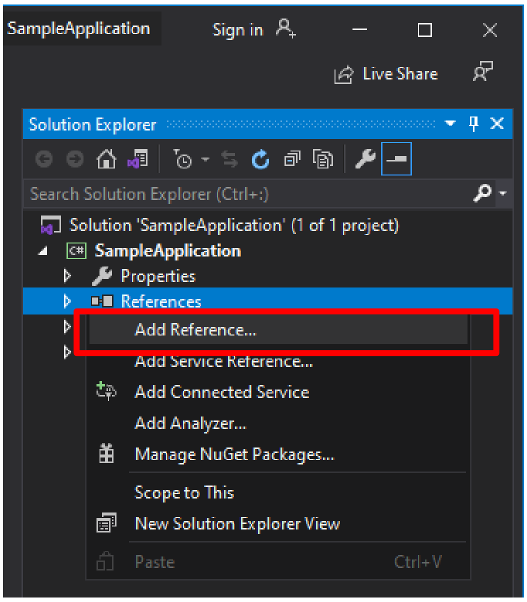
The Drawer can be used in the same way as the POS printer, but all features are executable without executing the Claim method. However, when exclusive permission is acquired for particular application by the Claim method, the OpenDrawer method cannot be executed by the application enabled with the same name. If there is no application with exclusive permission, this is not the case.

3.4. How to Implement the OPOS.NET Class Library

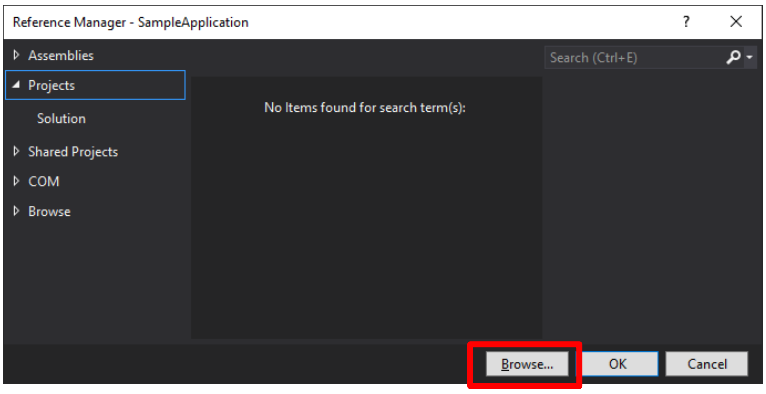
The following steps describe how to implement this class library control in the application.

To Implement Using Visual Stadio 2019 (C#)

1. Start Visual Studio 2019 and select the project type you want to create. In this example, [Windows Forms App (.NET Framework)] is selected. After selecting, right-click the “References” in [Solution Explorer] and select [Add Reference…].

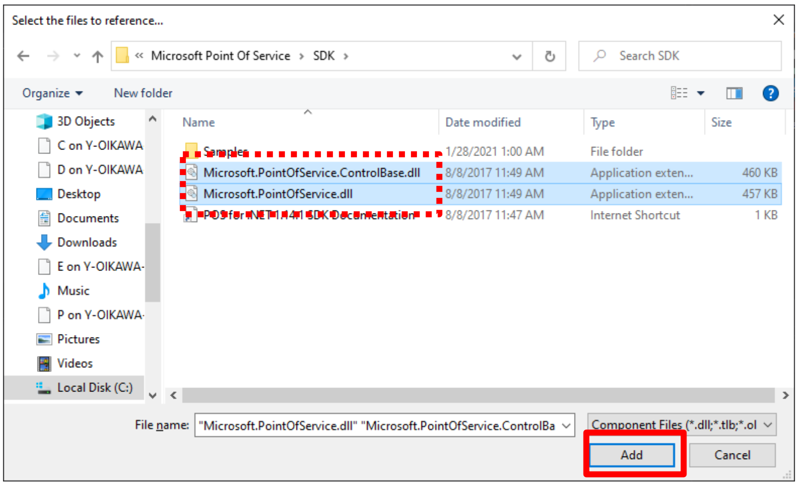


2. When the Reference Manager is displayed, select [Browse…].



No Items found for search term(s):

3. Select the following file in “C:¥Program Files (x86)¥Microsoft Point Of Service¥SDK(*1)” and click [Add].
   ・Microsoft.PointOfService.ControlBase.dll (*2)
   ・Microsoft.PointOfService.dll (*2)
   *1 : 32bit : ” C:¥Program Files¥Microsoft Point Of Service¥SDK”
   *2 : If you do not have the file, install POS for.NET again and add "SDK".



Select the files to reference...

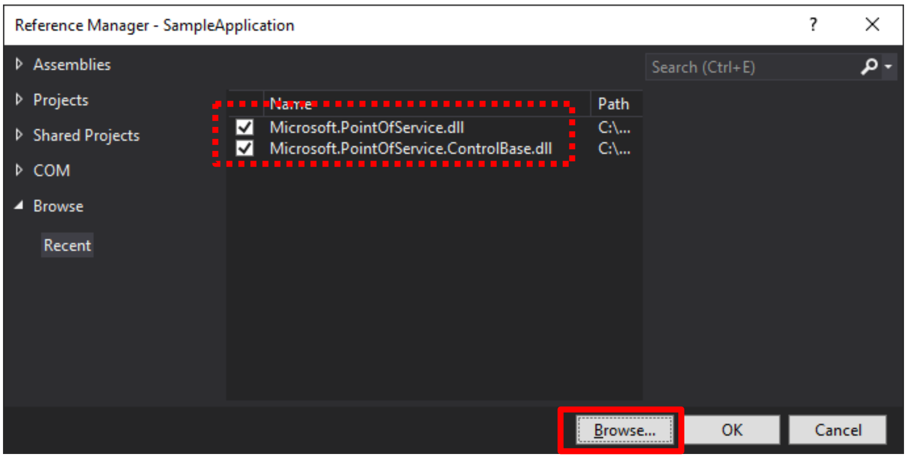
« Microsoft Point Of Service › SDK ›

Name	Date modified	Type	Size
Samples	1/28/2021 1:00 AM	File folder	
Microsoft.PointOfService.ControlBase.dll	8/8/2017 11:49 AM	Application exten...	460 KB
Microsoft.PointOfService.dll	8/8/2017 11:49 AM	Application exten...	457 KB
POS for .NET 1.14.1 SDK Documentation	8/8/2017 11:47 AM	Internet Shortcut	1 KB

File name: "Microsoft.PointOfService.dll" "Microsoft.PointOfService.ControlBa

Component Files (*.dll;*.tlb;*.ol

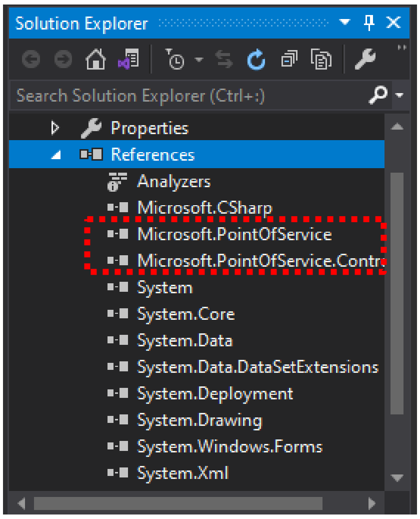
4. Make sure that you have added the file to the Reference Manager, check it and click [OK].



Reference Manager - SampleApplication

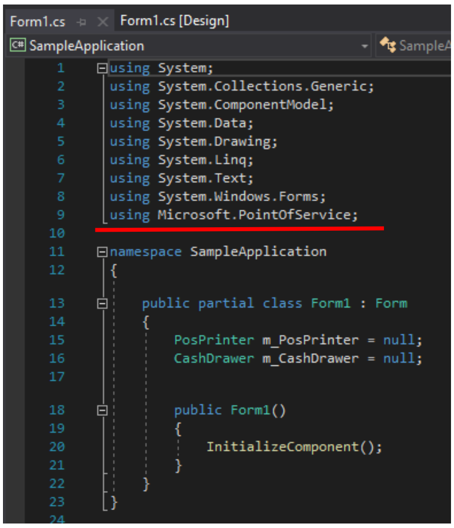
	Name	Path
☑	Microsoft.PointOfService.dll	C:\...
☑	Microsoft.PointOfService.ControlBase.dll	C:\...

5. Open "References" under [Solution Explorer] and verify that “Microsoft.PointOfService” and “Microsoft.PointOfService.ControlBase” have been added.



Properties
References
Analyzers
Microsoft.CSharp
Microsoft.PointOfService
Microsoft.PointOfService.Contr
System
System.Core
System.Data
System.Data.DataSetExtensions
System.Deployment
System.Drawing
System.Windows.Forms
System.Xml

6. Please add "using Microsoft.PointOfService" in the source code and use it.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.PointOfService;

namespace SampleApplication
{
    public partial class Form1 : Form
    {
        PosPrinter m_PosPrinter = null;
        CashDrawer m_CashDrawer = null;

        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

3.5. Exception

With the exception of accessing the properties DeviceControlVersion, DeviceControlDescription, and State, all POS for .NET method calls and property accesses can throw a PosControlException on failure.

Name	**Description**
ErrorCode	Error code indicating the cause of the error exception. Refer to the error code list.
ErrorCodeExtended	The extended error code that indicates the cause of the error exception. This may contain service-specific values.

Error code list

Remarks

Each property sets this property. It is set when it obtains property, or when it set writable property.

This property is always readable. It returns Closed**(101)** till it calls up **Open** Method. The ErrorCode values are just the following.

Value	Meaning
Closed (101)	You try to access the closed device.
NotClaimed (103)	You try to access Exclusive Use Device for which you need to obtain exclusive access right before using method/property setting process.
NoService (104)	Control cannot communicate to Service Object. Perhaps, you must correct Set Up Error or Configuration Error.
Disabled (105)	It cannot be executed when Device is disabled.
Illegal (106)	You try to execute unavailable operation to Device or operation not supported. Or you use unavailable parameter value.
NoHardware (107)	POS Printer is OFF or OFFLINE.
Failure (111)	Even though Device is connected to System, Power is ON, and it is ONLINE, Device cannot execute the requested process.
Timeout (112)	The Service Object waiting for the response from Device does time-out, or the Control waiting for the response from Service Object does time-out.
Busy (113)	Present SO state cannot accept this request. For example, when asynchronous Output is executed, some methods cannot accept it.
Extended (114)	Particular Error State occurs. Error State Code can be confirmed with **ErrorCodeExtended** Property

Extended error code list

Remarks

When **ErrorCode** is **Extended(114)**, particular-to-class error information value described in device class explanation to this property is set.
When **ErrorCode** is another value**,** Service Object can set particular-to-SO value to this property. These values have meaning only when Application processes them adding particular-to-SO values.

This class library has the following values.

ErrorCodeExtended	Constant Name(Contents)	Details
201	ExtendedErrorCoverOpen	Printer Cover is open. To recover from Error, close the cover. In this case, you don't have to make OPOS disable, **Release**, **Close**. If Cover opens in printing, after closing cover, the not-yet-printed data will be printed. However, there is the possibility of interrupted printing space among the printing data.
203	ExtendedErrorReceiptEmpty	Run-out-of-receipt-paper. To recover from error, feed receipt paper. At this time, there is no need to make OPOS disable, and **Release**, **Close**. If run-out-of-paper happens in printing, the printing data not yet printing will be printed after feeding receipt paper.
206	ExtendedErrorTooBig	Bitmap is too big for the Printer to process. Check the bitmap file.
207	ExtendedErrorBadFormat	The format is not correct as a bitmap file. There is a possibility that the file name is not correct or the contents of the file are corrupt.
281	BadFile	The format is not correct as a firmware file. There is a possibility that the file name is not correct or the contents of the file are corrupt.

ErrorCodeExtended	Constant Name(Contents)	Details
10001	Blackmark	Black Mark Sensor Error happens. There is the possibility that paper cannot respond to Black Mark. At such time, there is no need to make OPOS disable, and Release, Close. Change the receipt to the one responding to Black Mark. *It does not occur when Black Mark Error Sensor is not on. (Check Black Mark Enable.)
10003	Fatal	Fatal error occurs in Printer. There is no recover way. Do OPOS **Close**.
10006	Overheat	Printer Header Overheat Error happened. To recover from the error, leave it for a while until the head temperature cools down and the error does not occur again. At such time, there is no need to make OPOS disable, and **Release**, **Close**. If Head Overheat occurs in printing, the printing data not yet printing will be printed after the head temperature recovery. However, there is the possibility of interrupted space of printing of the printing data.
10008	Cutterjam	Cutter Jam Error happened. To recover from the error, open the cover and remove the cause (such as the jammed paper), and close the cover. If the cover does not open, follow procedure on Printer UsersManual.

4. OPOS Interface Specifications (Printer)

4.1. List

Properties

Common	Type	Access	May Use After	Initial Value, Conditions
CapCompareFirmwareVersion	bool	R	Open	TRUE
CapPowerReporting	PowerReporting	R	Open	Standard (1)
CapStatisticsReporting	bool	R	Open	FALSE
CapUpdateStatistics	bool	R	Open	FALSE
CheckHealthText	string	R	Open	“”
Claimed	bool	R	Open	FALSE
DeviceEnabled	bool	R/W	Open&Claim	FALSE Made writable after Open and Claim.
FreezeEvents	bool	R/W	Open	FALSE Made writable after Open.
OutputId	int	R	Open	1
PowerNotify	PowerNotification	R/W	Open	Disabled (0) Made writable after Open, and unwritable after Enabled.
PowerState	PowerState	R	Open	Unknown (2000)
State	ControlState	R	--	Closed (1)
ServiceObjectDescription	string	R	Open	“Fujitsu Isotec FP POS Printer Service Object”
ServiceObjectVersion	Version	R	Open	1013XXX
DeviceDescription	string	R	Open	“FP 1 Station Thermal POSPrinter (C)20xx Fujitsu Isotec Limited”
DeviceName	string	R	Open	“FP 1 Station Thermal POSPrinter”

Specific	**Type**	**Access**	**May Use After**	**Initial Value, Conditions**
CapCharacterSet	CharacterSetCapability	R	Open	Kanji (11)
CapConcurrentJrnRec	bool	R	Open	FALSE
CapConcurrentJrnSlp	bool	R	Open	FALSE
CapConcurrentPageMode	bool	R	Open	FALSE
CapConcurrentRecSlp	bool	R	Open	FALSE
CapCoverSensor	bool	R	Open	TRUE
CapMapCharacterSet	bool	R	Open	TRUE
CapTransaction	bool	R	Open	TRUE
CapJrnPresent	bool	R	Open	FALSE
CapJrn2Color	bool	R	Open	FALSE
CapJrnBold	bool	R	Open	FALSE
CapJrnDhigh	bool	R	Open	FALSE
CapJrnDwide	bool	R	Open	FALSE
CapJrnDwideDhigh	bool	R	Open	FALSE
CapJrnEmptySensor	bool	R	Open	FALSE
CapJrnItalic	bool	R	Open	FALSE
CapJrnNearEndSensor	bool	R	Open	FALSE
CapJrnUnderline	bool	R	Open	FALSE
CapJrnCartridgeSensor	PrinterCartridgeSensors	R	Open	None(0)
CapJrnColor	PrinterColors	R	Open	None(0)
CapRecPresent	bool	R	Open	TRUE
CapRec2Color	bool	R	Open	The initial value may vary according to the contents of the PosExplorer.
CapRecBarCode	bool	R	Open	TRUE
CapRecBitmap	bool	R	Open	TRUE
CapRecBold	bool	R	Open	TRUE
CapRecDhigh	bool	R	Open	TRUE
CapRecDwide	bool	R	Open	TRUE
CapRecDwideDhigh	bool	R	Open	TRUE
CapRecEmptySensor	bool	R	Open	TRUE
CapRecItalic	bool	R	Open	FALSE
CapRecLeft90	bool	R	Open	TRUE
CapRecNearEndSensor	bool	R	Open	The initial value may vary according to the contents of the PosExplorer.
CapRecPapercut	bool	R	Open	TRUE
CapRecRight90	bool	R	Open	TRUE
CapRecRotate180	bool	R	Open	TRUE
CapRecStamp	bool	R	Open	FALSE
CapRecUnderline	bool	R	Open	TRUE

Specific	**Type**	**Access**	**May Use After**	**Initial Value, Conditions**
CapRecCartridgeSensor	PrinterCartridgeSensors	R	Open	None(0)
CapRecColor	PrinterColors	R	Open	None(0)
CapRecMarkFeed	PrinterMarkFeeds	R	Open	None(0)
CapRecPageMode	bool	R	Open	TRUE
CapSlpPresent	bool	R	Open	FALSE
CapSlpFullslip	bool	R	Open	FALSE
CapSlp2Color	bool	R	Open	FALSE
CapSlpBarCode	bool	R	Open	FALSE
CapSlpBitmap	bool	R	Open	FALSE
CapSlpBold	bool	R	Open	FALSE
CapSlpDhigh	bool	R	Open	FALSE
CapSlpDwide	bool	R	Open	FALSE
CapSlpDwideDhigh	bool	R	Open	FALSE
CapSlpEmptySensor	bool	R	Open	FALSE
CapSlpItalic	bool	R	Open	FALSE
CapSlpLeft90	bool	R	Open	FALSE
CapSlpNearEndSensor	bool	R	Open	FALSE
CapSlpRight90	bool	R	Open	FALSE
CapSlpRotate180	bool	R	Open	FALSE
CapSlpUnderline	bool	R	Open	FALSE
CapSlpBothSidesPrint	bool	R	Open	FALSE
CapSlpCartridgeSensor	PrinterCartridgeSensors	R	Open	None(0)
CapSlpColor	PrinterColors	R	Open	None(0)
CapSlpPageMode	bool	R	Open	FALSE
CapSlpRuledLine	LineDirection	R	Open	None(0)
AsyncMode	bool	R/W	Open	FALSE
CartridgeNotify	PrinterCartridgeNotify	R/W	Open	Disabled (0) Unwritable
CharacterSet	int	R/W	Open,Claim & Enable	The initial value may vary according to the contents of the PosExplorer.
CharacterSetList	int[]	R	Open	The initial value may vary according to the contents of the PosExplorer.
CoverOpen	bool	R	Open,Claim & Enable	FALSE
ErrorLevel	PrinterErrorLevel	R	Open	None(1)
ErrorStation	PrinterStation	R	Open	None(0)
ErrorString	string	R	Open	“”
FontTypefaceList	string[]	R	Open	“Arial, Times New Roman”
FlagWhenIdle	bool	R/W	Open	FALSE Made writable after Open

Specific	**Type**	**Access**	**May Use After**	**Initial Value, Conditions**
MapCharacterSet	bool	R/W	Open	TRUE
MapMode	MapMode	R/W	Open	Dots (1) Made writable after Open
PageModeArea	System.Drowing.Point	R	Open	“”
PageModeDescriptor	PageModeDescriptors	R	Open	Bitmap + Barcode + BitmapRotate + BarcodeRotate
PageModeHorizontalPosition	int	R/W	Open	0
PageModePrintArea	System.Drowing.Rectangle	R/W	Open	Rectangle(0,0)
PageModePrintDirection	PageModePrintDirection	R/W	Open	None(0)
PageModeStation	PrinterStation	R/W	Open	None(0)
PageModeVerticalPosition	int	R/W	Open	0
RotateSpecial	Rotation	R/W	Open	Normal (1) Made writable after Open
JrnLineChars	int	R/W	Open,Claim & Enable	0 Unwritable
JrnLineCharsList	int[]	R	Open	
JrnLineHeight	int	R/W	Open,Claim & Enable	0 Unwritable
JrnLineSpacing	int	R/W	Open,Claim & Enable	0 Unwritable
JrnLineWidth	int	R	Open,Claim & Enable	0
JrnLetterQuality	bool	R/W	Open,Claim & Enable	FALSE Unwritable
JrnEmpty	bool	R	Open,Claim & Enable	FALSE
JrnNearEnd	bool	R	Open,Claim & Enable	FALSE
JrnCartridgeState	PrinterCartridgeStates	R	Open,Claim & Enable	OK(0)
JrnCurrentCartridge	PrinterColors	R/W	Open,Claim & Enable	None(0) Unwritable
RecLineChars	int	R/W	Open,Claim & Enable	The initial value may vary according to the contents of the PosExplorer. Made writable after Open.

Specific	**Type**	**Access**	**May Use After**	**Initial Value, Conditions**
RecLineCharsList	int[]	R	Open	The initial value may vary according to the contents of the PosExplorer.
RecLineHeight	int	R/W	Open,Claim & Enable	The initial value may vary according to the contents of the PosExplorer. Unwritable
RecLineSpacing	int	R/W	Open,Claim & Enable	The initial value may vary according to the contents of the PosExplorer. Made writable after Open.
RecLineWidth	int	R	Open,Claim & Enable	The initial value may vary according to the contents of the PosExplorer.
RecLetterQuality	bool	R/W	Open,Claim & Enable	TRUE Made writable after Open.
RecEmpty	bool	R	Open,Claim & Enable	FALSE
RecNearEnd	bool	R	Open,Claim & Enable	FALSE
RecSidewaysMaxLines	int	R	Open,Claim & Enable	The initial value may vary according to the contents of the PosExplorer.
RecSidewaysMaxChars	int	R	Open,Claim & Enable	The initial value may vary according to the contents of the PosExplorer.
RecLinesToPaperCut	int	R	Open,Claim & Enable	The initial value may vary according to the contents of the PosExplorer.
RecBarCodeRotationList	Rotation[]	R	Open	Normal (0), Right90(257), Left90(258), Rotate180(259)
RecCartridgeState	PrinterCartridgeStates	R	Open,Claim & Enable	Unknown(68435456)
RecCurrentCartridge	PrinterColors	R/W	Open,Claim & Enable	None(0) Unwritable
RecBitmapRotationList	Rotation[]	R	Open	Normal (0), Right90(257), Left90(258), Rotate180(259)
SlpLineChars	int	R/W	Open,Claim & Enable	0 Unwritable
SlpLineCharsList	int[]	R	Open	

Specific	**Type**	**Access**	**May Use After**	**Initial Value, Conditions**
SlpLineHeight	int	R/W	Open,Claim & Enable	0 Unwritable
SlpLineSpacing	int	R/W	Open,Claim & Enable	0 Unwritable
SlpLineWidth	int	R	Open,Claim & Enable	0
SlpLetterQuality	bool	R/W	Open,Claim & Enable	FALSE Unwritable
SlpEmpty	bool	R	Open,Claim & Enable	FALSE
SlpNearEnd	bool	R	Open,Claim & Enable	FALSE
SlpSidewaysMaxLines	int	R	Open,Claim & Enable	0
SlpSidewaysMaxChars	int	R	Open,Claim & Enable	0
SlpMaxLines	int	R	Open,Claim & Enable	0
SlpLinesNearEndToEnd	int	R	Open,Claim & Enable	0
SlpBarCodeRotationList	Rotation[]	R	Open	
SlpPrintSide	int	R	Open,Claim & Enable	0
SlpCartridgeState	int	R	Open,Claim & Enable	0
SlpCurrentCartridge	int	R/W	Open,Claim & Enable	0 Unwritable
SlpBitmapRotationList	Rotation[]	R	Open	

* In the Access column, R indicates Read-Only, R/W indicates Read/Write. The item in May Use After is the method and property required for initialization, Open indicates the Open method, Claim indicates the Claim method and Enable indicates setting the DeviceEnabled property to TRUE. If required procedure is not executed, the error may be set in the ErrorCode property. When May Use After is Open & Claim or Open, Claim & Enable, the property is available for acquisition after the Open method is executed, but the value may not be initialized until all Open, Claim & Enable are executed. To acquire such property, access it after the conditions are met.

List of Methods

Common	**Initialization**
Open	--
Close	Open
Claim	Open
Release	Open, Claim
ClearOutput	Open, Claim & Enable
CheckHealth	Open, Claim & Enable
DirectIO	Open, Claim & Enable
ResetStatistics	Open, Claim & Enable
RetrieveStatistics	Open, Claim & Enable
UpdateStatistics	Open, Claim & Enable

Specific	**Initialization**
PrintNormal	Open, Claim & Enable
PrintTwoNormal	Open, Claim & Enable
PrintImmediate	Open, Claim & Enable
BeginInsertion	Open, Claim & Enable
EndInsertion	Open, Claim & Enable
BeginRemoval	Open, Claim & Enable
EndRemoval	Open, Claim & Enable
CutPaper	Open, Claim & Enable
RotatePrint	Open, Claim & Enable
PrintBarCode	Open, Claim & Enable
PrintBitmap	Open, Claim & Enable
TransactionPrint	Open, Claim & Enable
ValidateData	Open, Claim & Enable
SetBitmap	Open, Claim & Enable
SetLogo	Open, Claim & Enable
ChangePrintSide	Open, Claim & Enable
MarkFeed	Open, Claim & Enable
ClearPrintArea	Open, Claim & Enable
PageModePrint	Open, Claim & Enable
PrintMemoryBitmap	Open, Claim & Enable

List of Events

Event	Initialization
DirectIOEvent	Open, Claim & Enable
ErrorEvent	Open, Claim & Enable
OutputCompleteEvent	Open, Claim & Enable
StatusUpdateEvent	Open, Claim & Enable

4.2. Printing Data and Escape Sequence

This class library supports the following Escape Sequence.

1) Escape Sequence which operates only when assigned time

Name	Data	Contents
Paper Cut	ESC |#P	It cuts the receipt paper. The character '#' is the character string of ASCII decimal number which shows the percentage of required cutting. It is possible to omit '#'. When the value is between '1' to '99', partial cutting is performed. When the value is '100' or omitted, full cutting is performed. When the value is other than any value between '1' to '100', it is ignored In addition to that, if there are data buffered to POS Printer, (in the case that POS Printer does not print even though printing request is done) it cannot do cutting. To execute cutting, it must be at the head of the line. It is also unavailable in 90 degrees rotating to the left or to the right by **RotatePrint** Method, and back in operation after clearing 90 degrees rotating to the left or to the right.
Feed and Paper Cut	ESC |#fP	It cuts the receipt paper after feeding the paper for the lines of **RecLinesToPaperCut**. Character '#' is defined by "Paper Cut" Escape Sequence. In addition to that, if there are data buffered to POS Printer, (in case that POS Printer does not print even though printing request is done) it cannot do cutting. To execute cutting, it must be at the head of the line. It is also unavailable in 90 degrees rotating to the left or to the right by **RotatePrint** Method, and back in operation after clearing 90 degrees rotating to the left or to the right.
Feed, Paper Cut, and Stamp	ESC |#sP	It is not supported.

Name	Data	Contents
Print bitmap	ESC |#B	It prints the bitmap saved by **SetBitmap** Method. '#' is the Bitmap number and supports 20 bitmap printing '1' to '20'. It is possible to change printing quality by changing **RecLetterQuality** property value in printing. As for handling printing quality, it is same as **PrintBitmap** Method. When '#' is omitted, it is handled as character string. Printing is available during 90 degrees rotating to the left or to the right by the **RotatePrint** method, but printing may not be performed properly because printing area of the bitmap size is not calculated. When '#' is omitted, it is regarded character string data starting with the character "B". When the number that is not stored in the **SetBitmap** method, the print command is issued to the printer, but printing is not performed.
Print top logo	ESC |tL	It prints the top logo saved by **SetLogo** Method.
Print bottom logo	ESC |bL	It prints the bottom logo saved by **SetLogo** Method.
Fire stamp	ESC |sL	It is not supported.
Feed line	ESC |#lF	Feeds the paper forward by lines. The character '#' is replaced by an ASCII decimal string telling the number of lines to be fed. If '#' is omitted, then one line is fed. '#' supports the values from '1' to '255'. (If the value is smaller than '1', the command is not executed, and if the value is larger than '255', the command is executed regarding that '255' is assigned.) If print data is not presence, line feed operation is executed according to the amount of line feed, and if print data is presence, the height of the print data is fed. If the value specified for "#" exceeds 35.4 in (approx. 900 mm), the command is executed feeding the paper by 35.4 in (approx. 900 mm).. In 90 degrees rotating to the left or to the right by **RotatePrint** Method, it prints next printing location after Returns of feed-assigned lines.

Name	**Data**	**Contents**
Feed unit	ESC |#uF	Feeds the paper forward by the units defined with **MapMode**. The character '#' is replaced by an ASCII decimal string telling the number of units to be fed. If '#' is omitted, then one unit for each **MapMode** is fed. MapMode = Dots(1) '#' supports the values from '1' to '127'. (If the value is smaller than '1', the command is not executed, and if the value is larger than '127', the command is executed regarding that '127' is assigned.) MapMode = Twips(2) '#' supports the values from '1' to '903'. (If the value is smaller than '1', the command is not executed, and if the value is larger than '903', the command is executed regarding that '903' is assigned.) MapMode = English(3) '#' supports the values from '1' to '627'. (If the value is smaller than '1', the command is not executed, and if the value is larger than ''627', the command is executed regarding that ''627' is assigned.) MapMode = Metric(4) '#' supports the values from '1' to '1594'. (If the value is smaller than '1', the command is not executed, and if the value is larger than '1594', the command is executed regarding that '1594' is assigned.) The line feed setting of the printer does not affect the amount of line feed. It is executed on the halfway of the line, and when the specified amount of line feed is less than 1 line, then 1 line is fed In 90 degrees rotating to the left or to the right by **RotatePrint** Method, it will print after the interval of the value of next printing location assigned by unit feed.
Feed reverse	ESC |#rF	It is not supported.

Name	**Data**	**Contents**
Send embedded data	ESC |#rE	The successive character string of "#E" is passed to the device without any change. The character '#' is replaced by an ASCII decimal string specifying the number of bytes following the escape sequence to be passed directly to the device. If '#' is omitted, it is not regarded as the escape sequence, and handled as print data. Character string cannot output a control code and the cord of 80H - FFH as expected. In this case, set it in either of Nibble(1), and set printing data. When the print data specified by '#' is not set after the escape sequence is specified, available print data is sent. (Example: When ESC|2E"a" is specified, only "a" is sent because the character string is set only for one byte.) In rotate printing 90 degrees to the left or to the right by the **RotatePrint** method, the data column specified by Send embedded data is not counted as the character string, the width cannot be calculated. Adjust the printing width by inserting empty space and so on.
Barcode printing (Refer to the next page)	ESC |#rR	Prints the barcode. The character '#' is replaced by an ASCII decimal string and the number of characters following the R to use in the definition of the characteristics of the barcode to be printed. See details below. The barcode may be printed during rotate printing 90 degrees to the left or to the right by the **RotatePrint** method, but printing may not be performed normally because the print area is not calculated by the specified barcode width. When the other character string data specified exceeds the barcode width, printing is executed. In the **RotatePrint** method, please use it without specifying **BarCode**. Even if the **RotateSpecial** property is specified, the rotation print is not done. The available width that can be set by the parameter is up to the value of the **RecLineWidth** property and is not affected by the **RotateSpecial** property.

Starting with Release 1.13, the application can use the ESC|#R escape sequence to print barcodes in-line with other print commands. The character '#' is the number of characters following the R to use in the definition of the characteristics of the barcode to be printed.

In the data following the R, other lower case letters and numbers are used to identify different values. The same value definitions as defined for the printBarCode method headers and definitions are used for the various barcode values. Converting to string the values from the definitions are consistent.

The attribute symbols are defined as follows:

s	symbology
h	height
w	width
a	alignment
t	human readable text position
d	start of data
e	end of sequence

The attributes must appear in the order specified in the above list. (It cannot be omitted)

Using a basic UPCA, center aligned, with bottom text, 200 dots height and 400 dots wide, the command is as follows:

```
ESC|33Rs101h200w400a-2t-13d123456789012e
```

In addition, the threshold of each parameter is as follows. When the threshold exceeded it, the barcode is not printed.

Barcode	Width(dot)	Height(dot)	Alignment
Except two dimension code	The most narrow width of the individual barcod – **RecLineWidth**	1 - 255	All values defined by **PrintBarcode** method are available.
PDF417	172 - **RecLineWidth**	12 - 831	All values defined by **PrintBarcode** method are available.
PDF417 During right and left 90 degrees turn by **RotatePrint** method.	172 - 831	12 – **RecLineWidth**	As well as an appointed value, all values become the left hotchpotch
QR	21 - **RecLineWidth**	1 – 16 (as width of the module)	All values defined by **PrintBarcode** method are available.
QR During right and left 90 degrees turn by **RotatePrint** method.	21 - **RecLineWidth**	1 – 16 (as width of the module)	As well as an appointed value, all values become the left hotchpotch
MicroQR	11 - RecLineWidth	1 – 16 (as width of the module)	All values defined by PrintBarcode method are available.
MicroQR During right and left 90 degrees turn by RotatePrint method.	11 - RecLineWidth	1 – 16 (as width of the module)	As well as an appointed value, all values become the left hotchpotch

2) Escape Sequence Operating in Printing

It has characteristics that are remembered until explicitly changed.

No such escape sequence.

3) Escape Sequence Operating at the Time of Printing

It has the characteristics that are reset at the end of each print method, by an explicit reset (where applicable), or by a “Normal” sequence.

Name	Data	Contents
Bold	ESC |bC	Prints in bold. If ‘!’ is specified then bold is disabled.
Underline	ESC |#uC	Prints with underline. The character ‘#’ is replaced by an ASCII decimal string telling the thickness of the underline in printer dot units. Underlines of one dot or two dots are only supported. If it is omitted, underline of one dot is printed. When 2, or a larger number is specified for the font typeface selection, a single dot underline is printed even if two dots are specified.
Italic	ESC |iC	Support is provided only when 2 or more is specified in the font typeface selection.
Alternate color	ESC |#rC	It is not supported.
Red	ESC |rC	It prints with the second color of the receipt. Printing is possible only when "Printing Color" of Printer Setting is "Two Colors". When Color = mono is set in the setting program, printing is not affected by specifying this escape sequence.
Reverse video	ESC |rvC	Prints in a reverse video format. If ‘!’ is specified then reverse video is disabled.
Shadowing	ESC |#sC	It is not supported.
Strike lines	ESC |stC	Strike lines are added and printed.
Single high and wide	ESC |1C	Prints normal size.
Double wide	ESC |2C	Prints double-wide characters.
Double high	ESC |3C	Prints double-high characters.
Double high and wide	ESC |4C	Prints double-high/double-wide characters.
Scale horizontally	ESC |#hC	It prints characters, enlarging them horizontally. The character '#'is the ASCII decimal character string which shows horizontally enlarging rate. It supports form one to eight times. If "#" is omitted, it prints in life-size.

Name	**Data**	**Contents**
Scale vertically	ESC |#vC	It prints characters, enlarging them vertically. The character '#' is the ASCII decimal character string which shows vertically enlarging rate. It supports form one to eight times. At the time of "#" abbreviation, it prints in life-size.
RGB Color	ESC |#fC	Printed in '#' color. The replacement character '#' is replaced by ASCII decimal numbers, which represent RGB. They consist of three digits, representing red, green and blue. For instance, "255255000" is yellow. Each three digit numerical value of RGB can be specified from 0 to 255. If '#' is abbreviated, then printing will be performed in black. There is no impact to bitmap printing. If the specified value exceeds the range permitted for RGB, the escape sequence for the specified color will be ignored. The gradation for gradation printing of registry "RasterMode"="F" and multi-font printing must be four or more gradation levels. When fonts other than the device fonts of the printer are selected, the print data is prepared by performing a 16 level monochrome gradation conversion.
Center	ESC |cA	It aligns the text, from there after, in the center. It is not available unless you assign at the head of the line. It is unavailable in 90 degrees rotating to the left or to the right by RotatePrint Method.
Right justify	ESC |rA	It aligns the text, from there after, to the right. It is not available unless you assign at the head of the line. It is unavailable in 90 degrees rotating to the left or to the right by RotatePrint Method.
Normal	ESC |N	It regains POS Printer attribute to the normal state. It is impossible to regain from Centering or Align Right, unless you assign at the head of the row.
SubScript	ESC |tbC	It is not supported.
SuperScript	ESC |tpC	It is not supported.

4.3. Common Properties

The following sections describe the properties provided commonly to the POS printer.
We have two kinds of properties: Read-Only and Read/Write. If a property is for writing, R/W will be shown at the side of each property.
Also, only errors that are notified by PosControlException have special meaning.

For information about the Compatibility and SynchronizingObject properties, see MSDN.
"https://docs.microsoft.com/en-us/previous-versions/windows/embedded/ms845351(v=winembedded.10)"
"https://docs.microsoft.com/en-us/previous-versions/windows/embedded/ms845348(v=winembedded.10)"

CapCompareFirmwareVersion Property

Syntax

bool CapCompareFirmwareVersion;

Remarks

If **TRUE**, then the Service/device supports comparing the version of the firmware in the physical device against that of a firmware file.

This property is initialized to **TRUE** by the **Open** method.

CapPowerReporting Property

Syntax

PowerReporting CapPowerReporting;

Remarks

It identifies power reporting ability. The value to show power reporting ability is just the following.

Value	Meaning
Standard(1)	SO can judge two kinds of power states and can report it. (Online and OffOffline)

This property is initialized by **Open** Method.

CapStatisticsReporting Property

Syntax

bool CapStatisticsReporting;

Remarks

This property is initialized to **FALSE** by the **Open** method. Statistics reporting is not supported.

CapUpdateStatistics Property

Syntax

bool CapUpdateStatistics;

Remarks

This property is initialized to **FALSE** by the **Open** method. Statistics reporting is not supported.

CheckHealthText Property

Syntax

string CheckHealthText;

Remarks

It keeps the **CheckHealth** Method result called just before. The following are examples of the check.

- In case of Internal	"Internal HCheck: Successful", "Internal HCheck: OFF/OFFLINE"
- In case of External	"External HCheck: Not Supported"
- In case of Interactive	"Interactive HCheck: Not Supported"

This value is initialized to null character before the first **CheckHealth** method call.

Claimed Property

Syntax

bool Claimed;

Remarks

TRUE: The device is claimed for exclusive access.

FALSE: The device is released for sharing with other applications.

Claimed Property value is initialized to **FALSE** by **Open** Method.

DeviceDescription Property

Syntax

string DeviceDescription;

Remarks

"FP 1 Station Thermal POSPrinter (C) 20xx Fujitsu Isotec Limited" is set.

This property is a character string identifying the device, and holds the device and any pertinent information about it

This property is initialized by the **Open** method.

DeviceEnabled Property R/W

Syntax

bool DeviceEnabled;

Remarks

TRUE:

The device is made to enable (Operation state). If converted to **TRUE,** it is made to enable.

FALSE:

The device is made to disable. If converted to **FALSE**, it is made to disable.

Before this device is used, application must set this property **TRUE.** Also, while **DeviceEnable** is **TRUE**, Device Connection State (**PowerReporting**) is reported. This property is initialized to **FALSE** by **Open** Method.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
NotClaimed(103)	In order to make exclusive use device enable, it must acquire exclusive access right, before that.
NoHardware(107)	POS Printer is OFF or OFFLINE or the cable is not connected. Clear the problem, and then execute the property again. (* In the case of USB interface, even if a POS printer is connected, this error occurs when a serial number set by registry is different from a serial number set by a POS printer.) *In case of LAN interface model, same error occurs that if both of POS printer's IP addresses and the registry's are different, even the POS printer is connected in network.
Failulre(111)	The connection to the device is failed. Carry out once again after having confirmed whether there is a connection port and whether a connection port is not used by other programs.
Timeout(112)	Connection to the POS Printer could not be established. There is the possibility of cover open or running out of paper.
Busy(113)	Setting the property is failed because processing . Set the property after processing.
The others	Refer to **ErrorCode** Property.

DeviceName Property

Syntax

string DeviceName;

Remarks

"FP 1 Station Thermal POSPrinter" is set up.

This property shows the device and the any pertinent information about it.

This is a short version of **DeviceDescription** property and should be limited to 30 characters.

This property is initialized by the **Open** method.

FreezeEvents Property R/W

Syntax

bool FreezeEvents;

Remarks

In case of **TRUE**, Event is not reported from Control.

Event is kept by Control till the freeze is terminated.

In case of **FALSE**, if Event is reported from Control. If there is any Event kept during freeze, that Event is reported when **FreezeEvents** are converted to **FALSE**.

If interrupt by Event is undesirable, Application can select Event freeze. In case that **ErrorEvent** is frozen, **State** Property becomes **Busy(3)**. In such case, Control cannot be closed. In this case, cancel the event frozen by **ClearOutput** Method, or set **TRUE** and execute **Close** Method after **ErrorEventis** originated.

This property is initialized to **FALSE** by **Open** Method.

OutputID Property

Syntax

int OutputID;

Remarks

It has the identifier to uniquely identify asynchronous request. (Asynchronous Respond Method Call when **AsyncMode** Property is set to **TRUE**.)
When a method successfully initiates a synchronous or asynchronous output, the Control assigns an identifier to the request. For asynchronous output, when the output completes, the Control will fire an **OutputCompleteEvent** passing this output ID as a parameter.
Output ID Number is numbered cyclically between 1 and 65535.

PowerNotify Property R/W

Syntax

PowerNotification PowerNotify;

Remarks

This is the Power Notify Function type set by Application.
The values to show Power Notify Function are just the following.

Value	Meaning
Disabled(0)	Control does not give any power notification to Application. **StatusUpdateEvent** concerning Power Notification is not given and any setting is not done to **PowerState** Property. (Default Value)
Enabled(1)	When **DeviceEnabled** is set to **TRUE**, Control does **StatusUpdateEvent** notification and **PowerState** Property update.

PowerNotify Property can be set during Device is disable, which means when **DeviceEnabled** Property is **FALSE**.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	One of the following happens: Device is already enabled. Property setting value is illegal.
The others	Refer to **ErrorCode** items.

PowerState Property

Syntax

PowerState PowerState;

Remarks

While **PowerNotify** is Enabled**(1)**, Present Device Power State is set up.

The values to show power state are just the following.

Value	Meaning
Unknown(2000)	It cannot judge the device power state according to one of the following reasons. (Default Value) When **PowerNotify** = **Disabled(0),** Power Notify Function is disable. When **DeviceEnabled** = **FALSE**, Power State Monitor does not operate till Device becomes enable.
Online (2001)	Device power is ON and at the same time is ready.
OffOffline(2004)	Device power is OFF, or is not connected to the body. As for recovery way, refer to **ErrorCode** Property, NoHardware(107).

This property is initialized to Unknown(2000) by the **Open** method.

When **PowerNotify** is set to Enabled(1) and **DeviceEnabled** is **TRUE**, then this property is updated as the SO detects power condition changes.

ServiceObjectDescription Property

Syntax

string ServiceObjectDescription;

Remarks

"Fujitsu Isotec FP POS Printer Service Object" is set.

This property is initialized by the **Open** method

ServiceObjectVersion Property

Syntax

System.Version ServiceObjectVersion;

Remarks

"1013XXX" is set. Holds the Service Object version number. (XXX varies depending on the time the Control Object is distributed.) This property is initialized by the **Open** method.

State Property

Syntax

ControlState State;

Remarks

It shows the present state of Control.

Value	Meaning
Closed(1)	Control is closed. (Default)
Idle(2)	Control is in normal state and not busy.
Busy(3)	Control is in normal state and busy because it is outputting.
Error(4)	Error is reported, and if you want to restart usual I/O, Application must return Control into normal state.

This property is always readable.

4.4. Common Methods

The following section describes the methods provided commonly to the POS printer.

CheckHealth Method

Syntax

String CheckHealth (HealthCheckLevel *Level*);

Level parameter shows the type of health check executed with device. The following values can be assigned.

Value	Meaning
Internal (1)	Performs on line check. The result is set to the **CheckHealthText** property as follows: When the POS Printer is connected to POS and the power is turned on, "Internal HCheck: Successful" is set in the **CheckHealth** property. When the POS Printer is not connected to POS, or the power is turned off, "Internal HCheck: OFF/OFFLINE" is set in the **CheckHealth** property.
External (2)	It is not supported with this class library.. "Internal HCheck: NOT Supported" is set in the **CheckHealth** property. Returns Illegal(106).
Interactive(3)	It is not supported with this class library.. "Internal HCheck: NOT Supported" is set in the **CheckHealth** property. Returns Illegal(106).

Remarks

It is called at the time of device condition test. The result of this method is contained in **CheckHealthText** Property. **CheckHealth** Method is always synchronous.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	It shows that Health Check Procedure starts properly and that, when confirmed that, the device is working properly. However you cannot decide about its properness until you see the result of the test.
Illegal(106)	*Level* parameter not supported is assigned.
NoHardware(107)	After assigning and executing Internal(1), it is Offline.
Busy(113)	It cannot be executed while Output is ON.
The others	Refer to the items of **ErrorCode**.

Claim Method

Syntax

void Claim (int *Timeout*);

Timeout Parameter shows maximum waiting time (in ms.) till it acquires exclusive access right.

In case of zero, even if it cannot acquire the Device Exclusive Access Method, it returns the result immediately.

If **OPOS_FOREVER(-1)** is set, Method waits as long as till it can acquire exclusive access right.

Remarks

This method is called when exclusive access is required to device.

POS Printer Device cannot be used without acquiring exclusive access right.

In case of success, **Claimed** Property is set to **TRUE**.

If **Claim** Method is executed, it establishes connection to POS Printer Device, and confirms the process possibility. If the process is possible, it requires particular data and **Claim** Method, and ends normally.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	Unavailable ***Timeout*** Parameter is assigned.
Timeout(112)	Other application tries to access the device exclusively and waits for being released. But ***Timeout*** Time (in ms.) has passed. Or, POS Printer Device does not become POS-Printer-Device-Process-Possible State even after ***Timeout*** Time (in ms.) passes.

ClearOutput Method

Syntax

void ClearOutput ();

Remarks

It is called when to clear all the output devices buffered by asynchronous issue of each method of **PrintNormal**, **CutPaper**, **RotatePrint**, **PrintBarCode**, **PrintBitmap**, **TransactionPrint, PageModePrint.** Also, it does release when it is in rotation mode, in batch processing mode or page mode by **RotatePrint** Method, **TransactionPrint** Method or **PageModePrint** Method.

Pending Output Error Event (which is in case that **FreezeEvents** are wanting for being set to **FALSE**) is also cleared.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Output is cleared.
Failure(111)	Device is accessed exclusively by other process.
The others	Refer to items of **ErrorCode**.

Close Method

Syntax

void Close ();

Remarks

It is called when to release Device and its resource.

If **DeviceEnabled** Property is **TRUE,** Device is forced to disable.

If **Claimed** Property is **TRUE,** at first, exclusive access will be released.

Don't execute at the time of Event Processing. (within Event Handler)

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Device is disabled and closed.
Busy(113)	Asynchronous processing is under way.
The others	Refer to items of **ErrorCode**.

DirectIO Method

Syntax

DirectIOData DirectIO (int *Command*, int *pData*, object obj);

Remarks

In this class library, synchronous/asynchronous transmission of binary character string, and conversional synchronous/asynchronous transmission of hexadecimal character string are supported.

Synchronous transmission of binary character string

Command	OPOS_FIT_DIO_BIN_SYNC(0)
	OPOS_FIT_DIO_BIN_IMMIDIATE(1)
	OPOS_FIT_DIO_BIN_REALTIME(2)

pData	Not in use
pString	IN Binary character string
Function	It synchronously transmits binary character string to POS Printer. With this command, it becomes possible to directly transmit command to POS Printer Binary Character String is the following format. &H00 - &Hff(&HFF) The following are set when to transmit Partial Cut Command (1BH 6DH) to the printer side. "&H1B&H6D" (Small letters in the alphabet except "&H" are acceptable) It ignores the characters not admittable as HEX. Example: When "ABCD&H00EFG" is transmitted, only "0x00" is transmitted to Printer side.

Conversional synchronous transmission of hexadecimal character string

Command	OPOS_FIT_DIO_HEX_SYNC(5)
pData	Not yet used
pString	IN hexadecimal character string
Function	It synchronously transmits hexadecimal character string to POS Printer. With this command, it becomes possible to directly transmit command to POS Printer One byte character is displayed in 2 byte character. &H00 -> '00' Characters ranging from '0' – '9', 'A' – 'F', and 'a' – 'f' are available. If there is any character except the above, it returns Illegal(106) without transmitting the command.

Asynchronous transmission of binary character string

Command	OPOS_FIT_DIO_BIN_ASYNC(6)
pData	Not yet used
pString	IN Binary character string
Function	It asynchronously transmits binary character string to POS Printer. With this command, it becomes possible to directly transmit command to POS Printer. After execution **OutputCompleteEvent** is up. Binary character string is just like the following format. &H00 - &Hff(&HFF) The following are set when to transmit Partial Cut Command (1BH 6DH) to the printer side. "&H1B&H6D" (Small letters in the alphabet except "&H" are acceptable) It ignores characters not admittable as HEX. Example: When "ABCD&H00EFG" is transmitted, only "0x00" is transmitted to Printer side.

Conversional asynchronous transmission of hexadecimal character string

Command	OPOS_FIT_DIO_HEX_ASYNC(7)
pData	Not yet used
pString	IN hexadecimal character string
Function	It asynchronously transmits hexadecimal character string to POS Printer. With this command, it becomes possible to directly transmit command to POS Printer. After execution **OutputCompleteEvent** is up. One byte character is displayed in 2 byte character. &H00 -> '00' Characters ranging from '0' - '9', 'A' - 'F', and 'a' - 'f' are available. If there is any character except the above, it returns Illegal(106) without transmitting the command.

Setting of error correction level of QR code

Command	OPOS_FIT_DIO_SET_QRERRORLV(20)
pData	IN error correction level of QR code
pString	Not yet used
Function	Specifies error correction level of QR code. The default value is “0”. The default value is set when **DeviceEnabled**=True is executed.

Value	Error correction level
0	Level L(7%)
1	Level M(15%)
2	Level Q(25%)
3	Level H(30%)

Getting of error correction level of QR code

Command	OPOS_FIT_DIO_GET_QRERRORLV(21)
pData	OUT error correction level of QR code
pString	Not yet used
Function	The error correction level of the QR code is returned. Error correction level (0-3) of the QR code is set to obj.

Setting of error correction level of Micro QR code

Command	OPOS_FIT_DIO_SET_MICROQRERRORLV(22)
pData	IN error correction level of Micro QR code
pString	Not yet used
Function	Specifies error correction level of Micro QR code. The default value is “0”. The default value is set when **DeviceEnabled**=True is executed.

Value	Error correction level
0	Level L(7%)
1	Level M(15%)
2	Level Q(25%)

Getting of error correction level of Micro QR code

Command	OPOS_FIT_DIO_GET_MICROQRERRORLV(23)
pData	OUT error correction level of Micro QR code
pString	Not yet used
Function	The error correction level of the Micro QR code is returned. Error correction level (0-2) of the Micro QR code is set to obj.

For all the calling after Enable, if these values except command are set to ***Command***, it returns Illegal(106) And **DirectIO** Method is buffered in **TransactionPrint, RotatePrint and PageModePrint.** In this case, as for synchronous/asynchronous transmission of **DirectIO** Method, its synchronous/asynchronous execution depends on synchronous/asynchronous execution of **TransactionPrint** Method, **RotatePrint** Method and **PageModePrint** Method.

Value	**Meaning**
Illegal(106)	This method is unavailable.
NoHardware(107)	POS Printer is OFF or OFFLINE.
Failure(111)	OPOS Control is in error state. After releasing error state, execute it.
Busy(113)	It cannot be executed because it is outputting.
Extended(114)	**ErrorCodeExtended** = CoverOpen(201): POS Printer cover is open. (Only when synchronous method is executed.) **ErrorCodeExtended** = RecEmpty(203): Run out of receipt paper . (Only when synchronous method is executed.) **ErrorCodeExtended** = Blackmark(10001): Black Mark Error occurs. (Only when synchronous method is executed.) **ErrorCodeExtended** = Fatal (10003): Fatal error occurs. (Only when synchronous method is executed.) **ErrorCodeExtended** = Overheat (10006): Head Overheat Error occurs. (Only when synchronous method is executed.)
The others	Refer to the items of **ErrorCode**.

Open Method

Syntax

void Open ();

Remarks

It is called to open the device.

Opens the device specified in the PosExplorer.

If the Open method succeeds, common properties and other class-specific properties are initialized.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	It succeeds in Open.
Illegal(106)	The applicable Control has already opened.

Release Method

Syntax

void Release ();

Remarks

Call this device when to release exclusive access of Device.

If **DeviceEnabled** Property is **TRUE** and exclusive device, Device is made to disable. Don't execute at the time of Event Processing. (within Event Handler)

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Exclusive Access is released. **Claimed** Property become **FALSE.**
Illegal(106)	Application doesn't have exclusive access right to applicable Device.
Busy(113)	Asynchronous processing is under way.
The others	Refer to the explanation of **ErrorCode** Property.

ResetStatistics Method

Syntax

void ResetStatistics ();

Remarks

This method is not supported.

Exception

Value	Meaning
Illegal(106)	This method is not supported.

RetrieveStatistics Method

Syntax

string RetrieveStatistics ();

Remarks

This method is not supported.

Exception

Value	Meaning
Illegal(106)	This method is not supported.

Syntax

string RetrieveStatistics (StatisticCategories *statistics*);

Remarks

This method is not supported.

Exception

Value	Meaning
Illegal(106)	This method is not supported.

Syntax

string RetrieveStatistics (string[] *statistics*);

Remarks

This method is not supported.

Exception

Value	Meaning
Illegal(106)	This method is not supported.

UpdateFirmware Method

Syntax

void UpdateFirmware (string *FirmwareFileName*);

Parameter	**Description**
FirmwareFileName	Specifies either the name of the file containing the firmware or a file containing a set of firmware files that are to be downloaded into the device.

Remarks

This method updates the firmware of a device with the version of the firmware contained or defined in the file specified by the *FirmwareFileName* parameter regardless of whether that firmware's version is newer than, older than, or the same as the version of the firmware already in the device..

When this method is invoked, the Service Object should check that the specified firmware file exists. If so, this method should return immediately and the remainder of the update firmware process should continue asynchronously. The Service Object should notify the application of the status of the update firmware process by firing **StatusUpdateEvents** with values of OPOS_SUE_UF_PROGRESS(2100) + an integer between 1 and 100 indicating the completion percentage of the update firmware process.

For application convenience, the **StatusUpdateEvent** value OPOS_SUE_UF_COMPLETE(2200) is defined to be the same value as OPOS_SUE_UF_PROGRESS(2100) + 100. If an error is detected during the asynchronous portion of an update firmware process, one of the following **StatusUpdateEvents** will be fired:

After downloading the firmware to the POS printer, when the firmware version acquired from the file name and the version acquired from the POS printer are compared (same processing as the **CompareFirmware** method). If inconsistency is found,

OPOS_SUE_UF_FAILED_DEV_OK(2201) is notified instead of

OPOS_SUE_UF_COMPLETE(2200).

* When using this function, do not specify **AsyncMode, TransactionPrint and PageModePrint**.

Exception

Value	Meaning
OPOS_SUE_UF_FAILED_DEV_OK(2201)	The update firmware process failed but the device is still operational.

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	The method is executed successfully.
NoExist(109)	The file specified by *FirmwareFileName* does not exist
Extended(114)	**ErrorCodeExtended**= POS_EFIRMWARE_BAD_FILE(281): The specified firmware file or files exist, but one or more are either not in the correct format or are corrupt. (When the extension is other than "hx1", this error occurs)

UpdateStatistics Method

Syntax

void UpdateStatistics (Statistic s*tatistics*);

Remarks

This method is not supported.

Exception

Value	Meaning
Illegal(106)	This method is not supported.
Extended(114)	This method is not supported.

Syntax

void UpdateStatistics (StatisticCategories s*tatistics, object value*);

Remarks

This method is not supported.

Exception

Value	Meaning
Illegal(106)	This method is not supported.
Extended(114)	This method is not supported.

4.5. Specific Property

AsyncMode Property R/W

Syntax

bool AsyncMode;

Remarks

TRUE: Printing methods of **PrintNormal, CutPaper, PrintBarCode, PrintBitmap, RotatePrint, TransactionPrint, PageModePrint** are executed asynchronously.

FALSE: Method is executed synchronously.

This property is initialized to **FALSE** by the **Open** method.

CapCharacterSet Property

Syntax

CharacterSetCapability CapCharacterSet;

Remarks

It shows printable character setting of POS printer.

This property has one of the following values:

Value	**Meaning**
Kanji(11)	Character setting supports cord page 932. It supports single-byte katakana's between 0xA1 and 0xDF, and all the ASCII characters between 0x20 and 0x7F. Also, it supports the Shift JIS Code characters which are defined by the first, second and third JIS standard level .

This property is initialized by the **Open** method.

(Note)It does not support the Shift JIS Code characters if Korean / Thai / Simplified Chinese / Traditional Chinese language.

It supports KS code characters if Korean language is selected.

It supports TIS code characters if Thai language is selected.

It supports GB2312 code characters if Simplified Chinese language is selected.

It supports BIG5 code characters if Traditional Chinese language is selected.

CapCoverSensor Property

Syntax

bool CapCoverSensor;

Remarks

TRUE: POSPrinter has "Cover Open Sensor".

This property is initialized by the **Open** method.

CapMapCharacterSet Property

Syntax

bool CapMapCharacterSet;

Remarks

TRUE: the Service Object is able to map the characters to the character sets defined in the **CharacterSetList** property.

CapRec2Color Property

Syntax

bool CapRec2Color;

Remarks

TRUE: It is possible to print receipts in two colors. (*If the printing color is set up "Mono" by Setting up Program, Setting-up will be **FALSE**: Impossible to print in two colors.)

This property is initialized by the **Open** method.

CapRecBarCode Property

Syntax

bool CapRecBarCode;

Remarks

TRUE: It is possible to print bar-codes on receipts.

This property is initialized by the **Open** method.

CapRecBitmap Property

Syntax

bool CapRecBitmap;

Remarks

TRUE: It is possible to print bitmap on receipts.

This property is initialized by the **Open** method.

CapRecBold Property

Syntax

bool CapRecBold;

Remarks

TRUE: It is possible to have Bold attribute with receipts.

This property is initialized by the **Open** method

CapRecCartridgeSensor Property

Syntax

PrinterCartridgeSensor CapRecCartridgeSensor:

Remarks

0: The feature of the receipt cartridge sensor is not supported.

This property is initialized by the **Open** method

CapRecColor Property

Syntax

PritnerColors CapRecColor:

Remarks

0: The feature of the receipt color printing is not supported.

This property is initialized by the **Open** method

CapRecDhigh Property

Syntax

bool CapRecDhigh;

Remarks

TRUE: The receipt can print double high characters.

This property is initialized by the **Open** method.

CapRecDwide Property

Syntax

bool CapRecDwide;

Remarks

TRUE: The receipt can print double wide characters.

This property is initialized by the **Open** method.

CapRecDwideDhigh Property

Syntax

bool CapRecDwideDhigh;

Remarks

TRUE: The receipt can print double high/double wide characters.

This property is initialized by the **Open** method.

CapRecEmptySensor Property

Syntax

bool CapRecEmptySensor;

Remarks

TRUE: The receipt has an out-of-paper sensor.

This property is initialized by the **Open** method.

CapRecItalic Property

Syntax

bool CapRecItalic:

Remarks

FALSE: The receipt cannot print italic characters.
This property is initialized by the **Open** method

CapRecLeft90 Property

Syntax

bool CapRecLeft90;

Remarks

TRUE: The receipt can print in a rotated 90° left mode.
This property is initialized by the **Open** method.

CapRecMarkFeed Property

Syntax

PrinterMarkFeeds CapRecMarkFeed:

Remarks

0: The feature of handling mark sensed paper is not supported.
This property is initialized by the **Open** method

CapRecNearEndSensor Property

Syntax

bool CapRecNearEndSensor;

Remarks

TRUE: The receipt has a low paper sensor.
FALSE: The low paper sensor does not work.
This property is initialized by the **Open** method If PNESense in the setting program is set to Enabled, it is initialized to **TRUE**, and if Disabled, it is initialized to **FALSE**.

CapRecPageMode Property

Syntax

bool CapRecPageMode:

Remarks

TRUE: The printer is capable of supporting Page Mode for the receipt station.
This property is initialized by the **Open** method

CapRecPapercut Property

Syntax

bool CapRecPapercut;

Remarks

TRUE: The receipt can perform paper cuts.
This property is initialized by the **Open** method.

CapRecPresent Property

Syntax

bool CapRecPresent;

Remarks

TRUE: It is possible to print receipts.
This property is initialized by the **Open** method.

CapRecRight90 Property

Syntax

bool CapRecRight90;

Remarks

TRUE: It is possible to have 90-Degree-Rotaion-to-the-Right attribute of receipts.
This property is initialized by the **Open** method.

CapRecRotate180 Property

Syntax

bool CapRecRotate180;

Remarks

TRUE: It is possible to have 180-Degree-Rotaion attribute of receipts.

This property is initialized by the **Open** method.

CapRecStamp Property

Syntax

bool CapRecStamp;

Remarks

FALSE: It is impossible to print stamp on receipts.

This property is initialized by the **Open** method.

CapRecUnderline Property

Syntax

bool CapRecUnderline;

Remarks

TRUE: It is possible to have Underline attribute of receipts.

This property is initialized by the **Open** method.

CapTransaction Property

Syntax

bool CapTransaction;

Remarks

TRUE: Batch processing of POS Printer is valid.

This property is initialized by the **Open** method.

CartridgeNotify Property R/W

Syntax

PrnterCartridgeNotify CartridgeNotify;

Remarks

Contains whether cartridge state notification is available. This property is specified by the application.

Value	Meaning
Disabled(0)	The Control will not provide any cartridge state notifications to the application. No cartridge state notification **StatusUpdateEvents** will be fired, and **JrnCartridgeState**, **RecCartridgeState**, and **SlpCartridgeState** may not be set.

This property is initialized to Disabled(0) by the **Open** method.

Exception

When you call this property, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	This property cannot be set.
Other Values	Refer to **ErrorCode.**

CharacterSet Property R/W

Syntax

int CharacterSet;

Remarks

It sets up the characters for printing.
This property is initialized when Device is first enabled after **Open** Method.
One of the following values is set up in this property.
< English(Latin) language >

Value	Meaning
437	It selects PC437 (USA: Standard Europe) Character Set.
850	It selects PC850 (Multilingual) Character Set.
852	It selects PC852 (Latin2) Character Set.
857	It selects PC857 (Turkish) Character Set.
858	It selects PC858 (Euro) Character Set.
860	It selects PC860 (Portuguese) Character Set.
863	It selects PC863 (Canadian-French) Character Set.
864	It selects PC864 (Arabic without BOX DRAWINGS below 20) Character Set.
865	It selects PC865 (Nordic) Character Set.
866	It selects PC866 (Cyrillic #2) Character Set.
869	It selects PC869 (Greece) Character Set.
932	Windows Code Page; Japanese Version Shift-JIS.
PTR_CS_ASCII (998)	It set up ASCII Character. It supports ASCII Characters between 0x20 and 0x7F. The constant value is 998.
1252	It selects WPC1252 Character Set.
2859	It selects ISO8859-2 (1999 Latin Alphabet No.2) Character Set.
28597	It selects ISO8859-7 (1987 LatinGreek Alphabet) Character Set.

< Korean / Thai / Simplified Chinese / Traditional Chinese language>

Value	Meaning
437	It selects PC437 (USA: Standard Europe) Character Set.
850	It selects PC850 (Multilingual) Character Set.
858	It selects PC858 (Euro) Character Set.
860	It selects PC860 (Portuguese) Character Set.
863	It selects PC863 (Canadian-French) Character Set.
865	It selects PC865 (Nordic) Character Set.
874	Thailand Windows Code Page; TISCODE. (Only Thai language)
932	Windows Code Page; Japanese Version Shift-JIS.
936	China Windows Code Page; GB2312 (Only Simplified Chinese language)
949	Korea Windows Code Page; KSCODE. (Only Korean language)
950	Taiwan Windows Code Page; Big5 and HKSCS. (Only Traditional Chinese language)
PTR_CS_ASCII (998)	It set up ASCII Character. It supports ASCII Characters between 0x20 and 0x7F. The constant value is 998.

Exception

When you call this property, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	It succeeds in setting up this property.
Illegal(106)	Illegal values were used.
The others	Refer to the items of **ErrorCode.**

CharacterSetList Property

Syntax

int[] CharacterSetList;

Remarks

It is Character String of Character Setting up Number.

In English(Latin) language, "437,850,852,857,858,860,863,864,865,866,869,932,998,1252,28592,28597" are set up.

In Korean language, "437,850,858,860,863,865,932,949,998" are set up.

In Thai language, "437,850,858,860,863,865,874,932,998" are set up.

In Simplified Chinese language, "437,850,858,860,863,865,932,936,998" are set up.

In Traditional Chinese language, "437,850,858,860,863,865,932,950,998" are set up.

One of the values is set by Installer. This property is initialized by the **Open** method.

CoverOpen Property

Syntax

bool CoverOpen;

Remarks

TRUE: POS Printer Cover is open.

FALSE: POS Printer Cover is closed.

This property is initialized while Device is made to enable, and keeps the present state.

ErrorLevel Property

Syntax

PrinterErrorLevel ErrorLevel;

Remarks

It shows seriousness of error condition.

One of the following values is set up in this property.

Value	Meaning
None(1)	It is not error state.
Recoverable(2)	Recoverable error occurs. (at the time of Cover Open Error, Receipt End Error, Head Hot Error, Black Mark Error or Power Discontinuity Error)
Fatal(3)	Unrecoverable Error happens. (only in case of fatal error)

This property is set up by Control before **ErrorEvent** is notified. If the error is deleted, this property converts into None(1).

ErrorStation Property

Syntax

PrinterStation ErrorStation;

Remarks

Holds the POS printer (Receipt(2)) in printing when an error is detected.
This property is set up before **ErrorEvent** is notified.
When the power is turned off (or the cable is disconnected), "0" is set.

ErrorString Property

Syntax

string ErrorString;

Remarks

It keeps particular-to-vender description of present error.
This property is set up by Control before **ErrorEvent** is notified. If this description is not used, null character string is set up in property. When the error is deleted, this property is converted to null character string.
The following wordings are set up by POS Printer.

- When the cover opens.	"Cover Open"
- When receipt paper ended.	"Paper End"
- When the head is hot.	"Head Hot"
- When fatal error occurs	"Fetal Error"
- When Cutter jam Error occurs	"Cutter Jam Error"
- Power Discontinuity (Offline)	"Power Off or Offline"

FlagWhenIdle Property R/W

Syntax

bool FlagWhenIdle;

Remarks

TRUE: If POS Printer Control is in idle state, it notifies **StatusUpdateEvent**.
FALSE: This event is not notified.
If this status event is notified, **FlagWhenIdle** is automatically reset to **FALSE.**
By utilizing Status Event with this property, Application can know the end of all the asynchronous output. When Output ends normally, or when output is deleted by Event Handler which receives **ErrorEvent,** the event is notified.
If, **State** Property is already Idle(2) and when **FlagWhenIdle** Property is set up to **TRUE, StatusUpdateEvent** is immediately notified. Accordingly, Application can use this event without worrying about the time difference between asynchronous output end and setting up of this flag.
This property is initialized to **FALSE** by **Open** Method.
When you call this property, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	The property was set successfully.

FontTypefaceList Property

Syntax

string[] FontTypefaceList;

Remarks

An empty string is set. It indicates that only the default typeface is supported.
This property is initialized by the **Open** method.

MapCharacterSet Property

Syntax

bool MapCharacterSet;

Remarks

TRUE: On outputting data, the Service Object maps the characters transferred by the application to the character set selected in the **CharacterSet** property for printing data.
This property is initialized to **TRUE** by the **Open** method.

MapMode Property R/W

Syntax

MapMode MapMode;

Remarks

It shows mapping mode of the Printer. It defines measuring units used by other properties, such as line heights and line spacing.
It supports the following map modes. The values inside the parentheses are the values calculated in dots per each unit.

Value	Meaning
Dots(1)	Width of POS Printer dot 0.125mm (1 dot)
Twips(2)	1/1440 of one inch (7.0866 dot)
English(3)	0.001 inch (4.921 dot)
Metric(4)	0.01 mm (12.5 dot)

RecLineSpacing, RecLineWidth, and **RecLineHeight** change, too, if you set up **MapMode**.
At the first enablement after **Open** Method, It is initialized to Dots(1).

Exception

When you call this property, you may throw the following **PosControlException**:.

Value	Property
No Exception	The property was set successfully.
Illegal(106)	Illegal mapping mode is assigned.

PageModeArea Property

Syntax

System.Drawing.Point PageModeArea;

Remarks

Holds the page area for the selected PageModeStation expressed in the unit of measure given by MapMode. This page area can be different than the print area and is determined by the hardware capability of the printer. The string consists of two ASCII numbers separated by a comma, in the following order: horizontal size, vertical size.

For example, if the string is “450,800”, then the page size is 450 horizontal units by 800 vertical units, and the station print area is a rectangle beginning at the top left point (0,0), and continuing up to but not including the bottom right point (449,799).

PageModeDescriptor Property

Syntax

PageModeDesctiptors PageModeDescriptor;

Remarks

Indicates the page mode functions available for the station specified in the PageModeStation property by the logical sum of the following values.

In this class library, the logical sum of the following values is the initial value.

Value	Property
Bitmap(1)	Printing of bitmaps on the PageModeStation is supported.
Barcode(2)	Printing of barcodes on the PageModeStation is supported.
BitmapRotate(4)	Rotation of bitmaps on the PageModeStation is supported.
BarcodeRotate(8)	Rotation of barcodes on the PageModeStation is supported.

PageModeHorizontalPosition Property

Syntax

int PageModeHorizontalPosition;

Remarks

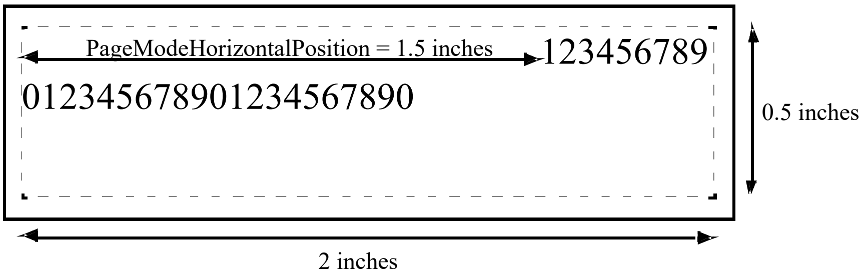
This value is used to specify the absolute horizontal position of the print start position in the page mode print area in the station specified in **PageModeStation**, expressed in the unit of measure given by **MapMode**.

The horizontal direction is the same as the actual **PageModePrintDirection** property. If the exact position cannot be supported then the position is set to the closest supported value.
A read/get on this property will return the specified absolute position value.

The following code sample shows usage of **PageModeHorizontalPosition**.

```
myptr.setMapMode(English);
myptr.setPageModeStation(Receipt);
myptr.pageModePrint(PageMode);
// Set print area to 2 inches by 0.5 inches
myptr.setPageModePrintArea(“0,0,2000,500”);
myptr.setPageModePrintDirection(LeftToRight);
myptr.setPageModeHorizontalPosition(1500);
myptr.printNormal(Receipt, “123456789012345678901234567890¥n”);
myptr.setPageModePrint(Normal);
```

The above code sample will generate the following receipt.



PageModePrintArea Property

Syntax

System.Drawing.Rectangle PageModePrintArea;

Remarks

Holds the print area for the selected **PageModeStation** expressed in the unit of measure given by **MapMode**. The maximum print area is the page area.

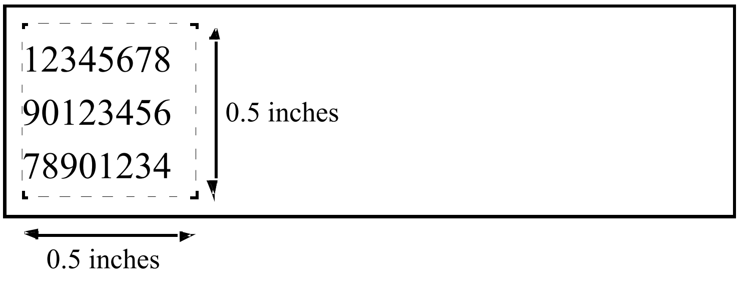
The string consists of four ASCII numbers separated by commas, in the following order: horizontal start, vertical start, horizontal size, vertical size. For example, if the string is “50,100,200,400”, then the station print area is a rectangle beginning at the point (50,100), and continuing up to but not including the point (249,499). This property is initialized to “0,0,0,0”.

Text written to the right edge of the print area will wrap to the next line. Any text or image written beyond the bottom of the print area will be truncated.

For example:

```
myptr.setMapMode(English);
myptr.setPageModeStation(Receipt);
myptr.pageModePrint(PageMode);
// Set print area to half inch square block
myptr.setPageModePrintArea(“0,0,500,500”);
myptr.setPageModePrintDirection(LeftToRight);
myptr.printNormal(Receipt,“123456789012345678901234567890¥n”);
myptr.setPageModePrint(Normal);
```

The above code sample will generate the following receipt.



PageModePrintDirection Property

Syntax

PageModePrintDirection PageModePrintDirection;

Remarks

Holds the print direction. The print direction shall be as follows:

Value	Property
LeftToRight(1)	Print left to right, starting at top left position of the print area. (normal printing)
BottomToTop(2)	Print bottom to top, starting at bottom left position of the print area. (rotated left 90 printing)
RightToLeft(3)	Print right to left, starting at bottom right position of the print area. (upside down printing)
TopToBottom(4)	Print top to bottom, starting at top right position of the print area. (rotated right 90 printing)

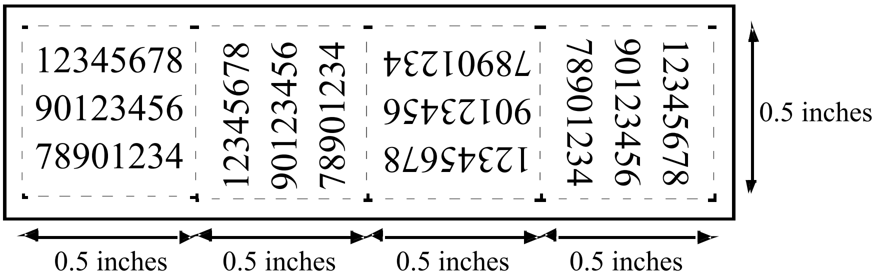
This property is initialized to LeftToRight when the device is first enabled following the **open** method.

Setting this property may also change **PageModeHorizontalPosition** and **PageModeVerticalPosition**. Setting this property will have an effect on the current print area.
By changing the print area, it is possible to generate a receipt or slip with text printed in multiple rotations.

For example:

```
myptr.setMapMode(English);
myptr.setPageModeStation(Receipt);
myptr.pageModePrint(PageMode);
// Set print area to half inch square block
myptr.setPageModePrintArea(“0,0,500,500”);
myptr.setPageModePrintDirection(LeftToRight);
myptr.printNormal(Receipt,“123456789012345678901234567890¥n”);
myptr.setPageModePrintArea(“500,0,500,500”);
myptr.setPageModePrintDirection(BottomToTop);
myptr.printNormal(Receipt,“123456789012345678901234567890¥n”);
myptr.setPageModePrintArea(“1000,0,500,500”);
myptr.setPageModePrintDirection(RightToLeft);
myptr.printNormal(Receipt,“123456789012345678901234567890¥n”);
myptr.setPageModePrintArea(“1500,0,500,500”);
myptr.setPageModePrintDirection(TopToBottom);
myptr.printNormal(Receipt,“123456789012345678901234567890¥n”);
myptr.setPageModePrint(Normal);
```

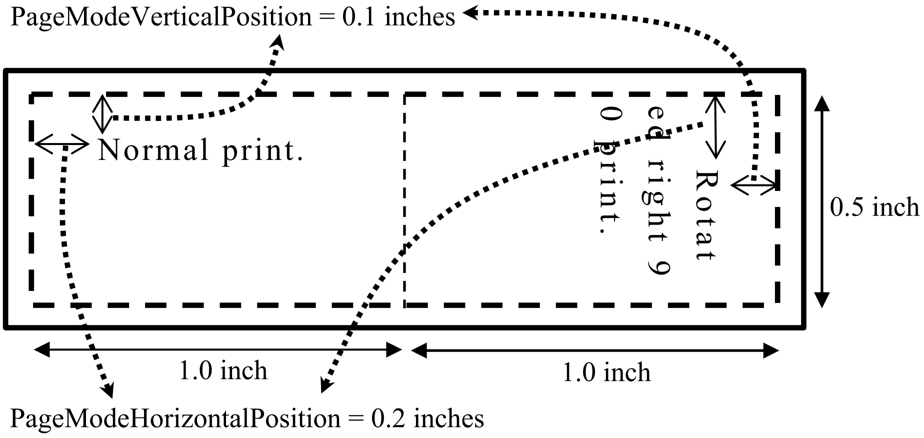
The above code sample will generate the following receipt.



It is also possible to generate rotated text.

```
myptr.setMapMode(English);
myptr.setPageModeStation(Receipt);
myptr.pageModePrint(PageMode);
myptr.pageModeVerticalPosition(100);
myptr.pageModeHorizontalPosition(200);
myptr.setPageModePrintArea(“0,0,1000,500”);
myptr.setPageModePrintDirection(LeftToRight);
myptr.printNormal(Receipt, “Normal print.¥n”);
myptr.setPageModePrintArea(“1000,0,1000,500”);
myptr.setPageModePrintDirection(TopToBottom);
myptr.printNormal(Receipt, “Rotated right 90 print.¥n”);
myptr.setPageModePrint(Normal);
```

The above code sample will generate the following receipt.


PageModeVerticalPosition = 0.1 inches
Normal print.
Rotated right 90 print.
0.5 inch
1.0 inch
1.0 inch
PageModeHorizontalPosition = 0.2 inches

PageModeStation Property

Syntax

PrinterStation PageModeStation;

Remarks

Set the print station for subsequent Page Mode properties.

In this class library, only Receipt(2) can be specified.

PageModeVerticalPosition Property

Syntax

int PageModeVerticalPosition;

Remarks

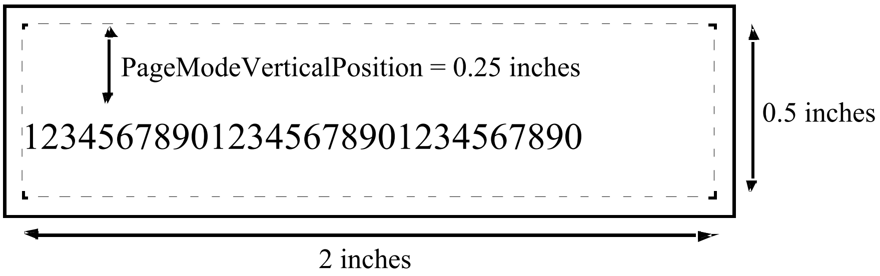
This value is used to specify the absolute vertical position of the print start position in the page mode print area in the station specified in **PageModeStation**., expressed in the unit of measure given by **MapMode**.

The vertical direction is perpendicular to the direction specified in the actual **PageModePrintDirection** property. If the exact position cannot be supported then the position is set to the closest supported value. A read/get on this property will return the specified absolute position value.

The following code sample shows usage of PageModeVerticalPosition.

```
myptr.setMapMode(English);
myptr.setPageModeStation(Receipt);
myptr.pageModePrint(PageMode);
// Set print area to 2 inches by 0.5 inches
myptr.setPageModePrintArea(“0,0,2000,500”);
myptr.setPageModePrintDirection(LeftToRight);
myptr.setPageModeVerticalPosition(250);
myptr.printNormal(Receipt,“123456789012345678901234567890¥n”);
 myptr.setPageModePrint(Normal);
```

The above code sample will generate the following receipt.



RecBarCodeRotationList Property

Syntax

Rotation[] RecBarCodeRotationList;

Remarks

This character string shows the possible direction of receipt bar-code rotation.
"0, R90, L90, 180" are set up.
This property is initialized by **Open** Method. The character strings consist of groups of character strings separated by commas, and indicating rotation direction. The following show character strings which indicate rotation direction.

Value	Meaning
0	Barcode may be printed in the normal orientation.
R90	Barcode may be rotated 90° to the right.
L90	Bar code may be rotated 90° to the left.
180	Barcode may be rotated 180° - upside down.

RecBitmapRotationList Property

Syntax

Rotation[] RecBitmapRotationList;

Remarks

This character string shows the directions in which a receipt bitmap may be rotated..
"0, R90, L90, 180" can be set.
This property is initialized by the **Open** method. The string consists of rotation strings separated by commas. The legal rotation strings are:

Value	Meaning
0	Bitmap may be printed in the normal orientation.
R90	Bitmap may be rotated 90° to the right.
L90	Bitmap may be rotated 90° to the left.
180	Bitmap may be rotated 180° - upside down.

RecCartridgeState Property

Syntax

PrinterCartridgeStates RecCartridgeState;

Remarks

Indicates the status of the currently selected Receipt cartridge (ink, ribbon or toner).

Since the POS printer is the thermal printer, this is fixed to the following value.

Value	Meaning
Unknown(268435456)	The device does not support the feature of notifying the cartridge state.

This property is initialized and kept current while the device is enabled.

RecCurrentCartridge Property R/W

Syntax

PrinterColors RecCurrentCartridgeState;

Remarks

Selection of the receipt cartridge is not supported. It is initialized to 0.

Exceptions

When you call this property, you may throw the following

PosControlException:.

Value	Meaning
Illegal(106)	Specifying cartridge state is invalid.
Other Values	Refer to the **ErrorCode** section.

RecEmpty Property

Syntax

bool RecEmpty;

Remarks

TRUE: Receipt paper is run out

FALSE: Enough receipt paper

This property is initialized while Device is enabled, and keeps present condition

RecLetterQuality Property R/W

Syntax

bool RecLetterQuality;

Remarks

TRUE: It prints in high quality printing mode.

FALSE: It prints in normal printing mode.

This property is initialized to **TRUE,** at the time of initial enablement of Device after **Open** Method.

The subject of high quality printing mode influence is built-in characters and down-load characters. And at the same time, in case of double-width and double-height characters, it can print with smoothing processing, but it prints a little bit slower.

In case of normal printing mode, bitmap is printed in 1/3 resolution. (The inputted size is same as that of high quality printing mode but its resolution is 1/3. Also, in case of printing of double-width- and-double-height built-in characters and larger than that, smoothing processing is not done.

When the bitmap is registered by **SetBitmap**, it is not affected by the **RecLetterQuality** at that time. (If the bitmap is registered by **SetBitmap**, printing the bitmap centered or aligned right in the normal print mode results printing position incorrect. In such case, it is recommended to print in high-quality print mode.)

In case of bitmap printing in Escape Sequence, in high quality printing mode, printing is in normal resolution, and in normal printing mode, printing is in.1/2 resolution. (double-width and double-height) The method follows **RecLetterQuality** in the same way.

(*When Registry is set up, if Smoothing setting is off, it does not do smoothing processing even at the time of setting up to **TRUE**.)

RecLineChars Property R/W

Syntax

int RecLineChars;

Remarks

It is the number of the half-size characters, per line of receipts.

Printing is done in the following font, according to the assigned number of half-size characters per line of receipt.

When 180 dpi mode is disabled, it is as follows.

Characters per Line	Printing Font (WidthxHeight)
Printer Setting: Paper width 80 mm (576 dots)	
48 (double-width 24)	12x24 dots (font A)
57 (double-width 28)	10x24 dots (font B)
72 (double-width 36)	8x16 dots (font C) * Only English(Latin) language.
Printer Setting: Paper width 80 mm (512 dots)	
42 (double-width 21)	12x24 dots (font A)
51 (double-width 25)	10x24 dots (font B)
64 (double-width 32)	8x16 dots (font C) * Only English(Latin) language.
Printer Setting: Paper width 58 mm (420 dots)	
35 (double-width 17)	12x24 dots (font A)
42 (double-width 21)	10x24 dots (font B)
52 (double-width 26)	8x16 dots (font C) * Only English(Latin) language.
Printer Setting: Paper width 58 mm (384 dots)	
32 (double-width 16)	12x24 dots (font A)
38 (double-width 19)	10x24 dots (font B)
48 (double-width 24)	8x16 dots (font C) * Only English(Latin) language.
Printer Setting: Paper width 50 mm (360 dots)	
30 (double-width 15)	12x24 dots (font A)
36 (double-width 18)	10x24 dots (font B)
45 (double-width 22)	8x16 dots (font C) * Only English(Latin) language.

When 180 dpi mode is enabled, it is as follows.

Characters per Line	Printing Font (WidthxHeight)
Printer Setting: Paper width 80 mm (512 dots)	
42 (double-width 21)	12x24 dots (font A)
51 (double-width 25)	10x24 dots (font B)
64 (double-width 32)	8x16 dots (font C) * Only English(Latin) language.
Printer Setting: Paper width 58 mm (360 dots)	
30 (double-width 15)	12x24 dots (font A)
36 (double-width 18)	10x24 dots (font B)
45 (double-width 22)	8x16 dots (font C) * Only English(Latin) language.

If this value changed into supported line character width, the character width is set up to the assigned value. If it cannot support exact width, it is set up to the nearest value in supported line width and at the same time larger value than supported line width. (For example, when to set up paper width to 80 mm and to set 40 for Printer, Service Object will select character size of "48".) If it cannot support character width, Error will return.

Setting **RecLineChars** may also update **RecLineHeight**, **RecLineSpacing**, **RecSideWayMaxChars** and **RecSidewaysMaxlines** properties.

Exception

When you call this property, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	The property was set successfully.
Illegal(106)	Illegal line character width is assigned.

RecLineCharsList Property

Syntax

int[] RecLineCharsList;

Remarks

It is character string which includes line character width supported by receipt.

This property is initialized to the following values by **Open** Method according to Printer paper width and setting.

When 180 dpi mode is disabled, it is as follows.

Printer Paper Width	**Value**
Paper Width 80mm (576 dots)	"48,57,72(*)"
Paper Width 80mm (512 dots)	"42,51,64(*)"
Paper Width 58mm (420 dots)	"35,42,52(*)"
Paper Width 58mm (384 dots)	"32,38,48(*)"
Paper Width 50mm (360 dots)	"30,36,45(*)"

* Only English(Latin) language.

When 180 dpi mode is enabled, it is as follows.

Printer Paper Width	**Value**
Paper Width 80mm (512 dots)	"42,51,64(*)"
Paper Width 58mm (360 dots)	"30,36,45(*)"

* Only English(Latin) language.

RecLineHeight Property R/W

Syntax

int RecLineHeight;

Remarks

It is the receipt print line height. It is written in the unit defined by **MapMode**.

If **RecLineChars** is converted, **RecLineHeight** will be updated to default line height of the selected width.

The value of **RecLineHeight** will be initialized to the default line height of POS Printer by **Open** Method.

The following are the applicable values, when 180 dpi mode is disabled. (*As for the value of Property, the value of **MapMode** Property is Dots(1))

Characters per Line	**Value of RecLineHight property**	
Printer Setting: Paper width 80 mm (576 dots)		
48	24	
57	24	
72	16	* Only English(Latin) language.
Printer Setting: Paper width 80 mm (512 dots)		
42	24	
51	24	
64	16	* Only English(Latin) language.
Printer Setting: Paper width 58 mm (420 dots)		
35	24	
42	24	
52	16	* Only English(Latin) language.
Printer Setting: Paper width 58 mm (384 dots)		
32	24	
38	24	
48	16	* Only English(Latin) language.
Printer Setting: Paper width 50 mm (360 dots)		
30	24	
36	24	
45	16	* Only English(Latin) language.

When 180 dpi mode is enabled, it is as follows.

Characters per Line	Value of RecLineHight property
Printer Setting: Paper width 80 mm (512 dots)	
42	24
51	24
64	16 * Only English(Latin) language.
Printer Setting: Paper width 58 mm (360 dots)	
30	24
36	24
45	16 * Only English(Latin) language.

Exception

When you call this property, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	This property setting is not possible. It can be only acquired.

RecLineSpacing Property R/W

Syntax

int RecLineSpacing;

Remarks

It is the spacing of each single-high print line, including both the printed line height plus the white space between each pair of lines. This property is written by the unit defined by **MapMode**.
When **RecLineChars** is converted, if the new **RecLineHeight** is larger than the value assigned by **RecLineSpacing,** the same value as that of **RecLineHeight** will be set up.
The value of **RecLineSpacing** will be initialized to the default line space of POS Printer after **Open** Method.
The setting is possible between **16(dot)** and **127(dot)**.

Exception

When this property is set, the following value is placed in the **ErrorCode** property:

Value	Meaning
No Exception	The property was set successfully.
Illegal(106)	Property setting range is illegal.
The others	Refer to the **ErrorCode** section.

RecLinesToPaperCut Property

Syntax

int RecLinesToPaperCut;

Remarks

It holds the number of lines that must be advanced before the receipt paper is cut. These lines are ones before reaching the paper cut mechanism.

This property is updated by the change of **RecLineChars** Property or **RecLineSpacing** Property.

RecLineWidth Property

Syntax

int RecLineWidth;

Remarks

It is the width of a line of **RecLineChars** and is written in the unit defined by **MapMode**.

This property is initialized after **Open** Method.

The following values are set up according to printer paper width.

When 180 dpi mode is disabled, it is as follows.

Printer Paper Width	Value
Paper width 80 mm	576
Paper width 80 mm	512
Paper width 58 mm	420
Paper width 58 mm	384
Paper width 50 mm	360

When 180 dpi mode is enabled, it is as follows.

Printer Paper Width	Value
Paper width 80 mm	512
Paper width 58 mm	360

RecNearEnd Property

Syntax

bool RecNearEnd;

Remarks

TRUE: Receipt paper is low.

FALSE: Receipt paper is not low.

This property is initialized when device enabled, and the current value is kept while it enabled.

RecSidewaysMaxChars Property

Syntax

int RecSidewaysMaxChars;

Remarks

In case of Sideways Mode (Printing with 90 degree rotation to the left or to the right), it is the maximum number of the half-size characters per line. Since the width of 90 degrees rotating to the left or to the right is declined to half when the CapRec2Color property is TRUE (2 colors), the number of printable characters is half of the value.

The following values are taken.

Characters per Line	**Characters per Line in Sideways Mode**	
2 Colors		
Printer Setting: Paper width 80 mm (576 dots)		
48 (double-width 24)	138	69
57 (double-width 28)	166	83
72 (double-width 36) *	207	103
Printer Setting: Paper width 80 mm (512 dots)		
42 (double-width 21)	138	69
51 (double-width 25)	166	83
64 (double-width 32) *	207	103
Printer Setting: Paper width 58 mm (420 dots)		
35 (double-width 17)	138	69
42 (double-width 21)	166	83
52 (double-width 26) *	207	103
Printer Setting: Paper width 58 mm (384 dots)		
32 (double-width 16)	138	69
38 (double-width 19)	166	83
48 (double-width 24) *	207	103
Printer Setting: Paper width 50 mm (360 dots)		
30 (double-width 15)	138	69
36 (double-width 18)	166	83
45 (double-width 22) *	207	103

* Only English(Latin) language.

RecSidewaysMaxlLines Property

Syntax

int RecSidewaysMaxLines;

Remarks

In case of Sideways Mode (Printing with 90 degree rotation to the left or to the right), it is the maximum number of the lines printed.
It is the value obtained by dividing **RecLineWidth** Property by **RecLineSpacing** Property. However, if the remainder of the divided value is equal to, or larger than **RecLineHeight** Property (font height), the value will be the one equal to the divided value plus +1. Accordingly, the property will change by changing **RecLineSpacing** Property.
However, exceptionally in case of fontC (*Refer to **RecLineChars** Property), calculation is done to the value **(RecLineWidth** – 7(dot)), against the above condition.
This property is initialized when Device is enabled for the first time after **Open** Method.

RotateSpecial Property R/W

Syntax

Rotation RotateSpecial;

Remarks

It shows the bar code rotation direction.
This property is initialized to Normal(1) with **Open** Method.
This property has one of the following values:

Value	Meaning
Normal(1)	Bar code can be printed to the normal direction.
Right90(257)	Bar code can be printed with the 90 degree rotation to the right.
Left90(258)	Bar code can be printed with the 90 degree rotation to the left.
Rotate180(259)	Bar code can be printed with the 180 degree rotation (upside-down).

Exception

When you call this property, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	The property was set successfully.
Illegal(106)	Illegal property value is assigned.

*The following specific POS printer properties are not supported.

bool CapConcurrentJrnRec;

bool CapConcurrentJrnSlp;

bool CapConcurrentRecSlp;

bool CapConcurrentPageMode;

bool CapJrn2Color;

bool CapJrnBold;

void CapJrnCartridgeSensor;

void CapJrnColor;

bool CapJrnDhigh;

bool CapJrnDwide;

bool CapJrnDwideDhigh;

bool CapJrnEmptySensor;

bool CapJrnItalic;

bool CapJrnNearEndSensor;

bool CapJrnPresent;

bool CapJrnUnderline;

bool CapSlp2Color;

bool CapSlpBarCode;

bool CapSlpBitmap;

bool CapSlpBold;

boolCapSlpBothSidesPrint;

void CapSlpCartridgeSensor;

void CapSlpColor;

bool CapSlpPageMode;

bool CapSlpDhigh;

bool CapSlpDwide;

bool CapSlpDwideDhigh;

bool CapSlpEmptySensor;

bool CapSlpFullslip;

bool CapSlpItalic;

bool CapSlpLeft90;

bool CapSlpNearEndSensor;

bool CapSlpPresent;

bool CapSlpRight90;

bool CapSlpRotate180;

bool CapSlpUnderline;

void JrnCartridgeState;

void JrnCurrentCartridge;

bool JrnEmpty;

bool JrnLetterQuality;

void JrnLineChars;

string JrnLineCharsList;

void JrnLineHeight;

void JrnLineSpacing;

void JrnLineWidth;

bool JrnNearEnd;

string SlpBarCodeRotationList;

string SlpBitmapRotationList;

void SlpCartridgeState;

void SlpCurrentCartridge;

bool SlpEmpty;

bool SlpLetterQuality;

void SlpLineChars;

string SlpLineCharsList;

void SlpLineHeight;

void SlpLinesNearEndToEnd;

void SlpLineSpacing;

void SlpLineWidth;

void SlpMaxLines;

bool SlpNearEnd;

void SlpSidewaysMaxChars;

void SlpSidewaysMaxLines;

void SlpPrintSide;

4.6. Exclusive-Use Methods

BeginInsertion Method

Syntax

void BeginInsertion (int *Timeout*);

Remarks

Because this method is only applicable for the Slip Printers, this is not supported by this class library.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	The POS printer does not have the slip.
Other Values	Refer to the **ErrorCode** section.

BeginRemoval Method

Syntax

void BeginRemoval (int *Timeout*);

Remarks

Because this method is only applicable for the Slip Printers, this is not supported by this class library.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	The POS printer does not have the slip.
Other Values	Refer to the **ErrorCode** section.

ChangePrintSide Method

Syntax

void ChangePrintSide(PrinterSide *Side*);

Remarks

Because this method is only applicable for the Slip Printers, this is not supported by this class library.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	The POS printer does not have the slip.
Other Values	Refer to the **ErrorCode** section.

ClearPrintArea Method

Syntax

void ClearPrintArea ();

Remarks

Clears the print data in the page mode print area defined by the PageModePrintArea property.

The entire page may be cleared by setting the PageModePrintArea to be the same as the PageModeArea and then using clearPrintArea or by exiting Page Mode with pageModePrint with Cancel.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	The method was set successfully.
Other Values	Refer to the **ErrorCode** section.

CutPaper Method

Syntax

void CutPaper (int *Percentage*);

Percentage Parameter indicates the percentage of the paper to be cut. When the value is between '1' to '99', partial cutting is performed. When the value is '100', full cutting is performed.
When the value is other than '1' to '100', Illegal(106) is returned.

Remarks

This method is called when to cut receipt paper.
This method is executed synchronously if **AsyncMode** is **FALSE** and asynchronously if **AsyncMode** is **TRUE.** When **PrintNormal** Method or **PrintImmediate** Method is called, paper cutting can be done using Escape Sequence of paper cutting, too. In addition to that, if POS Printer has buffered data (even though printing is requested, POS Printer does not print), it cannot cut paper. In order to cut receipt paper, it must be at the head of each line.

Exception

When you call this property, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Method ends properly.
Illegal(106)	Illegal percentage is assigned.
NoHardware(107)	POS Printer is OFF or OFFLINE.
Failure(111)	This class library is in error state. Execute after deleting error state.
Busy(113)	It cannot execute because it is outputting.
Extended(114)	**ErrorCodeExtended** = CoverOpen(201): POS Printer cover is open . (It returns only when **AsyncMode** is **FALSE**.) **ErrorCodeExtended** = RecEmpty(203): It runs out of receipt paper. (It returns only when **AsyncMode** is **FALSE**.) **ErrorCodeExtended** = Blackmark(10001): Black Mark error occurs. (It returns only when **AsyncMode** is **FALSE**.) **ErrorCodeExtended** = Fatal (10003): Fatal error occurs. (It returns only when **AsyncMode** is **FALSE**.) **ErrorCodeExtended** = Overheat (10006): Head Heat up error occurs. (It returns only when **AsyncMode** is **FALSE**.)
The others	Refer to the items of **ErrorCode**.

EndInsertion Method

Syntax

void EndInsertion ();

Remarks

Because this method is only applicable for the Slip Printers, this is not supported by this class library.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	The POS printer does not have the slip.
Other Values	Refer to the **ErrorCode** section.

EndRemoval Method

Syntax

void EndRemoval ();

Remarks

Because this method is only applicable for the Slip Printers, this is not supported by this class library.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	The POS printer does not have the slip.
Other Values	Refer to the **ErrorCode** section.

MarkFeed Method

Syntax

void MarkFeed (PrinterMarkFeeds *Type*);

Remarks

This method is not supported.

Exception

When you call this method, you may throw the following

PosControlException:.

Value	Meaning
Illegal(106)	The feature of handling mark sensed paper is not supported. (Refer to the **CapRecMarkFeed** Property.)
Other Values	Refer to the **ErrorCode** section.

PageModePrint Method

Syntax

void PageModePrint (PageModePrintControl *Control*);

Parameter	Description
Control	Page Mode control. See values below:

Control parameter values are as follows:

Value	Meaning
PageMode(1)	Enter Page Mode
Normal(3)	Print the print area and destroy the canvas and exit Page Mode.
Cancel(4)	Clear the page and exit the Page Mode without any printing of any print area.

Remarks

Enters or exits Page Mode for the station specified in the PageModeStation property.

If *control* is **PageMode**, then Page Mode is entered. Subsequent calls to **PrintNormal**, **PrintBarCode and PrintBitmap** will buffer the print data (either at the printer or the Service, depending on the printer capabilities) until **PageModePrint** is called with the *control* parameter set to Normal, or Cancel. (In this case, the print methods only validate the method parameters and buffer the data – they do not initiate printing. Also, the value of the **AsyncMode** property does not affect their operation: No **OutputID** will be assigned to the request, nor will an **OutputCompleteEvent** be enqueued.)

If *control* is Normal, the page mode ends and the state transits to the normal state. If some data is buffered by calls to the methods **PrintNormal**, **PrintBarCode**, **PrintBitmap**, and **PrintMemoryBitmap**, then the buffered data is printed. The buffered data will not be saved.

If *control* is Cancel, the page mode ends and the state transits to the normal state. If some data is buffered by calls to the methods **PrintNormal**, **PrintBarCode**, **PrintBitmap**, and **PrintMemoryBitmap**, then the buffered data is not printed and is not saved.

Note that when the **PageModePrint** method is called, all of the data that is to be printed in the PageModePrintArea will be printed and the paper is fed to the end of the PageModePrintArea.
If more than one PageModePrintArea is defined, then after the **PageModePrint** method is called, all of the data that is to be printed in the respective PageModePrintArea(s) will be printed and the paper will be fed to the end of the PageModePrintArea.

The entire Page Mode transaction is treated as one message. This method is performed synchronously if AsyncMode is false, and asynchronously if **AsyncMode** is true.

Calling the **ClearOutput** method cancels Page Mode. Any buffered print lines are also cleared.

The page mode function cannot use transaction printing and rotation printing at the same time. (For details, see "10. Page Mode Function" Exclusion of PageModePrint Method ")
If you start the page mode again while the page mode is starting, the print data buffered up to that point will be discarded.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Method finishes normally.
Illegal(106)	One of the following errors occurs. - Control does not exist. - Control is an invalid value. - TransactionPrint is starting. - RotationPrint is starting
NoHardware(107)	POS Printer is OFF or OFFLINE.
Failure(111)	This class library is in error state. After deleting error, execute it .
Busy(113)	It cannot perform because it outputting.
Extended(114)	**ErrorCodeExtended** = CoverOpen(201): POS Printer cover is open. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = RecEmpty(203): It runs out of paper. (Only when **AsyncMode** is

	FALSE, it is returned.) **ErrorCodeExtended** = Blackmark(10001): Black Mark Error occurs. Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Fatal (10003): Fatal error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Overheat (10006): Head overheat error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.)
The others	Refer to the items of **ErrorCode**.

PrintBarCode Method

Syntax

void PrintBarCode (PrinterStation *Station*, **string** *Data*, **BarCodeSymbology** *Symbology*, **int** *Height*, **int** *Width*, **int** *Alignment*, **BarCodeTextPosition** *TextPosition***);**

Parameter	**Description**
Station	Receipt(2) is assigned.
Data	Bar-code character string.
Symbology	Usable bar-code type (Refer to the values below)
Height	Bar code height, which is written in the unit defined by **MapMode** It is possible to set from 1 to 255dot. In case of PDF417 printing with upright/upside-down, it is possible to set from 1 to 831(dot). Or, in case of 90 degree rotation to the right or to the left, it is possible to set from 12 to the value of **RecLineWidth** Property (dot). Specify the range of 1-16 for width of the module for the QR code and micro QR code of two dimension bar code. Illegal(106) is notified when other values are specified.
Width	Bar code width, which is written in the unit defined by **MapMode**.
Alignment	Bar code location Refer to the value below.
TextPosition	Position of character strings. Refer to the value below.

Values of the *Symbology* Parameter of this release are just the following

Value	**Label type**
Upca(101)	UPC-A
Upce(102)	UPC-E
Ean8(103)	EAN 8 (= JAN 8)
Jan8(103)	JAN 8 (= EAN 8)
Ean13(104)	EAN 13 (= JAN 13)
Jan13(104)	JAN 13 (= EAN 13)
Itf(106)	Interleaved 2 of 5
Codabar(107)	Codabar(NW-7)
Code39(108)	Code 39
Code93(109)	Code 93
Code128(110)	Code 128
Pdf417(201)	PDF 417
QRCode(204)	QR code (two dimension barcode)
MicroQRCode(205)	Micro QR code (two dimension barcode)

The values of *Alignment* Parameter are just the following.

Value	**Meaning**
Left(-1)	Align with the left-most print column. (Because it is executed to Printing data, it is to align right toward POS Printer Printing direction in case of 180 degree rotation printing.)
Center(-2)	Align in the center of the station. In the case of two dimension barcode, this parameter is not supported in 90 degrees rotating to the left or to the right. it works as Left(-1).
Right(-3)	Align with the right-most print column. (Because it is executed to Printing data, it is to align left toward POS Printer Printing direction in case of 180 degree rotation printing.) In the case of two dimension barcode, this parameter is not supported in 90 degrees rotating to the left or to the right. It works as Left(-1).
The others	Distance from the left-most print column to the start of the bar code. It is written in the unit defined by **MapMode**. Illegal(106) will return, in the case that actual bar code printing width, calculated by OPOS from the bar code width assigned by *Width* Parameter plus the distance from the left-most print column exceeds **RecLineWidth** Property value. However, in the case that Right90(257), Left90(258) are assigned in **RoatateSpecial**

Property, Left(-1) is considered to be assigned and printing is done.
In the case of PDF417, this parameter is not supported in 90 degrees rotating to the left or to the right. it works as Left(-1).

TextPosition Parameter values are just the following.

Value	**Meaning**
None(-11)	No text is printed. Only prints the bar code.
Above(-12)	Print the text above the bar code.
Below(-13)	Print the text below the bar code.

Remarks

This method is called when to print bar codes with assigned POS Printer.

This method is executed synchronously if **AsyncMode** is **FALSE,** and asynchronously if **AsyncMode** is **TRUE.**

The values of Alignment Parameter in page mode are just the following.

Value	**Meaning**
Left(-1)	Align with the left-most print column.
Center(-2)	Align in the center of the station.
Right(-3)	Align with the right-most print column.
The others	Distance from the left-most print column to the start of the bar code. It is written in the unit defined by **MapMode**. The sum of *Width* and *Width* must not exceed the limit of the *Width* parameter.

.

Following are printable bar code conditions per each *Symbology.*

<MONO>

Symbology	Each printable character kind	Upright/Upside-down mode		90 degree rotation to the left/right	
		Character string length	Width (dots)	Character string length	Width (dots)
Upca *1	10 kinds ('0'-'9')	11-12	95-RecLineWidth Value	11-12	95-1662
Upce *1		11-12	51-RecLineWidth Value	11-12	51-1662
Jan8 *1		7-8	67-RecLineWidth Value	7-8	67-1662
Jan13 *1		12-13	95-RecLineWidth Value	12-13	95-1662
Code39 *1	43 kinds ('0'-'9', 'A'-'Z', space, '$', '%', '+', '-', '.', '/') (Start/Stop character of '*' is automatically added.)	1-34	47-RecLineWidth Value	1-101	47-1662
Itf *1	10 kinds ('0'-'9')	2-62	27-RecLineWidth Value	2-182	27-1662
Codabar *1	20 kinds ('0'-'9', 'A'-'D', '$', '+', '-', '.', '/', ':')	3-47	41-RecLineWidth Value	3-138	41-1662
Code93 *1	128 kinds (0x00-0x7F) (Lower stage is for two characters)	1-59 1-29	46-RecLineWidth Value	1-88 1-44	46-1662
Code128 *1	Code Set A: 0x00 - 0x5F Code Set B 0x20 - 0x7F Code Set C 0x00 - 0x63 However the characters including "{" are exception. For details refer to later.	3-51	46-RecLineWidth Value	3-74	46-1662

Symbology	Each printable character kind	Upright/Upside-down mode		90 degree rotation to the left/right	
		Character string length	Width (dots)	Character string length	Width (dots)
Pdf417	256 kinds including 0x00 to 0xFF. The character strings 0x00 to 0x7F conform to the ASCII code, and 0x80 to 0xFF conform to the extended character sets in the English table of PC437 (USA: Standard Europe)	1-1069	172-RecLineWidth Value	1-1069	172-831
QRCode	Figure (‘0’ ～ ‘9’)、 Capital letter (‘A’～’Z’)、 Special sign (space、‘$’, ‘%’, ‘*’ ,‘+’, ‘-’, ‘.’, ‘/’, ‘:’)、 Binary (0x00 ～ 0xFF)	1-7089	21-RecLineWidth Value	1-7089	21-RecLine Width Value
QRCode	Figure (‘0’ ～ ‘9’)、 Capital letter (‘A’～’Z’)、 Special sign (space、‘$’, ‘%’, ‘*’ ,‘+’, ‘-’, ‘.’, ‘/’, ‘:’)、 Binary (0x00 ～ 0xFF)	1-35	11-RecLineWidth Value	1-35	11-RecLine Width Value

*1 When the setup value of Width is minimum value (exclude "0") or less, the following barcodes are printed by a recommended dot width.

Upca, Upce, Jan8, Jan13, Code39, Itf, Codabar, Code93, Code128

Following is printing width decision algorithm for each bar code. As for final printing width (dot), printing is done with nearest value not exceeding the value assigned by *Width* Parameter of **PrintBarcode**, in changing parameters.

Symbology	Formula for calculating printing width
Upca	Bar code width = 95 * dotNarrow
Upce	Bar code width = 51 * dotNarrow
Jan8	Bar code width = 67 * dotNarrow
Jan13	Bar code width = 95 * dotNarrow
Code39	Bar code width = 6 * dotNarrow + 3 * dotWide + 1 * dotNarrow + (6 * dotNarrow + 3 * dotWide + 1 dotNarrow)* Length + 6 * dotNarrow + 3 * dotWide (Length = Number of characters printed)
Itf	Bar code width = 4 * dotNarrow + (3 * dotNarrow + 2 * dotWide) * Length + 2 * dotNarrow + 1 * dotWide (Length = Number of characters printed)
Codabar	Bar code width = (5 * dotNarrow + 2 * dotWide) * (Length – Wlen) + (4 * dotNarrow + 3 * dotWide) * Wlen + 1 * dotNarrow * (Length + 1) (Length = Number of characters printed) (Wlen = Number of characters among ":", "/", ".", "+", "A", "B", "C", "D")
Code93	Bar code width = 9 * dotNarrow + 9 * dotNarrow * Wlen + (9 * dotNarrow) * 2 * (Length - Wlen) + 9 * 2 * dotNarrow + 10 * dotNarrow (Length = Number of characters printed) (Wlen =Number of characters among "0"-"9", "A"-"Z", " ", "$", "%", "+", "-", "/")

Symbology	Formula for calculating printing width
Code128	Bar code width = 11 * dotNarrow * (Length + 1) + 13 * dotNarrow (Length = Number of characters printed – Special characters*[1]) *1: Number of characters which head is "{". If "{AA{BA" is assigned, Length = 6 – 2 = 4.
Pdf417	Bar code width = ((17 * (C + 2)) + (17 + 18)) * X Bar code height = RYX C: Number of columns X: Nominal width of narrow element R: Number of rows Y: Row height * For number of rows and number of columns, the minimum value that input data can convert as the bar code is selected. For nominal width of narrow element and row height, after number of rows and number of columns are determined, maximum size that does not exceed the Width and Height parameters is selected.
QRCode	Bar code width = Number of cells of vertical directions * Module width (1 – 16) Bar code height = Horizontal number of cells * Module width (1 – 16) * Width and height of the bar code are set to the maximum size that doesn't exceed the value of Width.
MicroQRCode	Bar code width = Number of cells of vertical directions * Module width (1 – 16) Bar code height = Horizontal number of cells * Module width (1 – 16) * Width and height of the bar code are set to the maximum size that doesn't exceed the value of Width.

*Relation between dotNarrow and dotWide

dotNarrow	1	2	3	4	5	6
dotWide	3	5	9	11	14	18

Notes for Bar Code Printing

1. When to print CODE39, "*" (Start/Stop Character) is automatically added. So, there is no deed of setting up in Character.

2. When to assign ITF, even-number character must be assigned. If odd-number is assigned, the character of ‘0’ will be added at the head of the characters.

3. When to assign CODABAR, the head and the tail of the characters must be among "A" - "D". Accordingly, three or more than three characters (the head character plus any characters plus the tail character) must be assigned. In the other cases, Illegal(106) returns.

4. When to assign UPC-E, development is done according to the following list. UPC-A Left Code shows top characters (2-6), UPC-A Right Code shows 7th-11th characters. The shortened code is actually printed as UPC-E. If the UPC-A top character assigned is except 0 or, characters not based on the following list is assigned, Illegal(106) returns

   Example 05810000226 -> Converted to c58226.
   09859363583 -> Illegal returns.

Manufacturer Code **Left Code for UPC-A**					**Item Code** **Right Code for UPC-A**					**Shortened Code**					
F1	**F2**	**F3**	**F4**	**F5**	**A1**	**A2**	**A3**	**A4**	**A5**	**Z1**	**Z2**	**Z3**	**Z4**	**Z5**	**Z6**
0-9	0-9	0	0	0	0	0	0-9	0-9	0-9	F1	F2	A3	A4	A5	0
0-9	0-9	1	0	0	0	0	0-9	0-9	0-9	F1	F2	A3	A4	A5	1
0-9	0-9	2	0	0	0	0	0-9	0-9	0-9	F1	F2	A3	A4	A5	2
0-9	0-9	3-9	0	0	0	0	0	0-9	0-9	F1	F2	F3	A4	A5	3
0-9	0-9	0-9	1-9	0	0	0	0	0	0-9	F1	F2	F3	F4	A5	4
0-9	0-9	0-9	0-9	1-9	0	0	0	0	5-9	F1	F2	F3	F4	F5	A5

5. When to print CODE128, set up characters as followed.

   1. One of "{A", "{B", "{C" must be assigned as the head of the bar code. Following that, each of CODE A, CODE B, CODE C must be set up.
   2. When to assign Function Code, assign "{1", "{2", "{3", or "{4". Each is to assign FNC1, FNC2, FNC3, or FNC4. For further information, in CODE C, only FUNC1is available. If you assign except FUNC1 in CODE C, Illegal(106) returns.
   3. When to print "{" in CODE B, assign "{{".
   4. When to set up SHIFT, assign "{S". After that, code set of one character sifts like CODE A <- -> CODE B. If you assign in CODE C, Illegal(106) returns.

Following are printable character in CODE A, CODE B, CODE C.

Character to Print			**Character to Print**		
CODE-A	**CODE-B**	**CODE-C**	**CODE-A**	**CODE-B**	**CODE-C**
SPACE	SPACE	00(00H)	U	U	53(35H)
!	!	01(01H)	V	V	54(36H)
"	"	02(02H)	W	W	55(37H)
#	#	03(03H)	X	X	56(38H)
$	$	04(04H)	Y	Y	57(39H)
%	%	05(05H)	Z	Z	58(3AH)
&	&	06(06H)	[	[	59(3BH)
'	'	07(07H)	/	/	60(3CH)
(	(	08(08H)	]	]	61(3DH)
)	)	09(09H)	^	^	62(3EH)
*	*	10(0AH)	_	_	63(3FH)
+	+	11(0BH)	NULL(00H)	`	64(40H)
,	,	12(0CH)	SOH(01H)	a	65(41H)
-	-	13(0DH)	STX(02H)	b	66(42H)
.	.	14(0EH)	ETX(03H)	c	67(43H)
/	/	15(0FH)	EOT(04H)	d	68(44H)
0	0	16(10H)	ENG(05H)	e	69(45H)
1	1	17(11H)	ACK(06H)	f	70(46H)
2	2	18(12H)	BEL(07H)	g	71(47H)
3	3	19(13H)	BS(08H)	h	72(48H)
4	4	20(14H)	HT(09H)	i	73(49H)
5	5	21(15H)	LF(0AH)	j	74(4AH)
6	6	22(16H)	VT(0BH)	k	75(4BH)
7	7	23(17H)	FF(0CH)	l	76(4CH)
8	8	24(18H)	CR(0DH)	m	77(4DH)
9	9	25(19H)	SO(0EH)	n	78(4EH)
:	:	26(1AH)	SI(0FH)	o	79(4FH)
;	;	27(1BH)	DLE(10H)	p	80(50H)
<	<	28(1CH)	DC1(11H)	q	81(51H)
=	=	29(1DH)	DC2(12H)	r	82(52H)
>	>	30(1EH)	DC3(13H)	s	83(53H)

Character to Print			Character to Print		
CODE-A	**CODE-B**	**CODE-C**	**CODE-A**	**CODE-B**	**CODE-C**
?	?	31(1FH)	DC4(14H)	t	84(54H)
@	@	32(20H)	NAK(15H)	u	85(55H)
A	A	33(21H)	SYN(16H)	v	86(56H)
B	B	34(22H)	ETB(17H)	w	87(57H)
C	C	35(23H)	CAN(18H)	x	88(58H)
D	D	36(24H)	EM(19H)	y	89(59H)
E	E	37(25H)	SUB(1AH)	z	90(5AH)
F	F	38(26H)	ESC(1BH)	{ "{{"	91(5BH)
G	G	39(27H)	FS(1CH)	|	92(5CH)
H	H	40(28H)	GS(1DH)	}	93(5DH)
I	I	41(29H)	RS(1EH)	~	94(5EH)
J	J	42(2AH)	US(1FH)	DEL	95(5FH)
K	K	43(2BH)			96(60H)
L	L	44(2CH)			97(61H)
M	M	45(2DH)			98(62H)
N	N	46(2EH)			99(63H)
O	O	47(2FH)	Following are used by assigning "{"		
P	P	48(30H)	FNC 3 "{3"	FNC 3 "{3"	
Q	Q	49(31H)	FNC 2 "{2"	FNC 2 "{2"	
R	R	50(32H)	SHIFT "{S"	SHIFT "{S"	
S	S	51(33H)	CODE C "{C"	CODE C "{C"	
T	T	52(34H)	CODE B "{B"	CODE A "{A"	CODE B "{B"
			FNC 4 "{4"	FNC 4 "{4"	CODE A "{A"
			FNC 1 "{1"	FNC 1 "{1"	FNC 1 "{1"

6. Following are *TextPosition* assignment, and bar code printing possibility condition according to *Width*. As for *Width* Parameter, if without special description, they mean that it is possible to print bar codes independently on *TextPosition,* within the printable area. As for the following list, it is prerequisite that *Width* Parameter is in units of dots and that it is within the printable area.

Symbology	***TextPosition*** **None**	***TextPosition*** **Above** **Below**
Upca	Printable	Width=95 - 189 Illegal
Upce	Printable	Width=51 - 101 Illegal
Jan8	Printable	Width=67 - 133 Illegal
Jan13	Printable	Width=95 - 189 Illegal
Code39	Printable	Printable
Itf	Printable	Printable
Codabar	Printable	Printable
Code93	* 1	* 1
Code128	* 2	* 2

*1: Relation between *width* and character number which become Illegal is within the following range.

$37 + 9 * wlen + 18(len - wlen) <= width < 74 + 18 * wlen + 36(len - wlen)$

- wlen = the number of '0' - '9', 'A' - 'Z', ' ', '$', '%', '+', '-', '.', '/' within character
- len = Character length

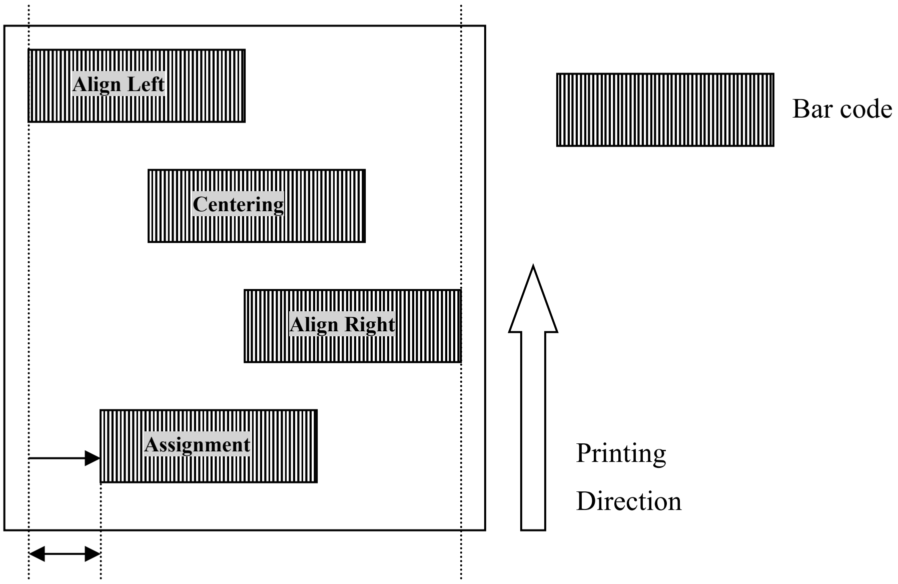
*2: Relation between width and character number which become Illegal is within the following range.

$24 + 11 * len <= width < 48 + 22 * len$

- len = The gained value by subtracting the number of "{A", "{B", "{C", "{1", "{2", "{3", "{4", "{S", "{{" (which are included in the character length) from the character length.

Bar Code Rotation Printing Using Rotate Special

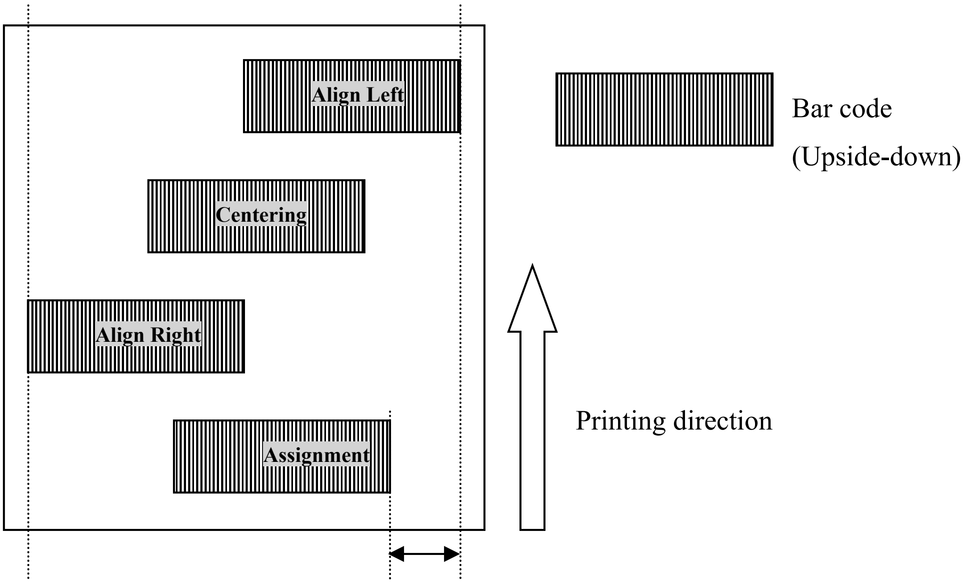
Printing-position-change by *Alignment* assignment, at the time of upright printing.



Position from the left most print column set by *Alignment*

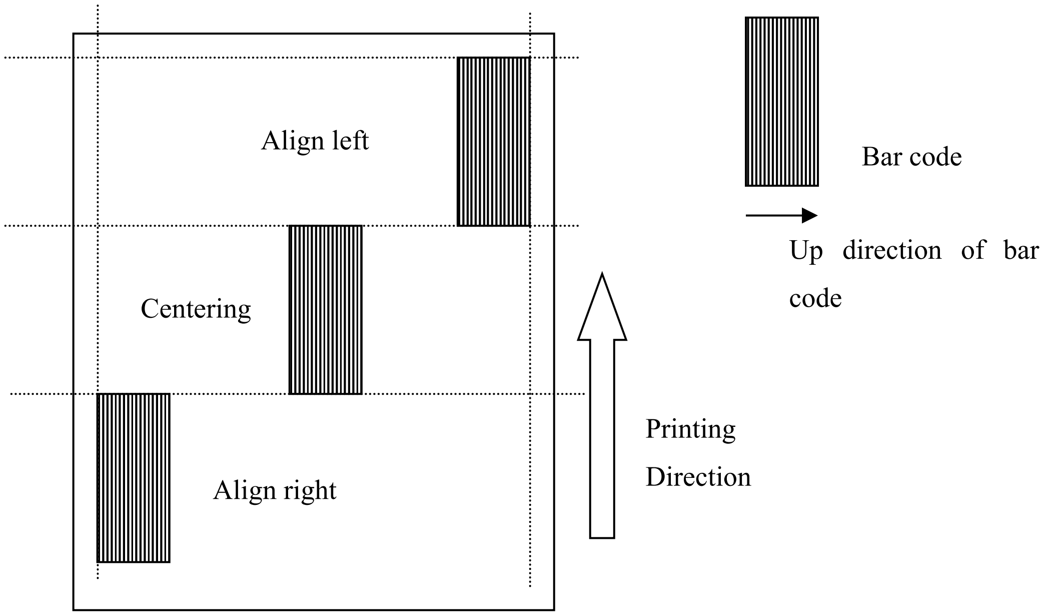
Printing-position-change by *Alignment* assignment, at the time of upright printing.

Printing-position-change by *Alignment* assignment, at the time of upside-down printing.

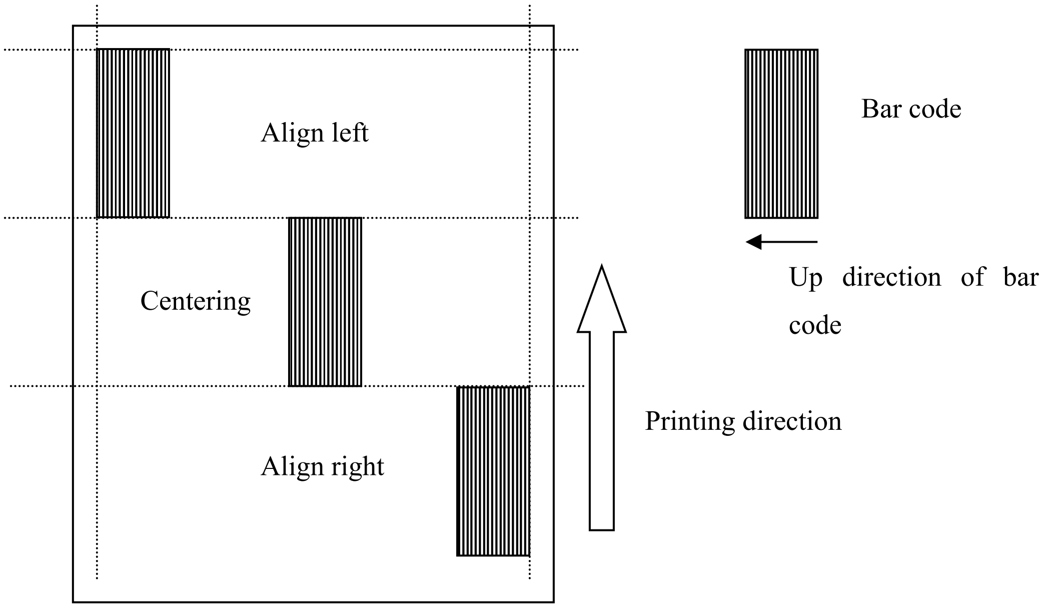


Position from the left most print column set y *Alignment*

Printing-position-change by *Alignment* assignment, at the time of 90-degree-rotation-to-the-right printing.



Printing-position-change by *Alignment* assignment, at the time of 90-degree-rotation-to-the-left printing.



*In case of two dimension barcode, all Aligment is fixed to Left(-1) at the time of 90-degree-rotation-to-the-left/right printing.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Method finishes normally.
Illegal(106)	One of the following errors occurs. - No Station - Station does not support bar code printing. - Height or Width is 0 or too large. - Symbology is not supported. - There is a character not supported by Symbology. - Alignment is illegal value or too large. (When the total value of assigned Alignment value plus actual bar code printing width (the value calculated using the nearest value to Width) exceeds printable width, in case of Alignment absolute location assignment. - TextPosition is illegal value.
NoHardware(107)	POS Printer is OFF or OFFLINE.
Failure(111)	This class library is in error state. After deleting error, execute it .
Busy(113)	It cannot perform because it outputting.
Extended(114)	**ErrorCodeExtended** = CoverOpen(201): POS Printer cover is open. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = RecEmpty(203): It runs out of paper. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Blackmark(10001): Black Mark Error occurs. Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Fatal (10003): Fatal error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Overheat (10006): Head overheat error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.)
The others	Refer to the items of **ErrorCode**.

PrintBitmap Method

Syntax

void PrintBitmap (PrinterStation *Station*, **string** *FileName*, **int** *Width***, int** *Alignment***);**

Parameter	**Description**
Station	It assigns Receipt(2)
FileName	Windows bitmap file name. The files must be uncompressed format. (Assign full pass or relative pass)
Width	Bitmap printing width. Refer to the following values.
Alignment	Bitmap printing position. Refer to the following values.

The *Width* parameter has one of the following values:

Value	**Meaning**
Asis(-11)	Prints the bitmap with one bitmap pixel per POS Printer dot.
The others	Bitmap width. It writes in the units defined by **MapMode**. Available values are from 1 to RecLineWidth Property value.

The *Alignment* parameter has one of the following values:

Value	**Meaning**
Left(-1)	Align left
Center(-2)	Centering
Right(-3)	Align right
The others	Distance from the left-most print column to the start of the bitmap. It is written in the units defined by **MapMode**. Total value with *Width* must not exceed restriction of *Width* Parameter. This parameter is not supported in 90 degrees rotating to the left or to the right. It works as Left(-1).

Remarks

This method is called to print a bitmap on the specified printer. The bitmap is converted to monochrome or 2 colors and printed.

When 2-color printing is set, black is printed as the first color and red is printed as the second color.

The size of the bitmap that can be registered is the horizontal size (W*idth*) that is the dots of **RecLineWidth** or less (when Alignment is absolute position specified, Width + Alignment <= **RecLineWidth**) and the vertical size that is 1662 dot for single color data and 831 dot or less for two color data with 2 color setting.

Because **PrintBitmap** sends bitmap data to the printer at the time of being called, the performance is not high. It is recommended to print the bitmap with **SetBitmap** and the Escape Sequence.

This method is synchronously executed if **AsyncMode** is **FALSE,** and asynchronously if **AsyncMode** is **TRUE**.

Width Parameter controls transformation of the bitmap. If *Width* is Asis, then no transformation is performed. The bitmap is printed with one bitmap pixel per one POS Printer dot.

If *Width* is not 0, then the bitmap will be transformed by stretching or compressing the bitmap such that its width is the specified width and the aspect ratio is unchanged.

* When the specified bitmap data is in monochrome, monochrome bitmap is set in the printer. For data other than in monochrome, when the **CapRec2Color** property is TRUE, 2-color bitmap printing is performed. When the property is FALSE, it is printed as monochrome data.

The values of Alignment Parameter in page mode are just the following.

Value	**Meaning**
Left(-1)	Align with the left-most print column.
Center(-2)	Align in the center of the station.
Right(-3)	Align with the right-most print column.
The others	Distance from the left-most print column to the start of the bitmap. It is written in the unit defined by **MapMode**. The sum of *Width* and *Width* must not exceed the limit of the *Width* parameter.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Method ends normally.
Illegal(106)	One of the following errors occurred. - No Station. - Station does not support bitmap printing. - Width is too large. - Alignment is illegal value or too large.
NoHardware(107)	POS Printer is OFF or OFFLINE.
Failure(111)	This class library is in error state. After deleting error, execute it .
NoExist(109)	It could not find the file assigned by *FileName*.
Busy(113)	It cannot perform because it outputting.
Extended(114)	**ErrorCodeExtended** = CoverOpen(201): POS Printer cover is open. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = RecEmpty(203): It runs out of paper. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Toobig (206): Assigned bitmap is too large. Printable bitmap size is: Width (number of dot of **RecLineWidth** Property) Height (monochrome: 1662 dot, 2-clor setting: 831 dot) **ErrorCodeExtended** = Badformat(207): Bitmap format is different from the assigned one. The assigned file is not bitmap file. When 2-color printing is set and data is the 24 bit bitmap. (When monochrome is specified, printing 24 bit bitmap is available.) **ErrorCodeExtended** = Blackmark(10001): Black Mark Error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Fatal(10003): Fatal error occurs. (Only when **AsyncMode** is **FALSE**, it is returned.) **ErrorCodeExtended** = Overheat (10006): Head overheat occurs. (Only when **AsyncMode** is **FALSE,** it is returned.)
The others	Refer to the items of **ErrorCode**.

PrintImmediate Method

Syntax

void PrintImmediate (PrinterStation *Station*, **string** *Data***);**

Parameter	**Description**
Station	It assigns Receipt(2).
Data	Characters to be printed. It consists of printable characters, Escape Sequence, Carriage return (13 decimal), and New line/line feed (10 decimal).

Remarks

This method is called when to print *Data* with POS Printer. In executing asynchronous printing (**state**=Busy(3)), Busy(113) returns. During an error event (state=Error(4)), Failure(111) returns. It performs a reverse line feed in the case that characters per line of the text exceed maximum-characters-per-line.

The special character value within *Data* is as follows:

Value	**Meaning**
New line/Line Feed (10)	After printing the data in the buffer, it feeds to the next print line. (No need of carriage return for printing the line.)
Carriage Return(13)	If a Carriage Return immediately precedes a Line Feed, it is ignored. Carriage Return acts like a Line Feed. **ValidateData** Method is used to determine whether a Carriage Return without Line Feed is possible and whether a reverse line feed is required to support it.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Method ends normally.
Illegal(106)	No POS Printer assigned (except receipt)
NoHardware(107)	POS Printer is OFF or OFFLINE.
Failure(111)	This class library is in error state. After deleting error, execute it .
Busy(113)	It cannot perform because it outputting.
Extended(114)	**ErrorCodeExtended** = CoverOpen(201): POS Printer cover is open. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = RecEmpty(203):It runs out of paper. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Blackmark(10001): Black Mark Error occurs. Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Fatal (10003): Fatal error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Overheat (10006): Head overheat error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.)
The others	Refer to the items of **ErrorCode**.

PrintMemoryBitmap Method

Syntax

void PrintMemoryBitmap (PrinterStation *Station*,

System.Drawing.Bitmap *Data*, **int** *Width*, **int** *Alignment***);**

Parameter	**Description**
Station	It assigns Receipt(2)
Data	An array of bytes holding the bitmap data.
Width	Printed width of the bitmap to be performed. See values below.
Alignment	Placement of the bitmap. See values below.

The *Width* parameter has one of the following values:

Value	**Meaning**
Asis(-11)	Prints the bitmap with one bitmap pixel per POS Printer dot.
Other Values	Bitmap width. Expressed in the unit of measure given by **MapMode**.

The *Alignment* parameter has one of the following values:

Value	**Meaning**
Left(-1)	Align left
Center(-2)	Centering
Right(-3)	Align right
The others	Distance from the left-most print column to the start of the bitmap. Expressed in the unit of measure given by **MapMode**.

Remarks

This method is called to print a memory-stored bitmap on the specified printer station. The bitmap passed as the pointer to the byte array is converted to monochrome or 2 colors and printed.

When 2 color printing is set, black is printed as the first color and red is as the second color.

The size of the bitmap that can be registered is the horizontal size (W*idth*) that is the dots of **RecLineWidth** or less (when Alignment is absolute position specified, Width + Alignment <= **RecLineWidth**) and the vertical size that is 1662 dot for single color data and 831 dot or less for two color data with 2 color setting.

This method is performed synchronously if **AsyncMode** is **FALSE**, and asynchronously if **AsyncMode** is **TRUE**.

The W*idth* parameter controls transformation of the bitmap. If width is Asis, then no transformation is performed. The bitmap is printed with one bitmap pixel per POS printer dot. Advantages of this option are that it:

- Provides the highest performance bitmap printing.
- Works well for bitmaps tuned for a specific printer's aspect ratio between horizontal dots and vertical dots.

If *Width* is not 0, then the will be transformed by stretching or compressing the bitmap such that its width is the specified width and the aspect ratio is unchanged.

Because it is not buffered to TransactionPrint, data can be sent to the printer in the middle of buffering.

* If gradation and dither are specified for bitmap printing, printing will be performed in grayscale.

Refer to the PrintBitmap property for printable image formats.

PrintMemoryBitmap method during page mode start is not supported.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Method ends normally.
Illegal(106)	One of the following errors occurred. - No *Station*. - *Station* does not support bitmap printing. - *Width* is too large. - *Alignment* is illegal value or too large.
NoHardware(107)	POS Printer is OFF or OFFLINE.
Failure(111)	This class library is in error state. After deleting error, execute it.
Busy(113)	It cannot perform because it outputting.
Extended(114)	**ErrorCodeExtended** = CoverOpen(201): POS Printer cover is open. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = RecEmpty(203): It runs out of paper. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Toobig (206): Assigned bitmap is too large. Printable bitmap size is: Width (number of dot of **RecLineWidth** Property) Height (monochrome: 1662 dot, 2-clor setting: 831 dot) **ErrorCodeExtended** = Badformat(207): Bitmap format is different from the assigned one. The assigned file is not bitmap file. **ErrorCodeExtended** = Blackmark(10001): Black Mark Error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Fatal(10003): Fatal error occurs. (Only when **AsyncMode** is **FALSE**, it is returned.) **ErrorCodeExtended** = Overheat (10006): Head overheat occurs. (Only when **AsyncMode** is **FALSE,** it is returned.)
The others	Refer to the items of **ErrorCode**.

PrintNormal Method

Syntax

void PrintNormal (PrinterStation *Station*, **string** *Data***);**

Parameter	Description
Station	It assigns Receipt(2).
Data	Characters to be printed　It consists of printable characters, Escape Sequence, Carriage return (13 decimal), and New line/line feed (10 decimal).

Remarks

This method is called when to print *Data* with POS Printer. It performs a reverse line feed in the case that characters per line of the text exceed maximum-characters-per-line.

This method is executed synchronously if **AsyncMode** is **FALSE,** and asynchronously if **AsyncMode** is **TRUE**.

The special character value within *Data* is as follows:

Value	Meaning
New line/Line Feed (10)	After printing the data in the buffer, it feeds to the next print line. (No need of carriage return for printing the line.)
Carriage Return(13)	If a Carriage Return immediately proceeds a Line Feed, it is ignored. CarriageReturn acts like a Line Feed. **ValidateData** Method is used to determine whether a Carriage Return without Line Feed is possible and whether a reverse line feed is required to support it.

Exception

When you call this method, you may throw the following

PosControlException:.

Value	Meaning
No Exception	Method ends normally.
Illegal(106)	No POS Printer assigned (except receipt)
NoHardware(107)	POS Printer is OFF or OFFLINE.
Failure(111)	This class library is in error state. After deleting error, execute it .
Busy(113)	It cannot perform because it is outputting.
Extended(114)	**ErrorCodeExtended** = CoverOpen(201): POS Printer cover is open. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = RecEmpty(203): It runs out of paper. (Only when **AsyncMode** is

	FALSE, it is returned.) **ErrorCodeExtended** = Blackmark(10001): Black Mark Error occurs. Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Fatal (10003): Fatal error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Overheat (10006): Head overheat error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.)
The others	Refer to the items of **ErrorCode** and other items.

PrintTwoNormal Method

Syntax

void PrintTwoNormal (PrinterStation *Stations*, string *Data1*, string *Data2*);

Parameter	Description
Station	POS Printer station to be used.
Data1	Characters to be printed on the first station.
Data2	Characters to be printed on the second station.

Remarks

This method is called to print two character strings on two print stations simultaneously.

In this class library, this method is not supported because it is subject to slip printers.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	No POS Printer assigned(except receipt)
The others	Refer to the items of **ErrorCode**.

RotatePrint Method

Syntax

void RotatePrint (PrinterStation *Station*, void *Rotation*);

Parameter	Description
Station	Receipt(2) is assigned.
Rotation	Rotation Direction. Refer to the following values.

The *Rotation* value is as follows:

Value	Meaning
Right90(257)	Start of 90 degree rotation printing to the right (clockwise)
Left90(258)	Start of 90 degree rotation printing to the left (counterclockwise)
Rotate180(259)	Start of 180 degree rotation printing (upside-down)
Normal(1)	End of rotation printing

Remarks

This method is executed synchronously if **AsyncMode** is **FALSE,** and asynchronously if **AsyncMode** is **TRUE**.
If *Rotation* is Rotate180, upside-down printing mode starts. The data called by **PrintNormal** and **PrintImmediate RotatePrint** is printed upside-down till it is called at the *Rotation* Parameter setting of Normal. The lines are printed in the order that they are sent to POS printer Control, and with the start of each line at the right margin of the printer. Printing methods of **PrintNormal** and **PrintImmediate** are used in upside-down printing mode.

When *Rotation* is Right90, Left90, sideways printing mode starts. Data called by **PrintNormal** Method is buffered till it is called at the *Rotation* Parameter setting of Normal. (In this case, the above method data is only buffered and printing does not start. At the same time, **AsyncMode** Property value does not affect the operation. In other words, **OutputID** is not assigned to the request and does not notify **OutputCompleteEvent,** either. In addition to this, each method succeeds, even if it is in error state in this case. For example, even though the power of POS Printer is off, error does not return in printing data buffering of **RotatePrint**.)

In case of sideways printing, width is automatically set to 0 to 1662dot for monochrome and 0 - 831dot for 2-color by the character data buffered by **PrintNormal** Method call. OPOS Control analyses character data buffered, and decides the width adjusting the maximum values of the width for all the lines (Refer to the following list). When BitMap print by the escape sequence and bar code print were appointed in letter data, the print of

BitMap and the bar code which do not fit into the width calculated by other letter data is not performed normally because inclusion of the width is not performed.
If the width of total character number exceeds 1662dot (831dot.for 2-color), printing width is 1662dot (831dot.for 2-color). The left data is printed, and it performs a reverse line feed within the page. In case, the width of character data is double size or more, by Escape sequence, the value is calculated by multiplying its multiple. (Example: In the case that it is assigned that font is A and that ANK character is double-width, the calculation result is 24 dots.)
If there is no buffered data, (when PrintNormal Method is not executed before) no printing is done.

Width per a character (dot)

Font (Refer to RecLineChars Property)	ANK	kanji
Font A	12 dot	24 dot
Font B	10 dot	20 dot
Font C * Only Standard mode	8 dot	16 dot

When **PrintBitmap** and **PrintMemoryBitmap** are issued in upside-down printing mode, bitmap is printed upside-down.
For the bitmap performed **SetBitmap** in upside-down printing mode, it is registered without upside-down.

When *Rotation* contains BarCode or Bitmap, the barcode (printed with **PrintBarCode**) or the bitmap (printed with **PrintBitmap** or the 'ESC|#B' escape sequence) can be printed upside-down. Their directions of rotation are controlled by the **RecBarCodeRotationList** and **RecBitmapRotationList** properties respectively.
When *Rotation* contains BarCode, the contents of **RotateSpecial** are ignored.

When *Rotation* is Normal, the rotation printing mode ends. If particular data is buffered by **PrintNormal** while sideways rotation printing mode is effective, the buffered data is printed. One whole block of rotated lines is treated just as one message.

When **ClearOutput** is called, the Rotation printing mode is terminated. Any buffered sideways printing line is deleted.

When the vertical length is specified by Escape Sequence with "n" times and print rotated,
Print may be overlapped or exceed the paper. In this case, input "LF" code before Escape Sequence to specify vertical length.

(Ex. To print the data 3 times length vertically, input "LF" code twice. To print the data "n" times length vertically, input "LF" code with the number of "n-1" times.)

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Method ends normally
Illegal(106)	No assigned POS Printer (except receipt) *Station* does not support assigned rotation. In different rotation mode, assign Normal(1) or re-execute after clearing rotation printing with **ClearOutput.** The page mode is starting. Please re-execute after exiting page mode.
NoHardware(107)	POS Printer is OFF or OFFLINE.
Failure(111)	This class library is in error state. After deleting error, execute it.
Busy(113)	It cannot perform because it is outputting.
Extended(114)	**ErrorCodeExtended** = CoverOpen(201): POS Printer cover is open. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = RecEmpty(203):It runs out of paper. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Blackmark(10001): Black Mark Error occurs. Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Fatal (10003): Fatal error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Overheat (10006): Head overheat error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.)
The others	Refer to the items of **ErrorCode**.

SetBitmap Method

Syntax

void SetBitmap (int *BitmapNumber*, **PrinterStation** *Station*, **string** *FileName*, **int** *Width***, int** *Alignment***);**

Parameter	Description
BitmapNumber	Number assigned to this bitmap. Valid values are from 1 to 20.
Station	Assign Receipt(2).
FileName	Windows bitmap file name. File must be uncompressed format. (To assign full pass or relative pass) If null number is set up, delete assigned *BitmapNumber* bit map from POSPrinter.
Width	Bitmap printing width. For value, refer to **PrintBitmap**.
Alignment	Location of bitmap printing. For value, refer to **PrintBitmp**.

Remarks

It is called when to save the information concerning bitmap soon to be printed.

Bitmap is printed by calling **PrintNormal** or **PrintImmediate** which has bitmap printing Escape Sequence inside printing data.

When 2-color printing is set, black is stored as the first color and red is stored as the second color

The bitmap that can be registered must be **RecLineWidth** dot (*Width*) or less (if *Alignment* is set to absolute position*,* it is *Width*+ *Alignment*<= **RecLineWidth),** and must be vertical size of 2304dot or less and the data size is 384 KB or less after dithering (after converting the data into interpretable bitmap data for POS Printer). When these conditions are not met, Toobig(206) is issued. In addition, when there is no free space on the nonvolatile memory, Toobig(206) is issued. In such case, set empty space in the *FileName* parameter to secure free space by deleting the bitmap data from the POS printer, then execute again.

When 2-color printing is set, the bitmap data in 24 bit color cannot be registered.

* In this class library, when **SetBitmap** is executed, the bitmap that is set is effective even after executing **Release** by this class library, because the bitmap is written on nonvolatile memory in the POS Printer. In other words,

once setting is complete, bitmap printing is effective with Escape Sequence.

* When the specified bitmap data is in monochrome, monochrome bitmap is set in the printer. For data other than in monochrome, when the **CapRec2Color** property is TRUE, 2-color bitmap printing is performed. When the property is FALSE, it is printed as monochrome data.

* When the bitmap registered to the parameter of the method by Left(-1), Center(-2) or Right(-3) is printed in 90 degrees rotating to the left or to the right by the **RotatePrint** method, the bit map is aligned left, centered or aligned right based on the printing standard of the normal direction (vertical).

Refer to the PrintBitmap property for registerable image formats.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Method ends normally.
Illegal(106)	No assigned POS Printer (except receipt) - *BitmapNumber* is illegal value - *No* POS Printer (except receipt) - *Station* does not support bitmap printing. - *Width* is too large. - *Alignment* is illegal value or too large.
NoExist(109)	Bitmap file assigned by *FileName* could not be found.
Failure(111)	Bitmap data could not be transmitted to POS Printer. It is possible that printer cover is open, it run out of paper, or power of POS Printer is OFF.
Busy(113)	It cannot be executed because of device outputting.

Extended(114)	**ErrorCodeExtended** = Toobig(206):
	Bitmap is too wide to print without conversion, or too large to convert
	ErrorCodeExtended = Badformat(207):
	Assigned file is not bitmap file or not-supported format.

The other	Refer to the item of **ErrorCode**.

SetLogo Method

Syntax

void SetLogo (PrinterLogoLocation *Location*, string *Data*);

Parameter	**Description**
Location	logo to set up, Top(1), or Bottom(2)
Data	Characters to form logo. It consists of printable characters, Escape Sequence, Carriage Return (13 decimal), and New line/Line Feed (10 decimal)

Remarks

It is called when to save data character string as top logo or bottom logo.

Logo is printed by calling **PrintNormal** Method/**PrintImmediate** Method which includes escape sequence of top log/bottom logo within printing data.

Exception

One of the following values is returned and contained in **ErrorCode** Property.

Value	**Meaning**
No Exception	Method ends normally.
Busy(113)	It is not executed because it is outputting.
Illegal(106)	Illegal *Location* is assigned.
The others	Refer to the items of **ErrorCode**.

TransactionPrint Method

Syntax

void TransactionPrint(PrinterStation *Station*, PrinterTransactionControl *Control*);

Parameter	Description
Station	Assign Receipt(2)
Control	Batch processing. Refer to the following values.

Following are *Control* values.

Value	Meani9ng
Transaction(11)	Start of batch processing
Normal(12)	End of batch processing after printing buffered data.

Remarks

This method is called when to enter/leave batch processing.

If *Control* is Transaction(11), it enters batch processing. Calls to the **PrintNormal**, **CutPaper**, **RotatePrint**, **PrintBarCode** and **PrintBitmap** methods after that, buffer printing data by Service Object until **TransactionPrint** is called by setting Normal(12) to *Control* Parameter. (In this case, the above printing data of the above method is only buffered and printing does not start. Also, **AsyncMode** Property value does not affect the operation. In other words, the request does not assign **OutputID** and does not notify **OutputCompleteEvent**. In addition to that, each method succeeds, not affected by error state of POS Printer. For example, even though the power of the POS Printer is OFF, it does not return error at the time of calling of each method, during buffering of printing data by **TransactionPrint.**)

If *Control* is Normal(12), it leaves batch processing. If data is buffered by the **PrintNormal**, **CutPaper**, **RotatePrint**, **PrintBarCode** and **PrintBitmap** methods, the data is to be printed. The whole batch processing is processed as one message. This method is executed synchronously if **AsyncMode** is **FALSE,** and asynchronously if **AsyncMode** is **TRUE**.

Batch processing mode is canceled by calling **ClearOutput.** The buffered printed lines are also deleted.

Be careful when to execute **RotatePrint** method. Until **TransactionPrint** is executed and it leaves batch processing mode, printing by the calling of **RotatePrint** Method with Right90(257), by the calling of **PrintNormal**

Method, and by calling of **RotatePrint** Method with PTR_RP_ Normal (1) is not done. Also, in case of calling of **RotatePrint** Method with Right90(257) and calling of **TransactionPrint** Method with Transaction(11), because buffering by **TransactionPrint** Method is prior to the others, it cannot print the data buffered during this time properly and without rotation printing. Accordingly, if **RotatePrint** Method is executed, it must be done after **TransactionPrint** Method.

If the batch processing mode is started again while the batch processing mode is starting, the print data buffered up to that point will be discarded.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Method ends normally.
Illegal(106)	No assigned POS Printer (except receipt) The page mode function is starting.
NoHardware(107)	POS Printer is OFF or OFFLINE.
Failure(111)	This class library is in error state. After deleting error, execute it.
Busy(113)	It cannot perform because it is outputting. (Only when **AsyncMode** is **FALSE,** it is returned.)
Extended(114)	**ErrorCodeExtended** = CoverOpen(201): POS Printer cover is open. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = RecEmpty(203): It runs out of paper. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Blackmark(10001): Black Mark Error occurs. Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Fatal (10003): Fatal error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.) **ErrorCodeExtended** = Overheat (10006): Head overheat error occurs. (Only when **AsyncMode** is **FALSE,** it is returned.)
The others	Refer to the items of **ErrorCode**.

ValidateData Method

Syntax

void ValidateData(PrinterStation *Station*, string *Data*);

Parameter	Description
Station	Receipt(2) is assigned
Data	Data to be validated. It includes printable data and escape sequence.

Remarks

It is called to validate whether the data sequence which includes one or more-than-one escape sequences **is** valid or not for assigned POS Printer, before calling **PrintNormal** Method and **PrintImmediate** Method.

This method does not do any printing, but is used to validate the POS Printer capability.

The escape sequence that returns Illegal(106) or that is not described in the case returning Failure(111) is not determined and No Exception is always returned.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Data is validated.
Illegal(106)	One or more-than-one escape sequences are out of the range, but Control can select valid alternatives. Or, subject station is not yet supported.
Failure(111)	One or more-than-one escape sequences are not supported. There is no alternative to select.
The others	Refer to the items of **ErrorCode**.

Cases which cause Illegal(106) returned.

Escape Sequence	Condition
Paper Cut/Feed and Paper Cut	'#' (percentage) is not supported. (Valid only when 1 to 100)
Underline	'#' (thickness) is not supported. (Valid only when 1 to 2)
Height Rate	'#' (rate) is not supported. (Valid only when 1 to 8)
Width Rate	'#' (rate) is not supported. (Valid only when 1 to 8)

Cases which cause Failure(111) returned.

Escape Sequence	Condition
Feed and Cut and Stamp Printing	Not supported.
Stamp Printing	Not supported.
Bitmap Printing	'#' (Bitmap number) is out of range. (Valid only when 1 to 20)
Reverse Feed	Not supported.
Custom	Not supported.
Red	Not supported when **CapRec2Color** is **FALSE**.
Shading	Not supported.
Subscript/Superscript	Not supported.

DrawRuled Line Method

Syntax

void DrawRuledLine (PrinterStation *Station*, string *PositionList*, LineDirection *Direction*, int *LineWidth*, LineStyle *LineStyle*, int *LineColor*);

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	This method is not supported

4.7. Event

DirectIOEvent Event

Syntax

DirectIOEventHandler DirectIOEvent;

Parameter	Description
int EventNumber	Event Number. Specific value assigned by Service Object.
int Data	Number value data. Specific value changing according to event number and Service Object.
object Object	A byte string object. The data is not converted and a single Bstring character is stored as a single byte. Acts as BinaryConversion = NIBBLE.

Remarks

For direct transmission to Application, Service Object notifies it.

It becomes possible for Service Object to supply Application with Event not supported in Control Object.

In the case that the data other than normal status notification is received from the Printer, the following *EventNumber* notifies it, byte by byte.

In *EventNumber*, NotAsb(=101) is set and the value of the decimal conversion of byte data (0 to 255) is ser in *pData*.

ErrorEvent Event

Syntax

DeviceErrorEventHandler ErrorEvent;

Parameter	Description
ErrorCode ErrorCode	Code which causes error event. Refer to the items of **ErrorCode** for values.
int ErrorCodeExtended	Extension code which causes error event. Refer to the values below.
ErrorLocus ErrorLocus	Output(1) is set up. Error occurs in asynchronous outputting.
ErrorResponse ErrorResponse	Respond to error event. Refer to the values below.

If **ErrorCode** is Extended(114), **ErrorCodeExtended** is set to one of the following values.

Value	Meaning
CoverOpen(201)	POS Printer cover is open.
RecEmpty(203)	It runs out of paper.
Fatal (10003)	Fatal error occurs in POS Printer.
Overheat (10006)	Head overheat occurred in POS Printer.
Cutterjam(10008)	A cutter jam error occurs.

Location contents assigned by ***ErrorResponse*** are preset to the default value of Retry(11).

Application sets one of the following values.

Value	Meaning
Retry(11)	It tries its asynchronous processing again. It has already left error state.
Clear(12)	It deletes all the buffered data including asynchronous output. (It has same effect as **ClearOutput** Method.) It has already left error state.

Remarks

It will notify when this class library state changes to error state, in executing method performable asynchronously.

OutputCompleteEvent Event

Syntax

OutputCompleteEventHandler OutputCompleteEvent;

OutputID Parameter shows ID number of completed asynchronous-output-request.

Remarks

It notifies when the asynchronous-output-request started before, ends normally.

StatusUpdateEvent Event

Syntax

StatusUpdateEventHandler StatusUpdateEvent;

Remarks

Status is set one of the following values.

Value	Meaning
CoverOpen(11)	POS Printer cover is open.
CoverOK(12)	POS Printer cover is closed.
RecEmpty(24)	No receipt paper.
RecNearempty(25)	Receipt paper is low.
RecPaperOK(26)	Receipt paper is ready.
RecCoverOpen(62)	The cutter jam error occurred.
RecCoverOK(63)	The cutter jam error is recoverd.
Idle(1001)	All the asynchronous output succeeds, or ends by being deleted. POS Printer's **State** is Idle(2) now. **FlagWhenIdle** Property must be **TRUE** in order to be notified by this event. And POS Printer Control automatically sets the property to **FALSE** before the event notifies it.
PowerOnline (2001)	Device is Power-ON and ready. (It notifies at the time of **PowerNotify** = Enabled(1))
PowerOffOffline(2004)	Device is Power-OFF or in OFF-LINE state. (It notifies at the time of **PowerNotify** = Enabled(1))
Progress(2100) + 1 to 100	(1 to 100 indicate the completion percentage) Specifies the completion percentage of the

	firmware.
Complete(2200)	The firmware is updated successfully.
FailedDevOK(2201)	The update firmware process failed but the device is still operational.

Remarks

It is notified when significant state change occurred on printer device side. When device starts to enable, Control issues the first **StatusUpdateEvent** to let application know the device state.

Reference

CapPowerReporting Property and **PowerNotify** Property

5. OPOS Interface Specifications (Drawer)

5.1. List

List of Properties

Common	Type	Access	May Use After	Initial Value, Condition
CapCompareFirmwareVersion	bool	R	Open	FALSE
CapPowerReporting	PowerReporting	R	Open	None(0)
CapStatisticsReporting	bool	R	Open	FALSE
CapUpdateStatistics	bool	R	Open	FALSE
CheckHealthText	sring	R	Open	""
Claimed	bool	R	Open	FALSE
DeviceEnabled	bool	R/W	Open & claim	FALSE Made writable after Open and claim.
FreezeEvents	bool	R/WOpen	Open	FALSE Made writable after Open.
PowerNotify	PowerNotification	R/W	Open	Disabled(0) Unwritable
PowerState	PowerState	R	Open	Unknown(2000)
State	ControlState	R	---	1
ControObjectDescription	sring	R	---	"FP CashDrawer Control Object (C) Fujitsu Isotec"
ControlObjectVersion	Version	R	---	1013XXX
ServiceObjectDescription	sting	R	Open	"Fujitsu Isotec FP CashDrawer Service Object"
ServiceObjectVersion	int	R	Open	1013XXX
DeviceDescription	sring	R	Open	"FP CashDrawer (C)20xx Fujitsu Isotec"
DeviceName	sring	R	Open	The name set in the parameter at Open.

Specific	Type	Access	May Use After	Initial Value, Condition
CapStatus	bool	R	Open	The initial value is the value of the PosExplorer "CapStatus."
CapStatusMultiDrawerDetect	bool	R	Open	FALSE
DrawerOpened	bool	R	Open & Enable	FALSE

* In the Access column, R indicates Read-Only, R/W indicates Read/Write. The item in May Use After is the method and property required for initialization, Open indicates the Open method, Claim indicates the Claim method and Enable indicates setting the DeviceEnabled property to TRUE. If required procedure is not executed, the error may be set in the ErrorCode property. When May Use After is Open & Claim or Open, Claim & Enable, the property is available for acquisition after the Open method is executed, but the value may not be initialized until all Open, Claim & Enable are executed. To acquire such property, access it after the conditions are met.

List of Methods

Common	Initialization
Open	---
Close	Open
Claim	Open
Release	Open & Claim
CheckHealth	Open & Enable
CompareFirmwareVersion	Open & Enable
DirectIO	Open
ResetStatistics	Open & Enable
RetrieveStatistics	Open & Enable
UpdateFirmware	Open & Enable
UpdateStatistics	Open & Enable

Specific	Initialization
OpenDrawer	Open & Enable
WaitForDrawerClose	Open & Enable

List of Events

Event	Initialization
DirectIOEvent	Open & Enable
StatusUpdateEvent	Open & Enable

5.2. Common Properties

The following sections describe the properties provided commonly to the Drawer. We have two kinds of properties: Read-Only and Read/Write. If a property is for writing, R/W will be shown at the side of each property.
Only if return value has a special meaning, it will be shown. As for error, in case that access is done without satisfying initializing condition, refer to ErrorCode property.

CapCompareFirmwareVersion Property

Syntax

bool CapCompareFirmwareVersion;

Remarks

If **TRUE**, then the Service/device supports comparing the version of the firmware in the physical device against that of a firmware file.
This property is initialized to **TRUE** by the **Open** method

CapPowerReporting Property

Syntax

PowerReporting CapPowerReporting;

Remarks

It identifies power reporting ability. The value to show power reporting ability is just the following.

Value	Meaning
None(0)	The power reporting ability does not work.
Standard(1)	SO can judge two kinds of power states and can report it. (Online and OffOffline)
Advanced(2)	The SO can determine and notify all three power states: Off, OffLine, and OnLine.

This property is initialized by the **Open** method.

CapStatisticsReporting Property

Syntax

bool CapStatisticsReporting;

Remarks

This property is initialized to **FALSE** by the **Open** method. Statistics reporting is not supported.

CapUpdateStatistics Property

Syntax

bool CapUpdateStatistics;

Remarks

This property is initialized to **FALSE** by the **Open** method. Statistics reporting is not supported.

CheckHealthText Property

Syntax

string CheckHealthText;

Remarks

It keeps the **CheckHealth** Method result called just before. The following are examples of the check.

- "Internal HCheck: Successful" (It succeeded in Internal check.)
- "External HCheck: Successful" (It succeeded in External check.)
- "External HCheck: Failure" (It failed in External check.)
- "InteractiveHCheck: Not Supported" (It is not supported.)

This value cannot be initialized before the first **CheckHealth** Method calling. (null character)

Claimed Property

Syntax

bool Claimed;

Remarks

TRUE: Exclusive Access right of device is acquired.

FALSE: Device is released in order to be shared to other applications.

In many cases, access event to Methods/property is possible after acquiring exclusive access right.

Claimed Property value is initialized to **FALSE** by **Open** Method.

DeviceDescription Property

Syntax

string DeviceDescription;

Remarks

"FP Cash Drawer (C) 20xx- Fujitsu Isotec." is set.

This property is a character string identifying the device, and holds the device and any pertinent information about it.

This property is initialized by the **Open** method.

DeviceEnabled Property R/W

Syntax

bool DeviceEnabled;

Remarks

TRUE:

The device is made to enable (Operation state). If converted to **TRUE,** it is made to enable.

FALSE:

The device is made to disable. If converted to **FALSE**, if possible, it is made to disable physically.

Inputting after that is canceled and output operation is not possible.

Before this device is used, application must set this property **TRUE.**

This property is initialized to **FALSE** by **Open** Method.

Exception

When this property is set, the following value is placed in the **ErrorCode** property:.

Value	Meaning
No Exception	It succeeds in Property setting.
Failure (111)	It failed to access to device. Because it failed to open connection port, make sure whether any connection port is used form other program.
NoHardware(107)	POS Printer is OFF or OFFLINE or the cable is not connected. Clear the problem, and then execute the property again. (* In the case of USB interface, even if a POS printer is connected, this error occurs when a serial number set by registry is different from a serial number set by a POS printer.) *In case of LAN interface, if the POS Printer

is not connected, No Exception is returned, not this error.

DeviceName Property

Syntax

string DeviceName;

Remarks

One of "FP2x00SERDR1","FP2x00SERDR2", "FP2x00USBDR1","FP2x00USBDR2", "FP2x00LANDR1_xxx.xxx.xxx.xxx", "FP2x00LANDR2_xxx.xxx.xxx.xxx" (xxx.xxx.xxx.xxx: IP address) is set.

This property shows the device and the any pertinent information about it.

This is a short version of **DeviceDescription** and should be limited to 30 characters.

This property is initialized by the **Open** method.

FreezeEvents Property R/W

Syntax

bool FreezeEvents;

Remarks

In case of **TRUE**, Event is not reported from Control.

Event is kept by Control till the freeze is terminated.

In case of **FALSE**, Event is reported from Control. If there is any Event kept during freeze, that Event is reported when **FreezeEvents** are converted to **FALSE**.

If interrupt by Event is undesirable, Application can select Event freeze.

This property is initialized to **FALSE** by **Open** Method.

PowerNotify Property R/W

Syntax

PowerNotification PowerNotify;

Remarks

This is the Power Notify Function type set by Application.

The values to show Power Notify Function are just the following.

Value	**Meaning**
Disabled(0)	Control does not give any power notification to Application. **StatusUpdateEvent** concerning Power Notification is not given and any setting is not done to **PowerState** Property. (Default Value)

This property is initialized to Disabled(0) by **Open** Method.

Exception

When you call this property, you may throw the following **PosControlException**:.

Value	**Meaning**
Illegal(106)	It cannot set this property.

PowerState Property

Syntax

PowerState PowerState;

Remarks

If it can be judged, Present Device Power State is set up. The value to show power state is just the following.

Value	**Meaning**
Unknown(2000)	It cannot judge the device power state.
Online(2001)	
Off(2002)	
Offline(2003)	
OffOffline(2004)	

ServiceObjectDescription Property

Syntax

string ServiceObjectDescription;

Remarks

Character string which shows Service Object which supports device and the device manufacturer

"Fujitsu Isotec FP Cash Drawer Service Object" is set.

This property is initialized by **Open** Method

ServiceObjectVersion Property

Syntax

System.Version ServiceObjectVersion;

Remarks

"1013XXX" is set. Holds the Service Object version number. (XXX varies depending on the time the Control Object is distributed.) This property is initialized by the **Open** method.

State Property

Syntax

ControlState State;

Remarks

It shows the present state of Control.

Value	Meaning
Closed(1)	Control is closed. (Default)
Idle(2)	Control is in normal state and not busy.
Busy(3)	The control is busy because it is in a normal state and performing output.
Error(4)	An error is reported and the application must return control to a normal state before normal I/O can resume.

This property is always readable.

5.3. Common Methods

CheckHealth Method

Syntax

string CheckHealth (HealthCheckLevel *Level*);

Level parameter shows the type of health check executed with device. The following values can be assigned.

Value	Meaning
Internal(1)	Health Check without using physical device is done. It always returns Sucess.
External(2)	It does perfect test using device. If possible, it opens the drawer. If drawer is open successfully, Sucess is returned. This method fails when exclusive access is done by other application.
Interactive(3)	It perform dialogue test. Not supported.

Remarks

It is called at the time of device condition test. The result of this method is contained in **CheckHealthText** Property. **CheckHealth** Method is always synchronous.

Exception

When you call this property, you may throw the following

PosControlException:.

Value	Meaning
No Exception	It shows that Health Check Procedure starts properly and that, when confirmed that, the device is working properly. However, you cannot decide about its properness until you see the result of the test.
Claimed(102)	Other device is doing exclusive access.
Illegal(106)	*Level* parameter not supported is assigned.
NoHardware(107)	It failed in health check procedure. The POS Printer connected to the drawer is OFF or OFFLINE. It is only contained in the case of External(2) setting.
Timeout(112)	Connection is succeeded with printer connected to Drawer, but Drawer Open cannot be detected after timeout period. Only when External(2) is set and CapStatus proper is set TRUE, it will be stored.
The others	Refer to the items of **ErrorCode**.

Claim Method

Syntax

int Claim (int *Timeout*);

Timeout Parameter shows maximum waiting time (in ms.) till it acquires exclusive access right.

In case of zero, even if it cannot acquire the Device Exclusive Access Method it returns the result immediately.

If **Forever (-1)** is set, Method waits as long as till it can acquire exclusive access right.

Remarks

This method is called when exclusive access is required to device.

It is not requisite to acquire exclusive access right because drawer device is sharable device.

In case of success, **Claimed** Property is set to **TRUE**.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Exclusive access right is confirmed and proccessable device connection is assured. **Claimed** Property is **TRUE**. If this application has already accessed the device exclusively, it is returned, too.
Illegal(106)	Unavailable ***Timeout*** Parameter is assigned.
Timeout(112)	Other application tries to access the device exclusively and waits for being released. But ***Timeout*** Time (in ms.) has passed. Or, POS Printer Device does not become POS-Printer-Device-Process-Possible State even after ***Timeout*** Time (in milli-sec.) passes.
The others	Refer to the items of **ErrorCode.**

Close Method

Syntax

void Close ();

Remarks

It is called when to release Device and its resource.

If **DeviceEnabled** Property is **TRUE,** Device is forced to disable.

If **Claimed** Property is **TRUE,** at first, exclusive access will be released.

Don't execute at the time of Event Processing. (within Event Handler)

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	Device is disabled and closed.
The others	Refer to items of **ErrorCode**.

DirectIO Method

Syntax

DirectIOData DirectIO (int *command*, int *data*, object *obj*);

Remarks

This method is called to communicate with the Service Object directly.

This method is not supported.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	This method is unavailable.
Other Values	Refer to the ErrorCode property.

Open Method

Syntax

void Open ();

Remarks

It is called to open the device.
When the **Open** method is successful, the Common properties and other Specific-to-Class Properties are initialized.

Exception

When you call this method, you may throw the following

PosControlException:.

Value	Meaning
Illegal(106)	The applicable Control has already opened.

Release Method

Syntax

void Release ();

Remarks

Call this device when to release exclusive access of Device.
If **DeviceEnabled** Property is **TRUE** and exclusive device, Device is made to disable. Don't execute at the time of Event Processing. (within Event Handler)

Exception

When you call this method, you may throw the following

PosControlException:.

Value	Meaning
No Exception	Exclusive Access is released. **Claimed** Property becomes **FALSE.**
Illegal(106)	Application doesn't have exclusive access right to applicable Device.
The others	Refer to the explanation of **ErrorCode**.

ResetStatistics Method

Syntax

void ResetStatistics ();

Remarks

This method is not supported.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	**Meaning**
Illegal(106)	This method is not supported.

RetrieveStatistics Method

Syntax

string RetrieveStatistics ();

Remarks

This method is not supported.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	**Meaning**
Illegal(106)	This method is not supported.

Syntax

string RetrieveStatistics (StatisticCategories *statistics*);

Remarks

This method is not supported.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	**Meaning**
Illegal(106)	This method is not supported.

Syntax

string RetrieveStatistics (string[] *statistics*);

Remarks

This method is not supported.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	**Meaning**
Illegal(106)	This method is not supported.

UpdateFirmwareMethod

Syntax

void UpdateFirmware (string *FirmwareFileName*);

Remarks

This method is not supported.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	This method is not supported.

UpdateStatistics Method

Syntax

void UpdateStatistics (Statistic[] *statistics*);

Remarks

This method is not supported.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	This method is not supported.

Syntax

void UpdateStatistics (StatisticCategories *statistics,* object *value*);

Remarks

This method is not supported.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	This method is not supported.

5.4. Specific Property

CapStatus Property

Syntax

bool CapStatus;

Remarks

TRUE: It can notify open/close state of Drawer.

FALSE: It **cannot** notify open/close state of Drawer.

This property is initialized by **Open** Method.

The property for the first drawer defined by "FPxxx[Device name]DR1" is set to **TRUE** and the property for the second drawer defined by "FPxxx[Device name]DR2" is set to **FALSE.** Each device name includes"SER" (Serial Interface), "USB" (USB Interface) or "LAN" (LAN Interface, IP address xxx.xxx.xxx.xxx is added to Device name.).

* Unless the **CapStatus** property is **TRUE** and the Printer class library is enabled (**DeviceEnabled**=TRUE) for the printer connected to the Drawer, Status notification of the Drawer is not supported by this property.

* Use the second drawer property defined by "FPxxx[Device Name]DR2" with FALSE.

The 2nd drawer does not support the status notification of drawer open/close.

CapStatusMultiDrawerDetect Property

Syntax

bool CapStatusMultiDrawerDetect;

Remarks

FALSE: All the drawers are closed or one or more than one drawers are open. It does not mean that it can notify the open/close state of any one drawer among multiple-drawer-configuration.

This property is initialized by **Open** Method.

DrawerOpend Property

Syntax

bool DrawerOpened;

Remarks

TRUE: Drawer is open.

FALSE: Drawer is closed.

When **CapStatus** Property is FALSE, Device cannot notify state change and this **DrawerOpened** Property is always FALSE.

When device is made to enable, this property is initialized to appropriate value.

* Unless the **CapStatus** property is **TRUE** and the Printer class library is enabled (**DeviceEnabled**=TRUE) for the printer connected to the Drawer, Status notification of the Drawer is not supported by this property.

*The second drawer defined by "FPxxx[Device Name]DR2" cannot know open/close state, so it is always **FALSE.** Each device name includes "SER" (Serial Interface) or "USB" (USB Interface).

5.5. Specific Method

OpenDrawer Method

Syntax

void OpenDrawer ()

Remarks

Open the drawer. This method fails if exclusive access is being made by another application.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
No Exception	It ends normally.
Claimed(102)	Other device is accessing exclusively.
Failure(111)	It could not transmit to device.
Timeout(112)	Connection is succeeded with printer connected to Drawer, but Drawer Open cannot be detected after timeout period. When CapStatus proper is set TRUE, it will be stored.
The others	Refer to the items of **ErrorCode**.

WaitForDrawerClose Method

Syntax

void WaitForDrawerClose (int *BeepTimeout*, int *BeepFrequency*, int *BeepDuration*, int *BeepDelay*);

Remarks

This method is not supported.

Exception

When you call this method, you may throw the following **PosControlException**:.

Value	Meaning
Illegal(106)	This method is not supported.

5.6. Event

DirectIOEvent Event

Syntax

DirectIOEventHandler DirectIOEvent;

Remarks

This method is not notified.

StatusUpdateEvent Event

Syntax

StatusUpdateEventHandler StatusUpdateEvent;

Remarks

The latest drawer state is set in *Status* Parameter.

Value	Meaning
Drawerclosed (0)	Drawer is closed.
Draweropen(1)	Drawer is open.

Remarks

It is notified when drawer open/close state changes.

* Unless the **CapStatus** property is **TRUE**, Status notification of the Drawer is not supported by this event.

6. About PosExplorer External Settings

The external setting values used in this class library are shown below.

You can set the PosExplorer settings manually, but use the settings program. Also, the settings are loaded when this class library executes the Open method, so changing the values during this class libary operation does not affect the operation. To implement the new settings, issue the Close method to this class library and then the Open method.

7. Log Files

The FP POSPrinter or FP CashDrawer class library has the function to output the log files by setting the **LogFolder**, **LogFileName**, **LogLevel** properties in the PosExplorer. The behavior of log file output is as follows:

1. Create a log file according to the following naming rule:

   LogFolder = "C:\OPOS\FIT\FP_NET\Log\"
   For **LogFileName = "POSPrinter"**, when the log file is set as **"C:OPOS\FIT\FP_NET\Log\POSPrinter_[HardwarePath]_yyyymmdd.log"** and the date when the POS is executed is January 30, 2021, the file name of the log file is created as **"C:OPOS\FIT\FP_NET\Log\POSPrinter_[HardwarePath]_20210130.log"**.
   * CashDrawer is the same.

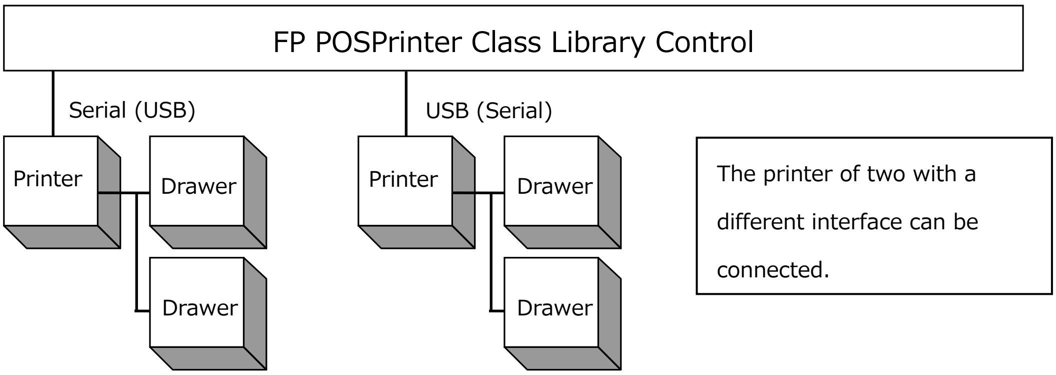
2. If the file created by the naming rule in the step 1 already exists and current month and the month when the file is updated last are different, then the existing log file is deleted. In other case, the file is created new or additionally written.

3. As the results, the log files of the last one-month are pooled on the POS (PC). Upper bound of the file size for each log file is not limited. If output includes the detailed logs, take special care to prevent shortage of the disk space.

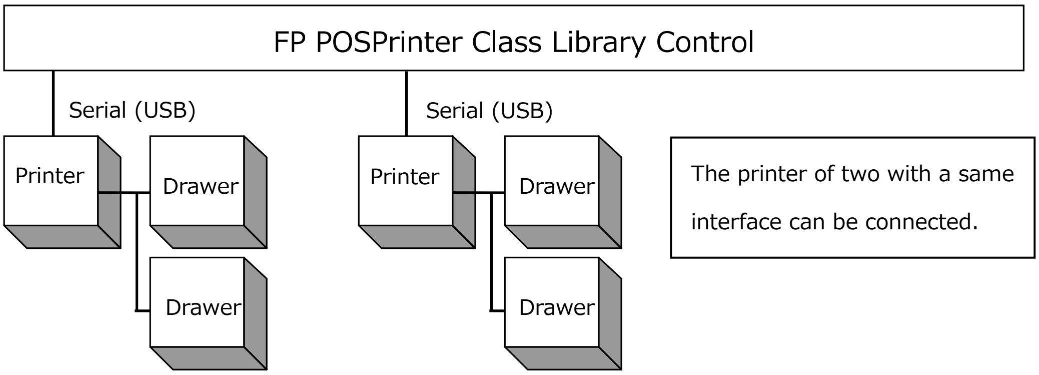
4. Based on the **LogLevel** setting, output the following log file.
   LogLevel = 0: Outputs no log.
   LogLevel = 1: Outputs normal trace log, warning and error log.
   LogLevel = 2: Outputs only error log.

8. Using Multiple Printers

The FP POSPrinter Class library control can use two of a serial and USB simultaneously.

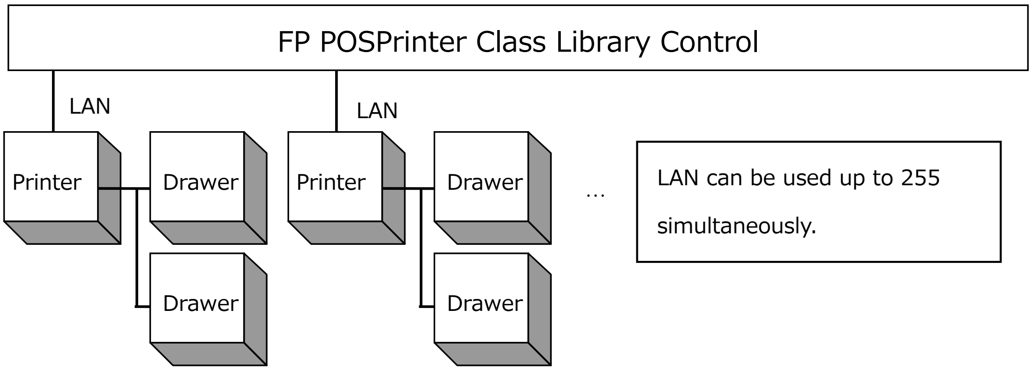


The printer of two with a different interface can be connected.



The printer of two with a same interface can be connected.

The FP POSPrinter Class library control can use up to 255 of LAN simultaneously.



LAN can be used up to 255 simultaneously.

9. Page Mode Function

9.1. About exclusive relationships between methods

The following table shows the exclusive relationship between the PageModePrint method and the TransactionPrint and RotatePrint methods that have the same buffering function.
When “Later” is executed after “Starter” is executed,
"○" : ErrorCode = No Exception
"×" : ErrorCode = Illegal
"△" : ErrorCode = No Exception (buffering data is discarded).

			Later								
			PageModePrint			RotatePrint				Trasaction Print	
			PageMode	Normal	Cancel	Right90	Left90	Rotate180	Normal	Transaction	Normal
Stater	Not specified		○	×	×	○	○	○	○	○	○
	PageModePrint	PageMode	△	○	○	×	×	×	×	×	×
		Normal	○	×	×	○	○	○	○	○	○
		Cancel	○	×	×	○	○	○	○	○	○
	RotatePrint	Right90	×	×	×	×	×	×	○	×	×
		Left90	×	×	×	×	×	×	○	×	×
		Rotate180	×	×	×	×	×	×	○	×	×
		Normal	○	×	×	○	○	○	○	○	○
	TransactionPrint	Transaction	×	×	×	○	○	○	○	△	○
		Normal	○	×	×	○	○	○	○	○	○

(Example) If TransactionPrint (Transaction) is executed after executing
PageModePrint (PageMode), ErrorCode "Illegal" will be returned.

9.2. Print data development position in page mode

In absolute positioning during page mode start, strings, barcodes, and bitmaps(PrintBitmap) are printed below the baseline.

Revision History

Document	FP-2000 Series Printer POS Printer, Cash Drawer Application Programmer's Guide of OPOS.NET Class Library for Serial/ USB/ LAN Interface

Revision	Date	Part/Season/Contents	Revised by
1.0	2021/07/30	Newly created.	FUJITSU ISOTEC LIMITED.