

EPSON OPOS ADK MANUAL

APPLICATION DEVELOPMENT GUIDE

GENERAL DEVELOPMENT

Version 3.00 Sep. 2024

Notes

- (1) Reproduction of any part of this documentation by any means is prohibited.
- (2) The contents of this documentation are subject to change without notice.
- (3) Comments and notification of any mistakes in this documentation are gratefully accepted.
- (4) This software cannot be used with other equipment that the specified.
- (5) EPSON will not be responsible for any consequences resulting from the use of any information in this documentation.

Trademarks

Microsoft®, Windows®, Windows Server®, Visual Basic® and Visual C++® are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

EPSON® and ESC/POS® are registered trademarks of Seiko Epson Corporation.

Other product and company names used herein are for identification purposes only and may be trademarks or registered trademarks of their respective companies.

Copyright © 2000 Seiko Epson Corporation

Contents

Section 1. Introduction.....	1
Section 2. General Development	2
2.1 Who Should Read This Manual?	2
2.2 OPOS Software System Hierarchy	2
2.3 Checking Connections of Devices	3
2.4 Using Extended Ports	3
2.5 Creating a LOG File	4
2.6 Logical Device Name	5
2.7 ClaimDevice and ReleaseDevice methods	7
2.8 UPOS Reference	7
Section 3. Usage.....	8
3.1 Using OPOS API.....	8
3.2 Using Constants.....	11
3.3 API Function Comparison/Pasting.....	12
Section 4. OPOS API Programming	13
4.1 Things to Consider when Programming	13
4.2 Code Explanations	13
4.3 Coding Samples.....	14
Section 5. Class Specific Information	19
Section 6. Error Information.....	20
6.1 When Executing Common Properties.....	20
6.2 When Executing Common Methods	21
Section 7. Warnings	28
7.1 Cautions for Programming.....	28
7.2 Precautions for VC++.....	29

Section 1. Introduction

This manual is an application development guide describing warnings and elements related to the programming in connection with development of applications using the API functions supported by OPOS.

Descriptions in this manual are intended for the following programming languages.

Microsoft Visual Basic 2019 or higher

Microsoft Visual C++ 2019 or higher

Please read this guide before you start programming and use it as help during programming. For explanation on the various devices, please refer to the manuals for each device. The programming examples given in the manual for each device are intended for Visual Basic.

For information on installing and setting up the EPSON OPOS ADK, please refer to the "EPSON OPOS ADK User's Manual (SetupPOS/ TMUSB)". For detailed explanations of API functions, please refer to the "UPOS 1.14" created by the OPOS Committee.

Section 2. General Development

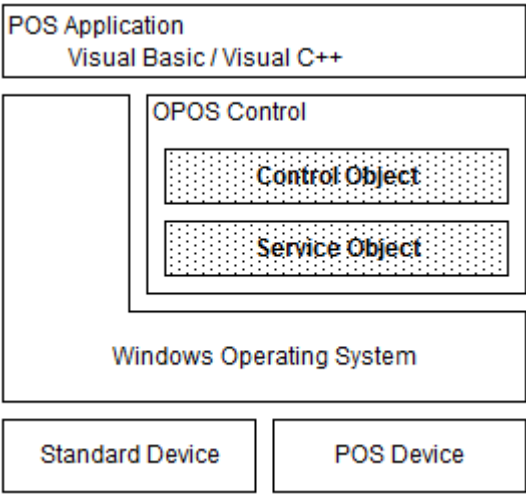
This section contains useful information ranging from setting of the OPOS software to developing, and other items that should be read and understood before attempting to develop an OPOS application.

2.1 Who Should Read This Manual?

Please refer to the “WHO SHOULD READ THIS DOCUMENT” section of UPOS 1.14.

2.2 OPOS Software System Hierarchy

OPOS contains two hierarchical objects called the Control Object and the Service Object. Control Object may be written as CO, and Service Object may be written as SO from here on.



SYSTEM HIERARCHY

2.2.1 Control Object

The POS Printer, MICR, Line Display, Cash Drawer, etc., exist as objects in the system, listed by device class (category), and are stored as ActiveX controls that Visual Basic and Visual C++ can use.

Consult the UPOS for a control's application interface.

EPSON uses CommonCO(CCO) provided by the OPOS Technical Committee.

Provided that the COs abide to the CPG, even the COs of other manufacturers can be used together with EPSON OPOS ADK.

2.2.2 Service Object

Service Objects exist as objects in the system, listed by device types. They encompass the dependent parts of the devices. They are used by the higher-ranking Control Objects. EPSON SOs are created dependent upon the devices supported by EPSON OPOS ADK.

2.3 Checking Connections of Devices

Before starting OPOS programming, please check that each of the device categories is connected correctly. Using the SetupPOS utility's interactive CheckHealth function makes it a simple task to check the connections. For details, please see the "EPSON OPOS ADK User's Manual (SetupPOS/ TMUSB)".

2.4 Using Extended Ports

EPSON OPOS ADK is able to support the use of RS-232C expansion cards with genuine Windows RS-232C drivers. This allows the use of ports up to COM 10. With genuine expansion LPT cards, it is also able to support the use of LPT expansion ports up to LPT3. When using expansion cards, please follow the card's product manual to configure all settings.

When using extended ports, use the SetupPOS utility to set the maximum values of the COM and LPT ports correctly.

2.5 Creating a LOG File

EPSON OPOS ADK contains tracing functions to support the creation of error free programs. Tracing functions save information on the running of OPOS API in a special file. If a problem occurs during the development of a program, this file can be used to help troubleshoot the problem.

Below is an explanation of trace file usage.

[Setting Trace Functions]

- 1) Using the SetupPOS utility, enable the tracing functions. The following two trace types are available.

Global trace

Private trace

Select either of the functions in accordance with the purpose.

- 2) Specify the desired trace log file name.

[Trace Modes]

The following three trace modes are available.

No Trace: Tracing functions are not performed.

Global Trace: Trace of all global OPOS API functions. Items set as Private are ignored.

Private Trace: Trace of the device specified in [DeviceName]. All Global Trace settings will be canceled. It is possible to specify a Private Trace on multiple devices.

[Viewing the Trace Log File]

The Log file is set up in the following style:

Property Reference: TimeStamp DeviceClass [DeviceName].PropName:
 {Property value}

Property Settings: TimeStamp DeviceClass [DeviceName].PropName
 ={Property value}

Calling of a Method: > TimeStamp DeviceClass [DeviceName].
 MethodName(Parameter list)

Completion of a Method: < TimeStamp DeviceClass [DeviceName].
 MethodName(Parameter list)

(This line becomes valid only when an address is passed to the parameter, and after completion, it is possible to view the parameters themselves.)

Method Results: < TimeStamp DeviceClass [DeviceName].

```

        ResultCode:{ResultCode}
    < TimeStamp DeviceClass[DeviceName].
    ResultCodeExtended:{ResultCodeExtended}

```

Below is an example of a trace file that has been created after tracing the actions of a POS printer (CO registered).

```

>POSPrinter1[TM-T88IV].Open("TM-T88IV")
<POSPrinter1[TM-T88IV].ResultCode:OPOS_SUCCESS
<POSPrinter1[TM-T88IV].ResultCodeExtended:0 (0x00000000)
POSPrinter1[TM-T88IV].ServiceObjectVersion:1013001 (0x000f7509)
POSPrinter1[TM-T88IV].ServiceObjectDescription:"EPSON POSPrinter OPOS
        Service Object Copyright (C) 2005-2011 Seiko Epson Corp."
>POSPrinter1[TM-T88IV].Close()
<POSPrinter1[TM-T88IV].ResultCode:OPOS_SUCCESS
<POSPrinter1[TM-T88IV].ResultCodeExtended:0 (0x00000000)

```

Be aware that when using tracing functions, performance of OPOS slows in comparison to when tracing is not performed. While developing, only enable the tracing functions when required. At other times, disable the tracing function.

[Restriction of the Trace Log File]

The trace log file size can be up to 1 MB. When it exceeds the limit, the file name is automatically changed and saved as (XXXOld.txt). However, it is possible to maintain the file only up to 2 MB or less after this because of this repetition. Please back up the file to an appropriate location when the message box appears telling you the file is full.

[Making the Log File by Multiprocessing]

The log file may divide into the plural when OPOS is used by multiprocessing. At this time, the number such as "1" or "2" is added at the end of the file name. Be aware to the referred file when operating by multiprocessing. It is likely to become similar when the application operates while executing SetupPOS.

2.6 Logical Device Name

A Device Name or Logical Device Name should be specified in the DeviceName.

- 1) A Device Name is a name key for a device that is registered. (Ex.:TM-T88V)
It can be confirmed by the SetupPOS utility.

- 2) The Logical Device Name is a specific character string created in relation to the device name. It can be set and correlated with the device name by the SetupPOS utility.

“TM-T88V” <---> “OPOS.Ptr.1”

By using a Logical Device Name, a program can be made more flexible.

For example, if the Visual Basic code opens devices with the following source,

```
Rc = OPOSPOSPrinter1.Open("OPOS.Ptr.1")
```

the Logical Device Names take the places of the actual device names as set in the setup program.

“OPOS.Ptr.1” <---> “TM-T88IV”

In case the device is changed, this can be accomplished by just changing the setup relations as the actual device can be selected and used freely without changing the entire program's source code.

2.7 ClaimDevice and ReleaseDevice methods

By using the Claim (ClaimDevice method), the exclusive access right to a claimed device can be acquired. By executing the Release (ReleaseDevice method), the exclusive access right to a claimed device can be released. It is possible to claim a device many times. However, even if the device is claimed many times just one call of the ReleaseDevice method will release the exclusive access rights to the device.

2.8 UPOS Reference

When using Visual Basic, please follow the conventions listed below.

Where the UPOS shows:

Syntax LONG CheckHealth(LONG Level);

The Visual Basic usage is:

Syntax CheckHealth(ByVal Level As Long)

Type Long

Below is list of the difference between various types in Visual C++ and Visual Basic.

<i>C</i>	<i>Basic</i>
BOOL	Boolean
BSTR	String (ByVal)
BSTR *	String
LONG	Long (ByVal)
LONG *	Long

Section 3. Usage

This section describes the procedures for developing in Visual Basic/Visual C++.

3.1 Using OPOS API

3.1.1 Using Visual Basic

There are several ways of using OPOS API in Visual Basic.

This manual describes one of the methods that OPOS API is used in the Visual Basic step in which components are added by pasting control objects to a form.

- 1) Please install EPSON OPOS ADK as outlined in the "EPSON OPOS ADK User's Manual (SetupPOS/ TMUSB)". Then set up each device using the SetupPOS utility.
- 2) Microsoft Visual Basic 2019 or higher is necessary to utilize OPOS API. Please confirm that this version is installed on the machine that is to be used.
- 3) After starting Microsoft Visual Basic, add the ActiveX controls to be used into the project by using the Custom Controls option.

[Common CO Version1.14.1]

OPOS CashDrawer Control 1.14.1	:	Control object that can utilize the Cash Drawer API
OPOS POSPrinter Control 1.14.1	:	Control object that can utilize the POSPrinter API.
OPOS MICR Control 1.14.1	:	Control object that can utilize the MICR API.
OPOS CheckScanner Control 1.14.1	:	Control object that can utilize the CheckScanner API.
OPOS LineDisplay Control 1.14.1	:	Control object that can utilize the LineDisplay API
OPOS ElectronicJournal Control 1.14.1	:	Control object that can utilize the ElectronicJournal API

It is also possible to use other CCOs than those of CCO Version1.14.1. Users that want to use other CCO, should first register the CCOs and then select the appropriate CCO by inserting the component.

- 4) After adding the necessary ActiveX controls into your project, the controls will be available from the toolbox. The API functions supported by OPOS become usable by choosing the control from the toolbox and placing them on the Form of your project.
- 5) An OPOS application using various API functions is now ready to be created.

When you create multiple forms and set them up to access a single ADK object in a Visual Basic project, the following problems occur when properties are accessed or methods are called:

- * The method or property accessed first does not return until all processing has been completed. (The project hangs.)
- * The Close() method causes a Visual Basic error when called.

Please make sure that all methods and properties accessed by one form are completely finished before any other form accesses a method or property.

If you want to get an Event in the midst of a procedure, use DoEvents. For example, for a cash drawer, which you programmed to execute WaitForDrawerClose immediately after executing the OpenDrawer method, an Event never occurs until control returns to the application. Putting a DoEvents instruction between the OpenDrawer method and WaitForDrawerClose method executes StatusUpdateEvent and enables you to get an Event.

3.1.2 Using Visual C++

There are several ways of using OPOS API in Visual C++, but this manual describes the method in which OPOS API is used by creating dialog box type applications in Visual C++.

- 1) Please install EPSON OPOS ADK as outlined in the "EPSON OPOS ADK User's Manual (SetupPOS/ TMUSB)". Then set up each device using the SetupPOS utility.
- 2) Microsoft Visual C++ 2019 or higher is necessary to utilize OPOS API. Please confirm that this version is installed on the machine that is to be used.
- 3) Start Microsoft Visual C++.
- 4) Choose "New" from the "File" menu, and then choose MFC AppWizard(exe) from the "Project" tab. Enter the project name. (Example: test)
- 5) In step 1 of MFC AppWizard, set the application type as "Dialog base" before proceeding.
- 6) In step 2 of MFC AppWizard, specify "ActiveX controls" before proceeding.
- 7) In step 3 of MFC AppWizard, change the MFC library link to "Use static

library” before proceeding. (It is not necessary to make this change when the MFC library is used as a share.)

- 8) In step 4 of MFC AppWizard, confirm the creation class and then close AppWizard.
- 9) Display “Resource View” and select the main dialog (for example: IDD_TEST_DIALOG).
- 10) By right-clicking the dialog a pop-up menu appears. Select insertion of ActiveX controls, and select the control that you want to install. (Example: EPSON OPOS LineDisplay ActiveX Control)
- 11) After installing the required devices, display the properties of the installed controls and set the ID or confirm the default ID.
- 12) Start ClassWizard and display the member variable screen. Align the cursor with the control ID for the installed device and select “Add variable”.
- 13) Enter the name of the variable (for example: m_Dispatch) and click “OK” to close the ClassWizard. From now on, this member variable name is usable, and the property method can be used. (For example: m_Dispatch.Open(“Unit1”);)
- 14)

To use an already defined constant in the source code, the previously installed header file must be included. (For example: #include “oposdisp.h”)

3.1.3 Updating ActiveX Control

There are several ways of updating ActiveX control in Visual C++, but this manual describes the method in which OPOS API is used by dialog box type applications in Visual C++.

- 1) Please install new version of EPSON OPOS ADK as outlined in the "EPSON OPOS ADK User's Manual (SetupPOS/ TMUSB)".
- 2) Start Microsoft Visual C++, and open project of application, which uses EPSON OPOS ADK.
- 3) Delete old ActiveX control, and remove files that define old wrapper class from the project. Record the file names and removed control ID. (Example: default file name - coptr.h and coptr.cpp, default control ID - IDC_POSPRINTER1)
- 4) Remove these files from hard drive, also.
- 5) Choose “Add to Project” from the “Project” menu, and then choose “Components and Controls” menu, Select insertion of ActiveX controls from the “Registered ActiveX Controls” folder. (Example: EPSON OPOS Line Display ActiveX Control)

- 6) When select control, the dialog box appears. Make wrapper class to call ActiveX control by dialog. Set the "Class Name" and "File Name" same as removed variable's "Type" and removed file names. (Distinguish a small letter and a capital letter.)
- 7) When add control, control appears in control box. Insert control on dialog. And set the "Control ID" same as removed variable's "Control ID". (Distinguish a small letter and a capital letter.)
- 8) Start ClassWizard and display the member variable screen. Align the cursor with the control ID for the installed device. (Example: IDC_POSPRINTER1)
- 9) Click "Delete Variables" to delete variable corresponded the device, and record variable's "Type" and "Name".
- 10) Click "Add Variable..." to add variable corresponded the device. Set member variable's type same as removed member variable's type. (Distinguish a small letter and a capital letter.)
- 11) Display the "Message Maps" screen. Align the cursor with the Object ID for the installed device. Select Message that you want remove and click "Delete function".
- 12) Click "Add function". Set the function name same as removed function's name.

Notice:

After this process, two functions for event handling may be made. In that case, remove function that is not necessary.

3.2 Using Constants

All the constants used in the programs found in this manual and UPOS are defined in the .BAS files or .H files that are included in the EPSON OPOS ADK.

These constants are copied to the installation directory by extracting [Samples.zip], which exists in the same layer as the EPSON OPOS ADK installer, and executing [installSample.bat]. This software supplies the following two types of .BAS files. Visual Basic users should make use of these.

OPOS.BAS Constants defined in the OPOS API.

OPOSEPSN.BAS Original Epson constants.

Before creating a program, please be sure to include these .BAS files into your project.

This software supplies the following .H files. Visual C++ users should make use of

these.

OPOS.H Constants used with OPOS in general
OPOSSTAT.H Constants used with OPOS in general
OPOSCASH.H Constants used with OPOS CashDrawer
OPOSDISP.H Constants used with OPOS LineDisplay
OPOSPTR.H Constants used with OPOS POSPrinter
OPOSMICR.H Constants used with OPOS MICR
OPOSCSCN.H Constants used with OPOS CheckScanner
OPOSEJ.H Constants used with OPOS ElectronicJournal
EPSON.H Original EPSON constants used in general
EPSNCASH.H Original EPSON constants used with CashDrawer
EPSNDISP.H Original EPSON constants used with LineDisplay
EPSNPTR.H Original EPSON constants used with POSPrinter
EPSNMICR.H Original EPSON constants used with MICR
EPSNCSCAN.H Original EPSON constants used with CheckScanner
EPSNEJ.H Original EPSON constants used with ElectronicJournal

3.3 API Function Comparison/Pasting

All the usable methods and properties available in the OPOS can be referenced in UPOS, and can also be viewed using the Visual Basic Object Browser. To do this, choose the control module of the device you want to view with the Object Browser. A list of the methods and properties of the device will appear. Please select the item you want to view from this list. An outline of the method or property that you chose along with possible parameters will be shown. Using the Paste command, codes can be added to the procedure.

Section 4. OPOS API Programming

This section outlines important points of OPOS API programming and contains explanation of the shared functions not described in UPOS.

4.1 Things to Consider when Programming

- This section contains a summary explanation of programming. For space and convenience, some points on error management have been omitted. Errors that occur when running API's can be found as needed in other sections throughout this manual. And also refer to details of error specified in the "Device Class Specific Programming" section.
- The Visual Basic programming samples may include hard returns in the code due to manual space limitations. Visual Basic code must include a character '_' to signal a hard return to function properly.

4.2 Code Explanations

Codes used in programming POS applications are listed below with their hex equivalents.

EOT	&H04
ENQ	&H05
LF	&H0A
CLR	&H0C
CR	&H0D
DLE	&H10
CAN	&H18
ESC	&H1B
GS	&H1D
US	&H1F

4.3 Coding Samples

Several coding samples for Visual Basic and Visual C++ are introduced in the following. For the Visual Basic sample coding in the device class specific application development guide, please refer to these samples.

4.3.1 Judging Capability

The following is a coding example for judging “availability of journal functions” in the POSprinter.

<< Visual Basic >>

```
If OPOSPOSPrinter1.CapJrnPresent = True Then
    'Journal functions are available
Else
    'Journal functions are not available
End If
```

<< Visual C++ >>

```
if (m_Ptr1.GetCapJrnPresent())
{
    // Journal functions are available
}
else
{
    // Journal functions are not available
}
```

4.3.2 Retrieving Property Information

The following is a coding example for retrieving a list of the character sets of the POS printer.

<< Visual Basic >>

```
Dim CSList As String
CSList = OPOSPOSPrinter1.CharacterSetList
```

<< Visual C++ >>

```
CString csList;
csList = m_Ptr1.GetCharacterSetList();
```

4.3.3 Using Method Event

The following is a coding example for executing asynchronous printing on the POS printer and confirming that the printing has been completed. This example includes processing when printing finishes and in the event of the occurrence of an error.

[Visual BASIC event processing]

By placing a device's object on a form, events that can be used by the object are pre-set in the application's code. Please describe what to do when a given event occurs.

[Visual C++ Adding event handler]

1. Start ClassWizard and display the message map screen.
2. Choose CO in "Object ID".
3. To add the required event handlers, select these from the Event list in "Messages".

<< Visual Basic >>

[Main Program]

```
Global PrintID1 As Long
Global PrintID2 As Long
OPOSPOSPrinter1.AsyncMode = True
OPOSPOSPrinter1.PrintNormal PTR_S_JOURNAL, "Print Data" + Chr(&H0D) +
Chr(&H0A)
If OPOSPOSPrinter1.ResultCode = OPOS_SUCCESS Then
'Data was set successfully.
```

```

PrintID1 = OPOSPOSPrinter1.OutputID
Elseif OPOSPOSPrinter1.ResultCode = OPOS_E_ILLEGAL Then
    'The station does not exist.
Else
    'Other error.
End If
While PrintID2 <> PrintID1
    'Wait for print success
Wend
'Data was printed successfully.

```

[Event Program]

```

Private Sub OPOSPOSPrinter1_OutputCompleteEvent(ByVal OutputID As
Long)

```

```

    PrintID2 = OutputID
End Sub

```

```

Private Sub OPOSPOSPrinter1_ErrorEvent(ByVal ResultCode As Long,
                                       ByVal ResultCodeExtended As Long,
                                       ByVal ErrorLocus As Long,
                                       pErrorResponse As Long)

```

```

    'ResultCode is obtained and the error can be dealt with.
    'Errors such as OPOS_E_TIMEOUT, OPOS_E_BUSY,
    'and OPOS_E_OFFLINE are possible results.

```

```

If ResultCode = OPOS_E_EXTENDED Then

```

```

    If ResultCodeExtended = OPOS_EPTR_COVER_OPEN Then
        'Cover open.

```

```

    Elseif ResultCodeExtended = OPOS_EPTR_JRN_EMPTY Then
        'Journal paper out.

```

```

    End If

```

```

End If

```

```

pErrorResponse = OPOS_ER_RETRY

```

```

    'When the program should try again

```

```

pErrorResponse = OPOS_ER_CLEAR

```

```

    'When the program should clear the data

```

```

End Sub

```

<< Visual C++ >>

[Main Program]

(Global)

long IPrintID1;

long IPrintID2;

(Main Program)

long lrc;

m_Ptr1.SetAsyncMode(TRUE);

m_Ptr1.PrintNormal(PTR_S_JOURNAL, "Print Data\n");

if ((lrc = m_Ptr1.GetResultCode()) == OPOS_SUCCESS)

{

 // Data was set successfully.

 IPrintID1 = m_Ptr1.GetOutputID();

}

else if (lrc == OPOS_E_ILLEGAL)

{

 // The station does not exist.

}

else

{

 // Other error.

}

while(IPrintID2 != IPrintID1){

 // Wait for print success

}

 // Data was printed successfully.

[Event Program]

void (Dialog Class Name)::OnOutputCompleteEventPOSPrinter1(long OutputID)

{

 IPrintID2 = OutputID;

}

void (Dialog Class Name)::OnErrorEventPOSPrinter1(long ResultCode,

```

        long ResultCodeExtended,
        long ErrorLocus,
        long FAR* pErrorResponse)
// ResultCode is obtained and the error can be dealt with.
// Errors such as OPOS_E_TIMEOUT, OPOS_E_BUSY,
// and OPOS_E_OFFLINE are possible results.
{
    if (ResultCode == OPOS_E_EXTENDED)
    {
        if (ResultCodeExtended == OPOS_EPTR_COVER_OPEN)
        {
            // Cover open.
        }
        else if ( ResultCodeExtended == OPOS_EPTR_JRN_EMPTY)
        {
            // Journal paper out.
        }
    }
    *pErrorResponse = OPOS_ER_RETRY; // When the program should
try again
    *pErrorResponse = OPOS_ER_CLEAR; // When the program should
clear the data
}

```

Section 5. Class Specific Information

Special manuals containing class specific information and description of special programming methods have been prepared for the various device classes. Please refer to the manual(s) for the device(s) to be used. It is recommended to do this while also referring to the device's product manual.

Section 6. Error Information

The methods and properties common for all the devices return the same errors. The following sections explain the errors common for the methods and properties.

6.1 When Executing Common Properties

The following table lists the ResultCode and ResultCodeExtended when executing the Common properties.

Property name	ResultCode	ResultCodeExtended	Meaning
AutoDisable	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i>
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
BinaryConversion	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i>
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	OPOS_EX_BADPROPVAL	Set value is incorrect.
DataEventEnabled	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i>
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	0	Refer to <i>UPOS Specifications</i> .
DeviceEnabled	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCALIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_FAILURE	0	Refer to <i>UPOS Specifications</i> .
FreezeEvent	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	0	Refer to <i>UPOS Specifications</i> .
PowerNotify	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	0	Refer to <i>UPOS Specifications</i> .
		OPOS_EX_INCAPABLE	Function cannot be used.
		OPOS_EX_BADPROPVAL	Set value is incorrect.

6.2 When Executing Common Methods

The following table lists the ResultCode and ResultCodeExtended when executing the Common methods.

Method name	ResultCode	ResultCodeExtended	Meaning
Open	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	0	Already open.
		OPOS_EX_BADCO	The interface of the CO is illegal.
		OPOS_EX_MICRMODE	The port is locked by the other device.
		OPOS_EX_BADPORT	Registry information about communication port is illegal.
		OPOS_EX_REOPEN	Already open.
		OPOS_EX_BADDISPRANGE	DispatchRange is illegal.
		OPOS_EX_BADPEEKRANGE	PeekRange is illegal.
	OPOS_E_CLOSED (OPOS_E_NOSERVICE)	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED (OPOS_E_NOEXIST)	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED (OPOS_E_ILLEGAL)	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED (OPOS_E_FAILURE)	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_EXISTS	0	The file or registry key does not exist.
Close	OPOS_E_CLOSED (OPOS_SUCCESS)	0	Refer to <i>UPOS Specifications</i> .

Method name	ResultCode	ResultCodeExtended	Meaning
(ClaimDevice) *1	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_TIMEOUT	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	OPOS_EX_BADPARAM + 1	The 1st parameter is illegal.
		OPOS_EX_BADPORT	Registry information about communication port is illegal.
		OPOS_EX_DEVBUSY	The outputting cannot be executed because the communication port state is BUSY.
		OPOS_EX_PORTUSED	The Communication port is used by other application.
		OPOS_EX_TIMEOUT	The operation cannot be completed within the timeout period.
		OPOS_EX_NOINPUT	No data is received.
		OPOS_EX_MICRMODE	The port is locked by the other device.
		OPOS_EX_BADDEVICE	The connected device is illegal.
		OPOS_EPTR_MECHANICAL	A mechanical error occurred.
		OPOS_EPTR_CUTTER	A cutter error occurred.
		OPOS_EPTR_UNRECOVERABLE	An irrecoverable error occurred.
		OPOS_EPTR_AUTORECOVERABLE	An overheat error occurred. An auto recoverable error occurred.
		OPOS_EPTR_JRN_EMPTY	Journal station is out of paper.
		OPOS_EPTR_REC_EMPTY	Receipt station is out of paper.
		OPOS_EPTR_COVER_OPEN	Cover is open.
		OPOS_EPTR_REC_CARTRIDGE_REMOVED	No ink cartridge.
		OPOS_EPTR_REC_CARTRIDGE_EMPTY	The ink cartridge is almost empty.
		OPOS_EPTR_REC_HEAD_CLEANNING	Head cleaning in execution.
		OPOS_EPTR_SLP_CARTRIDGE_REMOVED	No ink cartridge.
		OPOS_EPTR_SLP_CARTRIDGE_EMPTY	The ink cartridge is almost empty.
		OPOS_EPTR_SLP_HEAD_CLEANNING	Head cleaning in execution.
		OPOS_EPTR_LABEL_JAM	A label paper jam occurred within the peeler system.
		OPOS_EPTR_LABEL_REMOVAL	Could not carry out the output processing since the device is at the waiting state of the label paper to be removed.
		OPOS_EPTR_BUTTON_OPERATION	Could not carry out the output processing since the device is at the push-waiting state of the button.

*1 For ClaimDevice, the error information differs by models.

Method name	ResultCode	ResultCodeExtended	Meaning
(ClaimDevice) ^{*1}	OPOS_E_FAILURE	0	Refer to <i>UPOS Specifications</i> .
		OPOS_EX_MICRMODE	The port is locked by the other device.
		OPOS_EPTR_UNRECOVERABLE	An irrecoverable error occurred.
		OPOS_EPTR_CUTTER	A cutter error occurred.
		OPOS_EPTR_MECHANICAL	A mechanical error occurred.
		OPOS_EPTR_OVERHEAT	An overheat error occurred.
	OPOS_E_TIMEOUT	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_OFFLINE	0	Refer to <i>UPOS Specifications</i> .
		OPOS_EPTR_COVEROPEN	Cover is open.
		OPOS_EPTR_JRN_EMPTY	Journal station is out of paper.
ReleaseDevice	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	OPOS_EX_NOTCLAIMED	The device was released without claiming.
CheckHealth	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_DISABLED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	OPOS_EX_BADPARAM+1	The 1st parameter is illegal.
		OPOS_EX_DEVBUSY	The outputting cannot be executed because the communication port state is BUSY.
		OPOS_EX_TIMEOUT	The operation cannot be completed within the timeout period.
		OPOS_EX_NOTSUPPORTED	Not supported.
		OPOS_EX_MICRMODE	The port is locked by the other device.
		OPOS_EX_INVALIDMODE	The state is invalid mode.
	OPOS_E_OFFLINE	0	Refer to <i>UPOS Specifications</i> .
		OPOS_EPTR_COVER_OPEN	Cover is open.
		OPOS_EPTR_JRN_EMPTY	Journal station is out of paper.
		OPOS_EPTR_REC_EMPTY	Receipt station is out of paper.
		OPOS_EPTR_REMOVE_BUTTON	Could not carry out the output processing since the device is at the waiting state of the label paper to be removed and the push-waiting state of the button.

^{*1} For ClaimDevice, the error information differs by models.

Method name	ResultCode	ResultCodeExtended	Meaning
(CheckHealth)	OPOS_E_FAILURE	OPOS_EX_INVALIDMODE	The state is invalid mode.
		OPOS_EX_DEVBUSY	The outputting cannot be executed because the communication port state is BUSY.
		OPOS_EX_TIMEOUT	The operation cannot be completed within the timeout period.
		OPOS_EX_NOINPUT	No data is received.
		OPOS_EX_MICRMODE	The port is locked by the other device.
		OPOS_EPTR_UNRECOVERABLE	An irrecoverable error occurred.
		OPOS_EPTR_AUTORECOVERABLE	An overheat error occurred. An auto recoverable error occurred.
		OPOS_EPTR_CUTTER	A cutter error occurred.
		OPOS_EPTR_MECHANICAL	A mechanical error occurred.
		OPOS_EPTR_LABEL_JAM	A label paper jam occurred within the peeler system.
		OPOS_EPTR_OVERHEAT	An overheat error occurred.
		OPOS_EX_UNAUTHORIZED	There was insufficient permission to access for processing.
	OPOS_E_EXTENDED	OPOS_EPTR_COVER_OPEN	Cover is open.
		OPOS_EPTR_JRN_EMPTY	Journal station is out of paper.
		OPOS_EPTR_REC_EMPTY	Receipt station is out of paper.
		OPOS_EPTR_SLP_EMPTY	Slip station is out of paper.
		OPOS_EPTR_REC_CARTRIDGE_REMOVED	No ink cartridge.
		OPOS_EPTR_REC_CARTRIDGE_EMPTY	The ink cartridge is almost empty.
		OPOS_EPTR_REC_HEAD_CLEANING	Head cleaning in execution.
		OPOS_EPTR_SLP_CARTRIDGE_EMPTY	The ink cartridge is almost empty.

Method name	ResultCode	ResultCodeExtended	Meaning
(CheckHealth)	(OPOS_E_EXTENDED)	OPOS_EPTR_SLP_HEAD_CLEANING	Head cleaning in execution.
		OPOS_EPTR_LABEL_REMOVAL	Could not carry out the output processing since the device is at the waiting state of the label paper to be removed.
		OPOS_EPTR_BUTTON_OPERATION	Could not carry out the output processing since the device is at the push-waiting state of the button.
		OPOS_EEJ_NOT_ENOUGH_SPACE	There is not enough space in the Medium to continue this operation.
	OPOS_E_BUSY	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOHARDWARE	0	Refer to <i>UPOS Specifications</i> .
ClearInput	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLAIMED	0	Refer to <i>UPOS Specifications</i> .
ClearOutput	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLAIMED	0	Refer to <i>UPOS Specifications</i> .
DirectIO	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_DISABLED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	0	Refer to <i>UPOS Specifications</i> .
		OPOS_EX_DEVBUSY	The outputting cannot be executed because the communication port state is BUSY.
		OPOS_EX_INVALIDMODE	The state is invalid mode.
		OPOS_EX_NOTSUPPORTED	Not supported.
		OPOS_EX_PORTUSED	The Communication port is used by other application.
		OPOS_EX_BADPORT	Registry information about communication port is illegal.

Method name	ResultCode	ResultCodeExtended	Meaning
(DirectIO)	(OPOS_E_ILLEGAL)	OPOS_EX_TIMEOUT	The operation cannot be completed within the timeout period.
		OPOS_EX_INCAPABLE	No function.
		OPOS_EX_MICRMODE	The port is locked by the other device.
		OPOS_EX_BADPARAM+1	The 1st parameter is illegal.
		OPOS_EX_BADPARAM+2	The 2nd parameter is illegal.
	OPOS_E_NOHARDWARE	0	The device is power OFF or unconnected.
	OPOS_E_FAILURE	OPOS_EX_DEVBUSY	The outputting cannot be executed because the communication port state is BUSY.
		OPOS_EX_TIMEOUT	The operation cannot be completed within the timeout period.
	OPOS_E_EXTENDED	OPOS_EDISP_TOOBIG	The size is too big.
		OPOS_EDISP_BADFORMAT	The format of the specified file is illegal.
ResetStatistics	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_DISABLED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	OPOS_EX_BADPARAM+1	The 1st parameter is illegal.
		OPOS_EX_INCAPABLE	No function.
	OPOS_E_EXTENDED	OPOS_ESTATS_ERROR	The specified statistics name is un-updatable or un-resettable.
RetrieveStatistics	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_DISABLED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	OPOS_EX_BADPARAM+1	The 1st parameter is illegal.
		OPOS_EX_INCAPABLE	No function.

Method name	ResultCode	ResultCodeExtended	Meaning
UpdateStatistics	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_DISABLED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	OPOS_EX_BADPARAM+1	The 1st parameter is illegal.
		OPOS_EX_INCAPABLE	No function.
	OPOS_E_EXTENDED	OPOS_ESTATS_ERROR	The specified statistics name is un-updatable or un-resettable.
CompareFirmware Version	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_DISABLED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	OPOS_EX_INCAPABLE	No function.
UpdateFirmware	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_DISABLED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_ILLEGAL	OPOS_EX_INCAPABLE	No function.
ClearInputProperties	OPOS_SUCCESS	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLOSED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_NOTCLAIMED	0	Refer to <i>UPOS Specifications</i> .
	OPOS_E_CLAIMED	0	Refer to <i>UPOS Specifications</i> .

Section 7. Warnings

7.1 Cautions for Programming

- After a method has been called and until it is returned, the calling thread is placed in a hanged (frozen) state.

For example, when there is only one form in the program and a method is called, the program is completely frozen until the method is returned. Some methods (ClaimDevice, WaitForDrawerClose, etc.) take a long time to complete, so please arrange the program to plan on this.

This problem can be avoided if the program calls methods from another form.

- The errors that may be returned from the Open or ClaimDevice methods may vary depending on the device being used. This is partly dependent on the port connection that the method is performed. Different types of devices by different makers will return different values. A device that is actually connected to a port being addressed by the Open method will return errors related to the Open method. Devices being connected by the ClaimDevice method will return errors related to the ClaimDevice method. For details on this, please refer to the explanation of errors in the “Class Specific Programming” section.
- It is illegal to use DoEvents management to perform event management. If DoEvents management is used, an application error will appear, and the program may stop responding and the message calling for forced shutdown may appear.
- Depending on the device being used, it is possible to cause an error after the Open method is executed if the power is turned OFF/ON. Please not to turn the power OFF/ON after the Open method has been called. If power is turned OFF/ON, please execute the Close method and then execute the Open method again.

7.2 Precautions for VC++

7.2.1 LineDisplay

When using LineDisplay control in coexistence with MFC, the two method names “CreateWindow” and “DestroyWindow” will duplicate already existing function names and an error will occur at the time of building. To avoid this, it is necessary to change the function names in the wrapper class (Codisp.cpp Codisp.h). The following sample shows how to change these.

Points to change in Codisp.h

```
long CreateWindow0(    long ViewportRow, long ViewportColumn,
                        long ViewportHeight, long ViewportWidth,
                        long WindowHeight, long WindowWidth);
                        long DestroyWindow0();
```

Points to change in Codisp.cpp

```
long CCodisp::CreateWindow0(    long ViewportRow, long ViewportColumn,
long ViewportHeight, long ViewportWidth,
long WindowHeight, long WindowWidth)
{
.....
}

long CCodisp::DestroyWindow0()
{
.....
}
```


7.2.2 Cautions when Using the Sample Programs

Each of the sample programs provided with OPOS ADK creates with the device name as “Unit 1”. To operate the sample program, please specify the Logical Device Name as “Unit 1”.

To keep the capacity required for the program as small as possible, the dynamic-link library feature is set to use.

7.2.3 Cautions for Use of DirectIO

The third argument of DirectIO is the pointer to BSTR.

```
long DirectIO(long Command, long* pData, BSTR* pString)
```

The Service Object treats the contents of this pointer as Unicode(WideChar). Accordingly, the argument pString must be a pointer to BSTR type Unicode data.

BSTR type Unicode data can be created using methods like the following.

```
BSTR pString = SysAllocString(L“abcd”);
```

```
CString m_strString = “abcd”;  
BSTR pString = m_strString.AllocSysString();
```

```
LPCSTR lpszA = “abcd”;  
int nLen = MultiByteToWideChar(CP_ACP, 0, lpszA, 4, NULL, NULL);  
BSTR pString = ::SysAllocStringLen(NULL, nLen);  
MultiByteToWideChar(CP_ACP, 0, lpszA, 4, pString, nLen);
```

When building, use the _UNICODE symbol and use the following types.

TCHAR, LPTSTR, LPCTSTR, CString

Please use _T macros for literal characters.