EPSON
EXCEED YOUR VISION

PLQ-22CS/PLQ-22CSM API Reference Guide for Windows

Overview

Explains the features and development environment.

Driver Setup

Explains how to install and uninstall the printer and scanner drivers, and verify proper operation.

Programming Guide

Explains the programming method for developing an application that uses PLQ-22CS/PLQ-22CSM.

TWAIN API Reference

Explains the TWAIN APIs used by PLQ-22CS/PLQ-22CSM.

PLQ-22 API Reference

Explains the PLQ-22 APIs.

PLQ-22 .NET API Reference

Explains the PLQ-22 API used in .NET environment.

Sample Programs

Explains the sample programs provided by PLQ-22CS/PLQ-22CSM.

M00027001
Rev. B

Cautions

- No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Seiko Epson Corporation.
- The contents of this document are subject to change without notice. Please contact us for the latest information.
- While every precaution has taken in the preparation of this document, Seiko Epson Corporation assumes no responsibility for errors or omissions.
- Neither is any liability assumed for damages resulting from the use of the information contained herein.
- Neither Seiko Epson Corporation nor its affiliates shall be liable to the purchaser of this product or third parties for damages, losses, costs, or expenses incurred by the purchaser or third parties as a result of: accident, misuse, or abuse of this product or unauthorized modifications, repairs, or alterations to this product, or (excluding the U.S.) failure to strictly comply with Seiko Epson Corporation's operating and maintenance instructions.
- Seiko Epson Corporation shall not be liable against any damages or problems arising from the use of any options or any consumable products other than those designated as Original EPSON Products or EPSON Approved Products by Seiko Epson Corporation.

Trademarks

EPSON® is a registered trademark of Seiko Epson Corporation.

MS-DOS®, Microsoft®, Windows®, Windows Vista®, Windows Server®, Visual Studio®, Visual Basic®, and Visual C++® are registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

For Safety

Key to Symbols

The symbols in this manual are identified by their level of importance, as defined below. Read the following carefully before handling the product.

CAUTION	Provides information that must be observed to avoid damage to your equipment or a malfunction.
NOTE	Provides important information and useful tips.

Restriction of Use

When this product is used for applications requiring high reliability/safety such as transportation devices related to aviation, rail, marine, automotive etc.; disaster prevention devices; various safety devices etc; or functional/precision devices etc, you should use this product only after giving consideration to including fail-safes and redundancies into your design to maintain safety and total system reliability.

Because this product was not intended for use in applications requiring extremely high reliability/safety such as aerospace equipment, main communication equipment, nuclear power control equipment, or medical equipment related to direct medical care etc, please make your own judgment on this product's suitability after a full evaluation.

About this Manual

Aim of the Manual

This manual is aimed at development engineers and provides all necessary information for developing an application using PLQ-22CS/PLQ-22CSM.

NOTE

- For customers in China, replace references to PLQ-22CS/PLQ-22CSM in this manual with PLQ-22KCS/PLQ-22KCSM.
- For customers in Taiwan, replace references to PLQ-22CS/PLQ-22CSM in this manual with PLQ-22CCS/PLQ-22CCSM.

Manual Content

The manual is made up of the following sections:

Contents

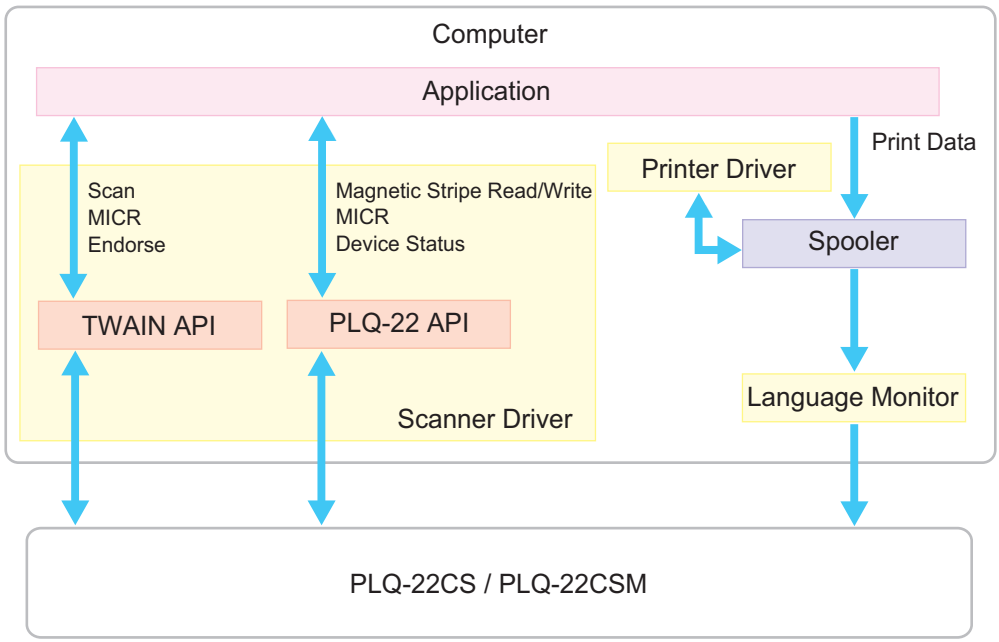
Overview

This chapter explains the features and specifications of the product.

System Overview

The PLQ-22CS/PLQ-22CSM uses the TWAIN API and the PLQ-22 API to operate the following features of the scanner driver:

- ❏ Scanner
- ❏ MICR Reader
- ❏ Magnetic Stripe Read/Write
- ❏ Endorsement Printing (Endorse)
- ❏ Device Status Check



CAUTION For PLQ-22 API, while TWAIN API is activated, the functions other than the Status function (those for MSRW and MICR) are not available.

Packages

Printer Driver Installer

A package that installs the printer driver.

Scanner Driver Installer

A package that installs the scanner driver (TWAIN API and PLQ-22 API).

Sample Programs

A compressed file containing sample PLQ-22 API programs.

Manuals

Setup Guide

Explains how to setup the PLQ-22CS/PLQ-22CSM.

User's Guide

Explains how to use the PLQ-22CS/PLQ-22CSM.

API Reference Guide for Windows

This manual. Provides necessary information for developing an application that uses the PLQ-22CS/PLQ-22CSM.

Features

The PLQ-22CS/PLQ-22CSM uses the TWAIN API and the PLQ-22 API to operate the following features:

Image Scanning

- Removal of background from the image enables you to see the text data clearly (Text Enhancement).
- Acquires barcode data.
- Uses OCR-A/B font recognition.
- Saves to file in BMP, JPEG, PDF, or TIFF format.
- Works with scanned images (auto-size, skew correction).
- Removes blank pages.
- Sets dropout color.

Endorsement Printing (Endorse)

- Allows you to switch between Real Endorsement, which prints physically, and Electronic Endorsement, which adds text data to the image data.
- Specifies font size.

MICR

- Uses with OCR function to acquire accurate MICR data.

Magnetic Stripe

- Acquires, edits and deletes Magnetic Stripe data.

Device Status Check

- Retrieves the status of the device.

Development Environment

OS

❏ Microsoft Windows XP (32 bit)

❏ Microsoft Windows Vista (32/64 bit)

❏ Microsoft Windows 7 (32/64 bit)

❏ Microsoft Windows Server 2003 (32/64 bit)

❏ Microsoft Windows Server 2008 (32/64 bit)

❏ Microsoft Windows Server 2008 R2

NOTE

Refer to User's Guide for system requirements.

Development Languages

❏ Visual C++

❏ Visual Basic .NET

Development Tools

Microsoft Visual Studio 2005

External Modules

If you are developing your application in .NET environment, the following external modules are required:

❏ .NET Framework

- Microsoft .NET Framework 2.0 SP2
- Microsoft .NET Framework 3.0 SP1
- Microsoft .NET Framework 3.5 SP1
- Microsoft .NET Framework 3.5.1

CAUTION

- The development of .NET applications applies only to the PLQ-22 API.
- A third-party software may be required to control the TWAIN API in a .NET environment.

Header Files

TWAIN API

- epcstwain.h

Required when using functions that are proprietary to the PLQ-22CS/PLQ-22CSM. These are included in this manual.

PLQ-22 API

- PLQStatusApiPartialInterface.h
- PLQStatusApiDef.h

Required for developing an application using the PLQ-22 API. These are included in the sample programs.
Refer to "Folder Structure" on page 188 for details.

Technical Support Web Site

Epson's Technical Support Web Site provides help with problems that cannot be solved using the troubleshooting information in your product documentation. If you have a Web browser and can connect to the Internet, access the site at:

http://support.epson.net/

If you need the latest drivers, FAQ's, manuals, or other downloadables, access the site at:

http://www.epson.com

Then, select the support section of your local Epson Web site.

Driver Setup

This chapter explains how to install and uninstall the PLQ-22/PLQ-22CSM drivers, and how to verify proper operation.

Installing the Drivers

Install the printer in the following order:

1. Install printer driver (Page 15)

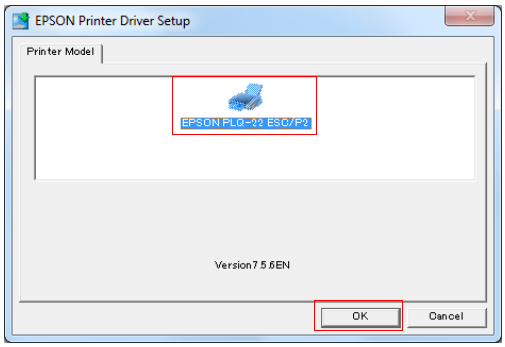
↓

2. Install scanner driver (Page 17)

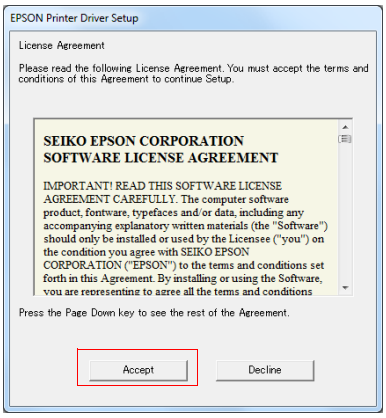
Printer Driver

Follow the steps below to install the printer driver:

1. Connect PLQ-22CS/PLQ-22CSM to the computer.
2. Double-click the “Setup.exe” icon. 
3. If the “User Account Control” screen appears, click [**Continue**].
4. Select [**EPSON PLQ-22 ESC/P2**] and click [**OK**].

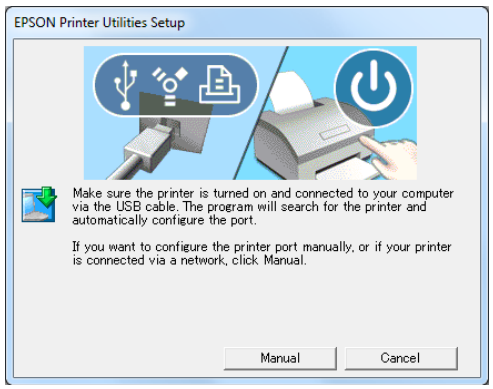


5 Review the License Agreement and click [**Accept**].



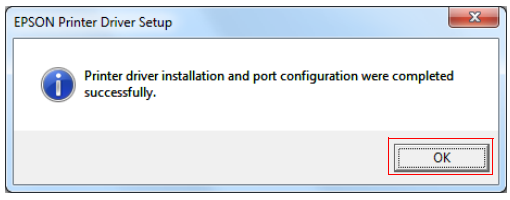
6 After a while, the following screen appears. Make sure that PLQ-22CS/PLQ-22CSM is connected to the computer, and turn on the power on PLQ-22CS/PLQ-22CSM.

> **NOTE** If you are using parallel interface, click [Manual] and specify the connection destination.



7 Make sure that PLQ-22CS/PLQ-22CSM is connected to the computer, and turn on the power.

8 When the following screen appears, click [**OK**]:



This completes the installation.

Scanner Driver

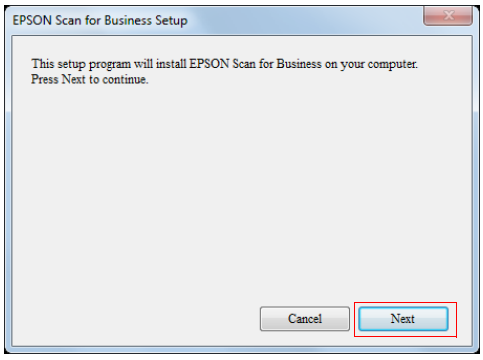
Follow the steps below to install the scanner driver:

> **CAUTION** Before installing the scanner driver, first complete the installation of the printer driver.

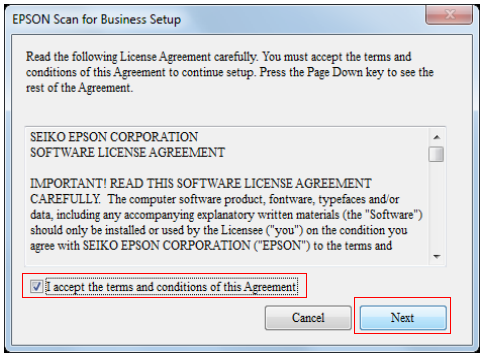
1. Double-click the "Setup.exe" icon. 

2. If the "User Account Control" screen appears, click [**Continue**].

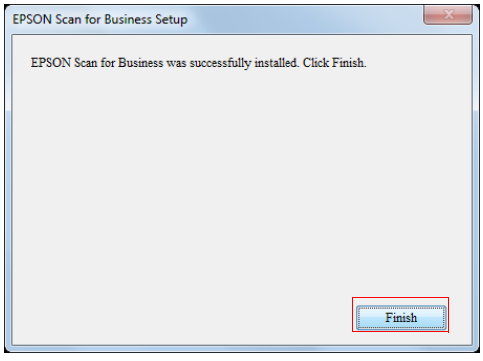
3. Click [**Next**].



4. After reviewing the License Agreement, select [**I accept the terms and conditions of the this Agreement**], and click [**Next**].



5. When the following screen appears, click [**Finish**]:



This completes the installation.

Silent Install

Silent install refers to the method of using the command line option to install a driver without displaying any dialogs or performing any operations.

Command Line Option

Printer Driver

```
setup /P:"EPSON PLQ-22 ESC/P2" /NODISP /NOLA /NOCOPYGAUGE /SKIPS /APD
```

> **NOTE** After executing this command, connect PLQ-22CS/PLQ-22CSM to the computer with a USB cable, and then turn on the power to PLQ-22CS/PLQ-22CSM.

Scanner Driver

```
setup /SI
```

Uninstalling the Drivers

Printer Driver

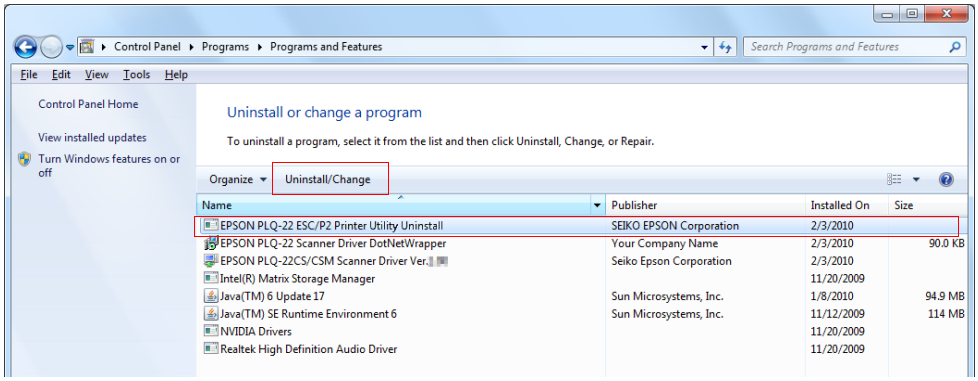
Follow the steps below to uninstall the printer driver:

1 Finish other operations on the computer.

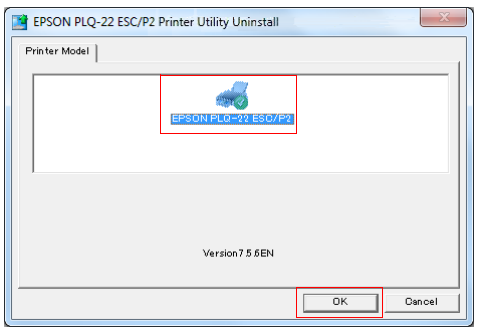
2 Open [**Uninstall a program**] or [**Add or Remove Programs**].

- On Windows 7:
  [**Start**] - [**Control Panel**] - [**Uninstall a program**]
- On Windows Vista:
  [**Start**] - [**Control Panel**] - [**Uninstall a program**]
- On Windows XP Professional:
  [**Start**] - [**Add or Remove Programs**]
- On Windows Server 2003:
  [**Start**] - [**Add or Remove Programs**]
- On Windows Server 2008:
  [**Start**] - [**Control Panel**] - [**Uninstall a program**]
- On Windows Server 2008 R2:
  [**Start**] - [**Control Panel**] - [**Uninstall a program**]

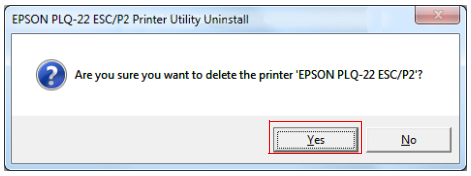
3 Select [**EPSON PLQ-22 ESC/P2 Printer Utility Uninstall**], and click [**Uninstall/Change**].



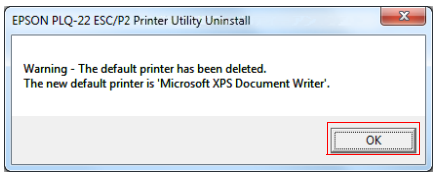
4 The following screen appears. From the [**Printer Model**] tab, select the printer driver you want to uninstall, and click [**OK**].



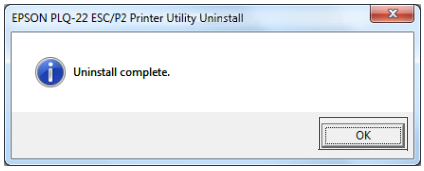
5 The following screen appears. Click [**Yes**].



6 If the printer driver that you uninstalled was the default printer, the following screen appears. Click [**OK**].



7 The following screen appears. Click [**OK**].



This completes the uninstallation.

Scanner Driver

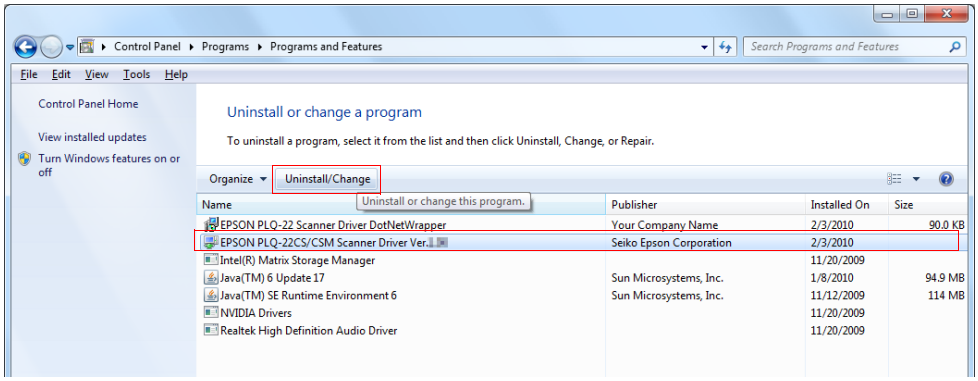
Follow the steps below to uninstall the scanner driver:

1 Finish other operations on the computer.

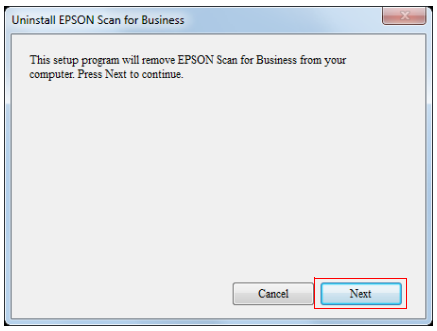
2 Open [**Uninstall a program**] or [**Add or Remove Programs**].

- On Windows 7:
  [**Start**] - [**Control Panel**] - [**Uninstall a program**]
- On Windows Vista:
  [**Start**] - [**Control Panel**] - [**Uninstall a program**]
- On Windows XP Professional:
  [**Start**] - [**Add or Remove Programs**]
- On Windows Server 2003:
  [**Start**] - [**Add or Remove Programs**]
- On Windows Server 2008:
  [**Start**] - [**Control Panel**] - [**Uninstall a program**]
- On Windows Server 2008 R2:
  [**Start**] - [**Control Panel**] - [**Uninstall a program**]

3 Select [**EPSON PLQ-22CS/CSM Scanner Driver Ver.x.xx**], and click [**Uninstall/Change**].

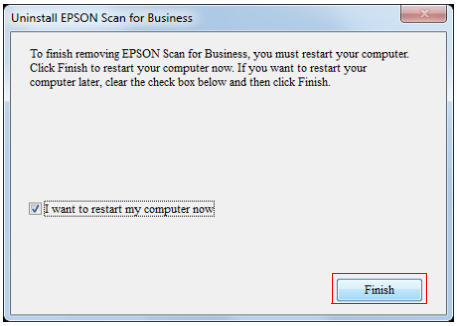


4 The following screen appears. Click [**Next**].



5 The following screen appears. Click [**Finish**].

> **NOTE** If you want to restart your computer later, uncheck “I want to restart my computer now”.



6 If you unchecked “**I want to restart my computer now**” in step 5, restart your computer.

This completes the uninstallation.

Verifying Proper Operation of PLQ-22CS/PLQ-22CSM

Printer Driver

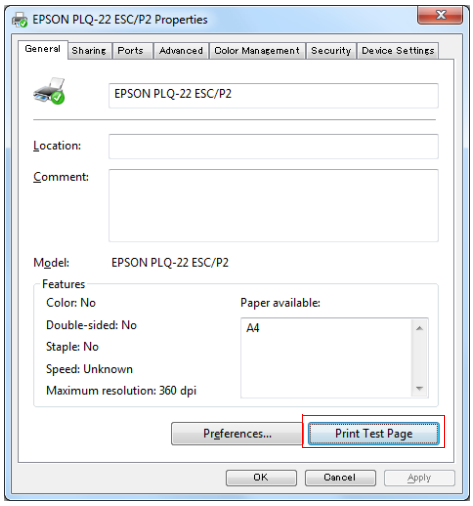
Print a test page.

1 Open [**Printers and Faxes**] (or [**Printers**]).

- On Windows 7:
  [**Start**] - [**Control Panel**] - [**Hardware and Sound**] - [**Devices and Printers**]
- On Windows Vista:
  [**Start**] - [**Control Panel**] - [**Printers**]
- On Windows XP Professional:
  [**Start**] - [**Printers and Faxes**]
- On Windows Server 2003:
  [**Start**] - [**Printers and Faxes**]
- On Windows Server 2008:
  [**Start**] - [**Control Panel**] - [**Printers**]
- On Windows Server 2008 R2:
  [**Start**] - [**Control Panel**] - [**Hardware and Sound**] - [**Devices and Printers**]

2 Right-click on [**EPSON PLQ-22 ESC/P2**], and click [**Properties**] (or [**Printer properties**]).
The properties appear.

3 Click [**Print Test Page**].

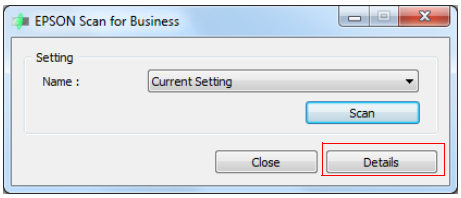


Scanner Driver

Use the EPSON Scan for Business to run a test scan.

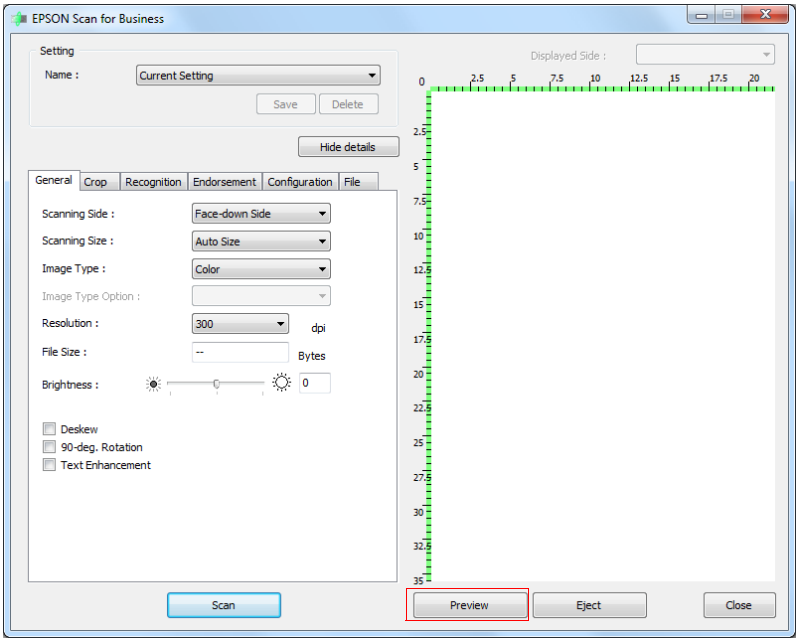
1 Launch EPSON Scan for Business from [**Start**] - [**All Programs**] - [**EPSON**] - [**EPSON Scan for Business**] - [**EPSON PLQ-22CS_CSM**] - [**EPSON Scan for Business**].

2 Click [**Details**].
The detailed mode screen appears.



3 Load a test sheet to PLQ-22CS/PLQ-22CSM.

4 Configure the scan settings and click [**Preview**].



Building an Identical Scanner Driver Environment

With the PLQ-22CS/PLQ-22CSM scanner driver, you can take the setting file from an already built environment, include it in the scanner driver install package so that the same environment can be copied to other computers.

- Scanner setting file:
  This file stores the scan conditions set in EPSON Scan for Business.
- Network setting file:
  This file stores server information for scanning over the network. This information is displayed in the EPSON Scan for Business Setting Utility.

Installing the Setting File

1 Log into the computer from which you want to copy the environment, with Administrator privileges.

2 Launch EPSON Scan for Business, and save the settings.
The file is save to the following folder. (File name: profXXX.ini)

- Windows XP/ Windows Server 2003:

  C:\Documents and Settings\All Users\Application Data\EPSON\EPSON Scan for Business\PLQ-22\Profile
- Windows Vista/ Windows 7/ Windows Server 2008:

  C:\ProgramData\EPSON\EPSON Scan for Business\PLQ-22\Profile

> **NOTE** On Windows Vista, Windows 7 and Windows Server 2008, launch EPSON Scan for Business with Administrator privileges (right-click and select [Run as administrator]).

3 Include the setting file you saved in step 2 to the scanner driver installer file.
Copy it to the following folder:

...Installer file (PLQ22ScnXXX)\PROFILE

4 On a second computer, use the installer file with the setting file to install as you would normally. (Page 17)

Installing Network Settings

1 Log into the computer from which you want to copy the environment, with Administrator privileges.

2 Launch the EPSON Scan for Business Setting Utility, and save the network scanner settings.
The file is save to the following folder. (File name: EnvSetting.ini)

C:\WINDOWS\twain_32\Epson\PLQ-22

> **NOTE** If the “User Account Control” screen appears while EPSON Scan for Business Setting Utility is launching, click [Continue].

3 Include the setting file you saved in step 2 to the scanner driver installer file.
Overwrite or replace into the following folder:

...Installer file (PLQ22ScnXXX)\ENVFILE

4 On a second computer, use the installer file with the setting file to install as you would normally. (Page 17)

Programming Guide

This chapter explains the programming method for developing an application that uses PLQ-22CS/PLQ-22CSM.

The Role of the PLQ-22 Scanner Driver

The PLQ-22 scanner driver has the following APIs, and the available functions vary by API.

TWAIN API

- Operating PLQ-22CS/PLQ-22CSM
- Image Scanning
- Acquiring MICR data
- Acquiring OCR data
- Acquiring barcode data
- Endorsement Printing

PLQ-22 API

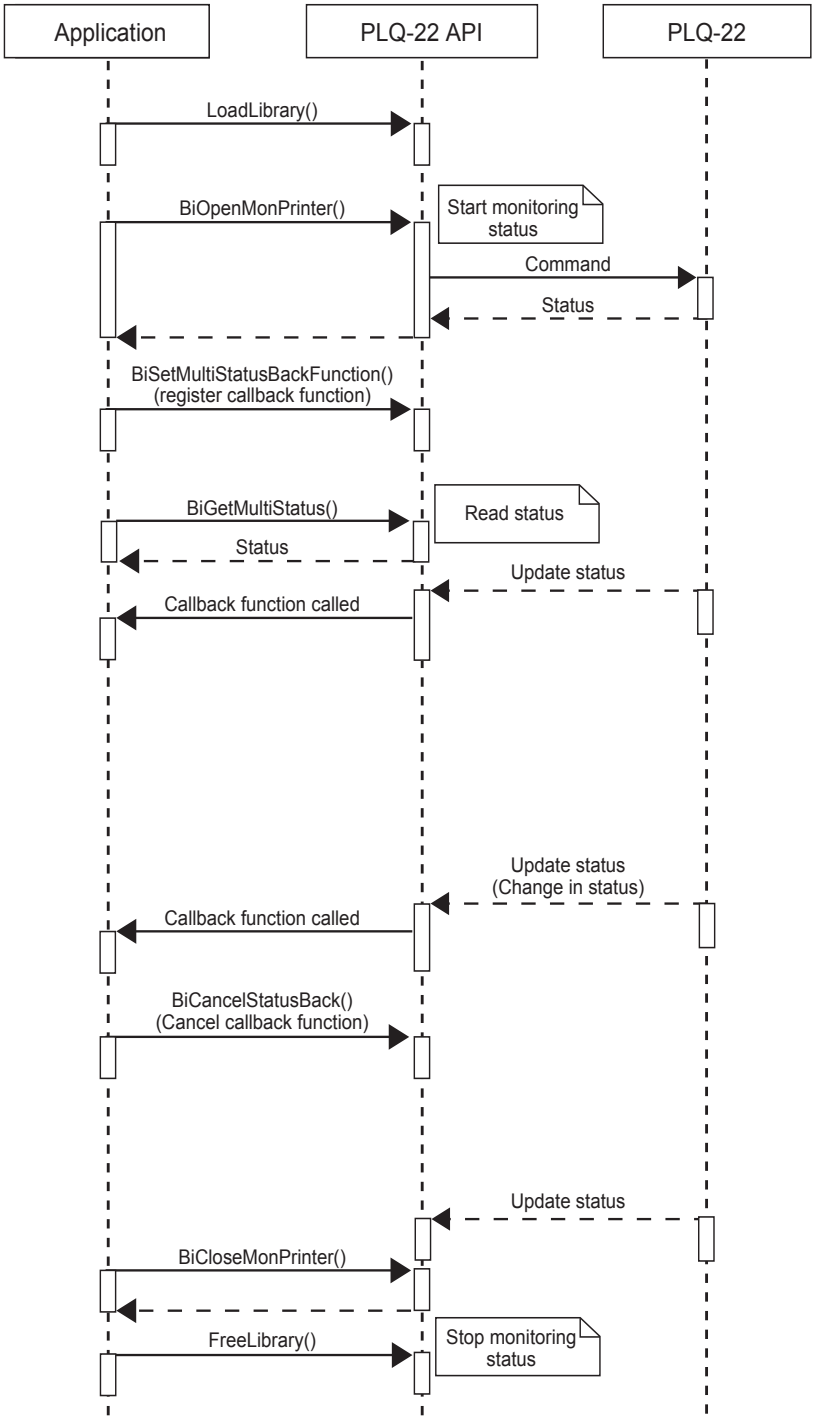
- Retrieving device status
- Acquiring MICR data
- Acquiring, editing, and erasing the Magnetic Stripe data

Programming Flow

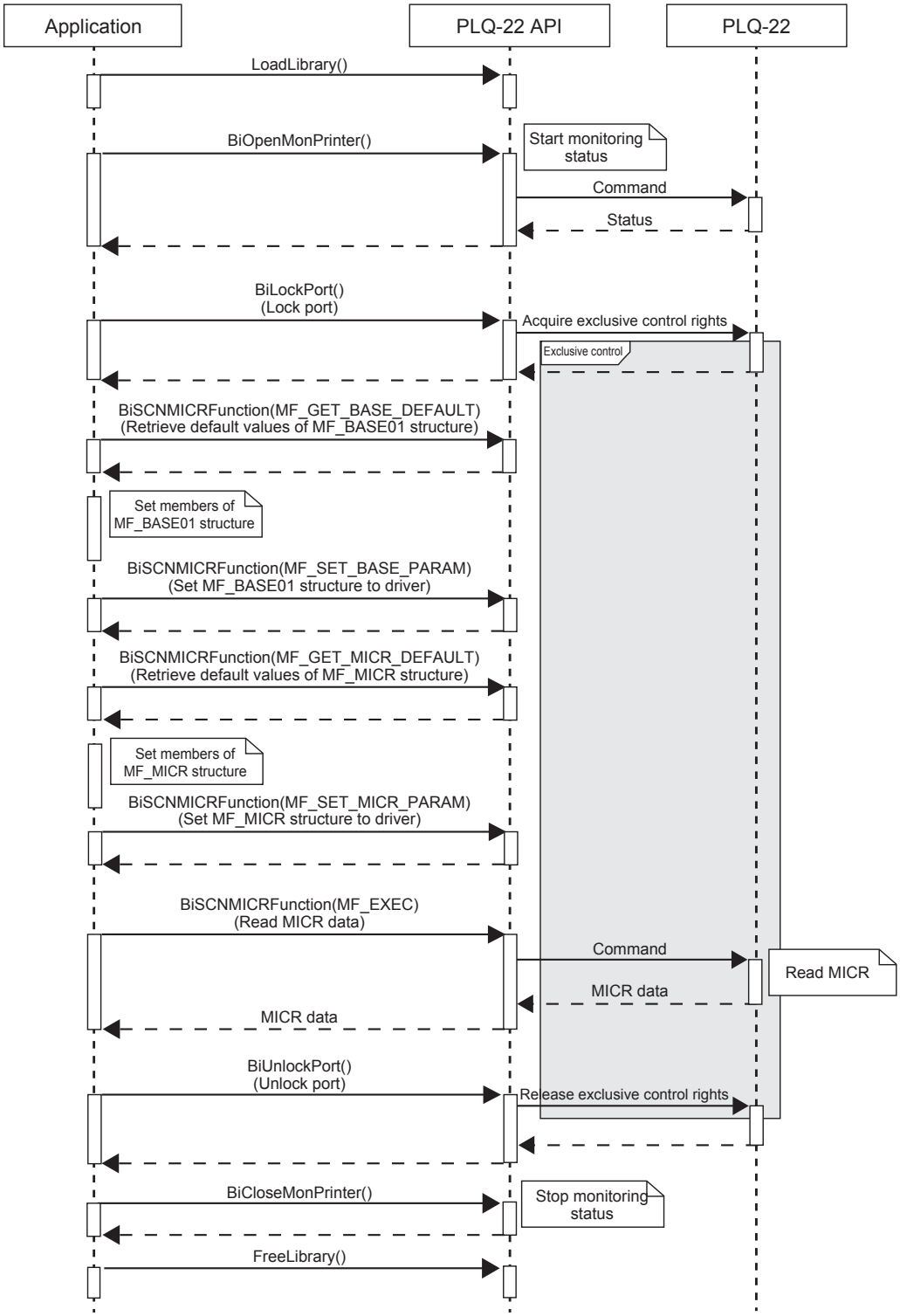
This section explains the programming methods for the PLQ-22 API functions, using sequence diagrams.

> **NOTE** For programming methods using the TWAIN API, refer to the TWAIN Specification from the TWAIN Working Group.

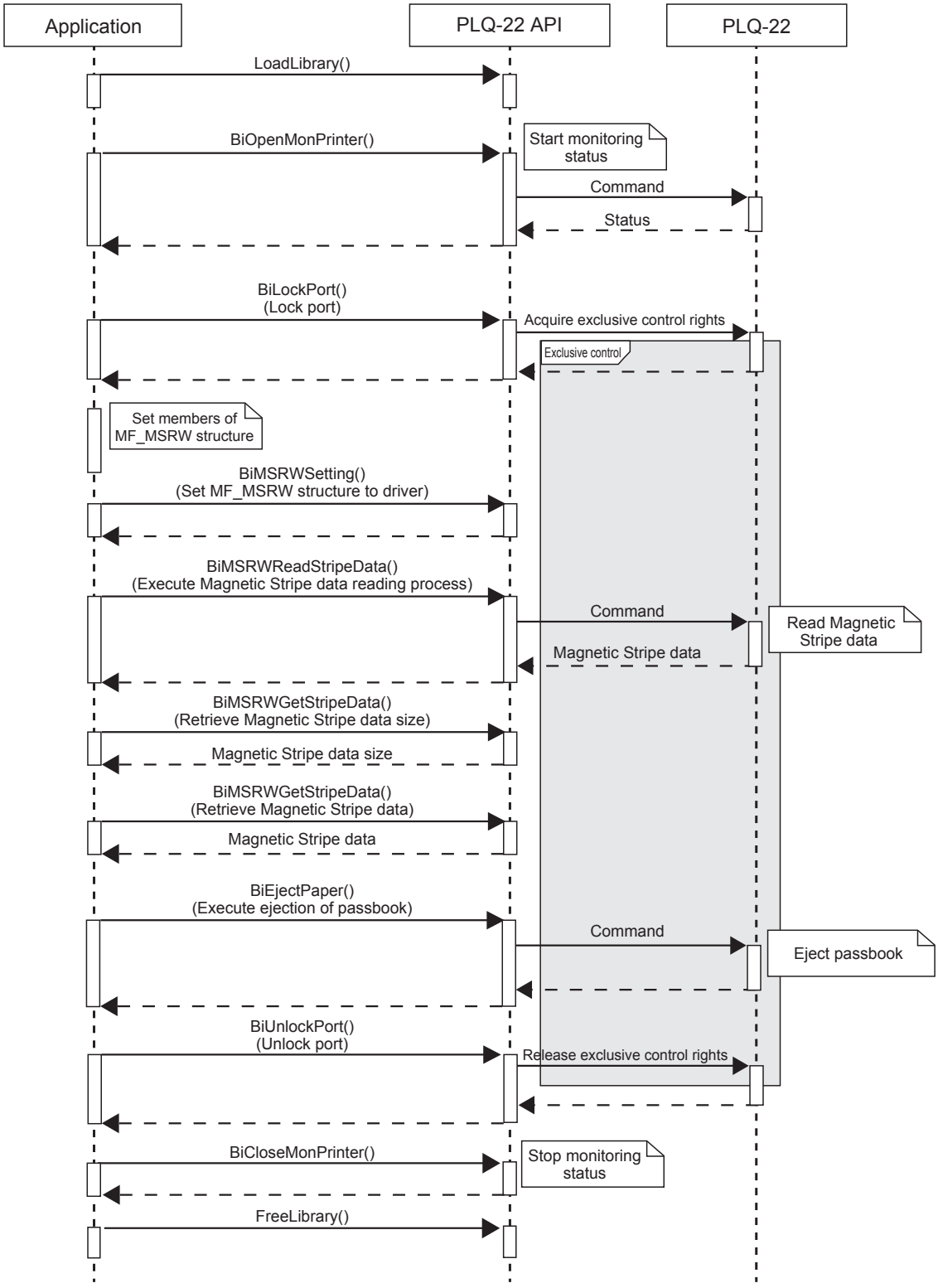
Method for Retrieving Device Status



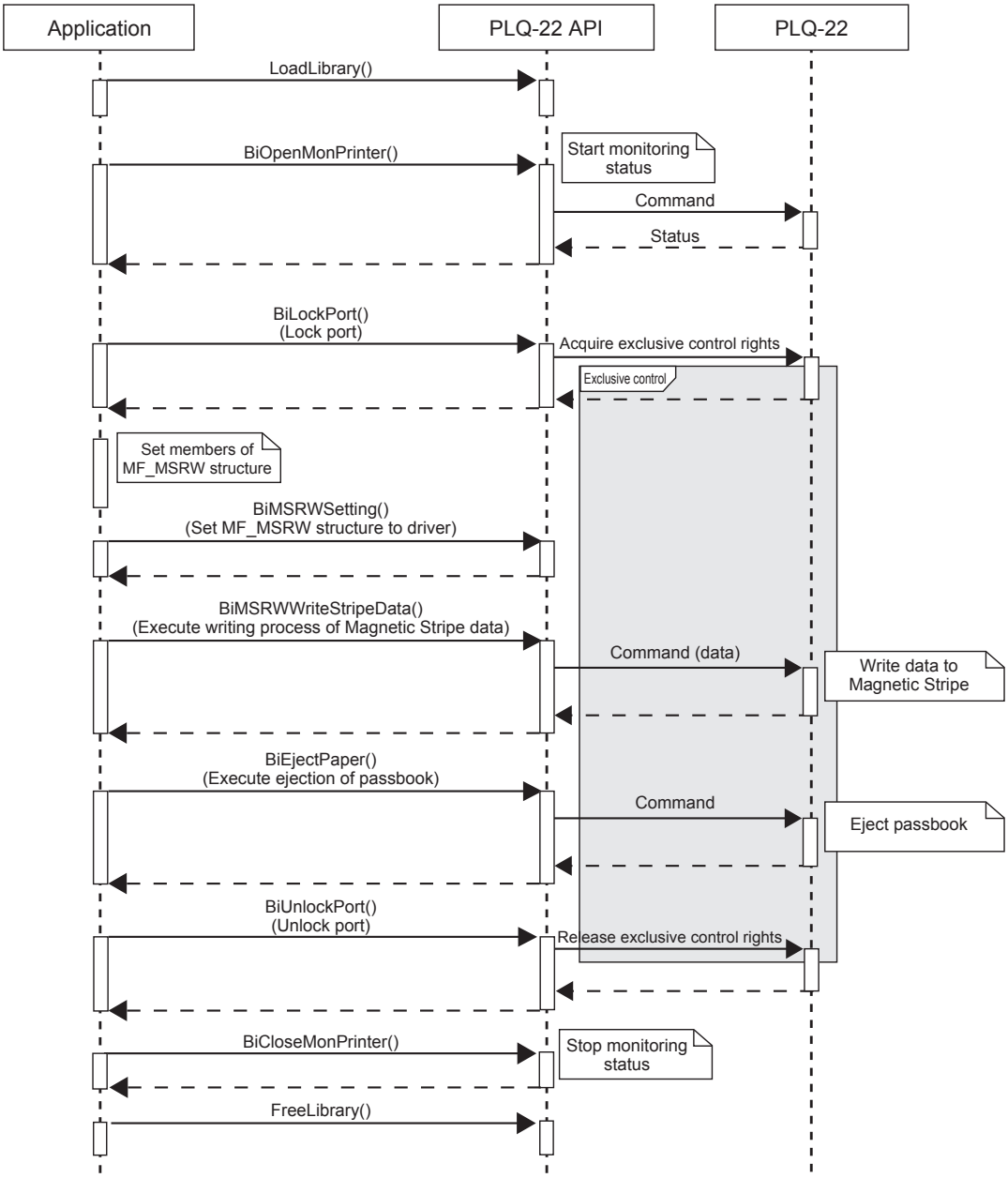
Method for Acquiring MICR Data



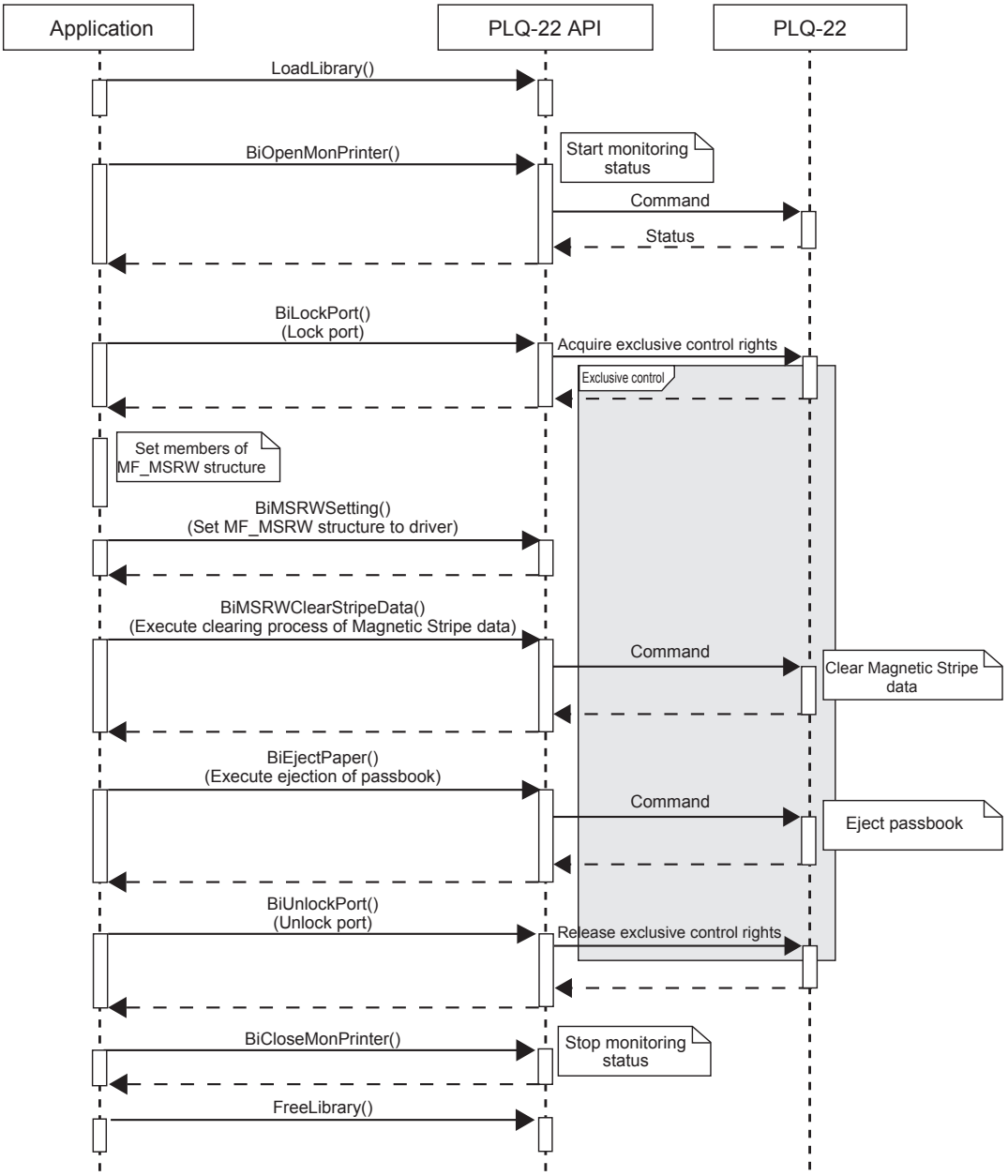
Method for Acquiring the Magnetic Stripe Data



Method for Writing the Magnetic Stripe Data



Method for Erasing the Magnetic Stripe Data



Troubleshooting Errors

Errors that occur in PLQ-22 API can be errors that are notified in the form of a device status, or errors that occur when calling a PLQ-22 API. This section explains the errors and how to troubleshoot them. The errors are to be handled accordingly in your application.

> NOTE
>
> For details on TWAIN, refer to the TWAIN Specification from the TWAIN Working Group.

Device Status

The following are errors that are returned when retrieving the device status:

Macro Definition (Constant)	Cause	Troubleshooting Action
ESCMC_ASB_OFFLINE	The device is offline.	Remove the cause of it being offline.
ESCMC_ASB_PAPER_FEED	The paper is inserted.	-
ESCMC_ASB_PAPER_SET	The paper is set.	-
ESCMC_ASB_COVER_STATUS	The device cover is open.	-
ESCMC_ASB_NO_PAPER	The paper is not set. The paper was ejected.	Set the paper.
ESCMC_ASB_PAPER_EJECT	Paper jam has occurred.	Remove the jammed paper.
ESCMC_ASB_COVER_OPEN	The device cover is open.	Close the cover.
ESCMC_ASB_FATAL_ERROR	Malfunction.	Send in the device for repair service.
ESCMC_ASB_HEAD_HOT_ERROR	The temperature of the head is high.	Power off and wait for a while.

Return Values

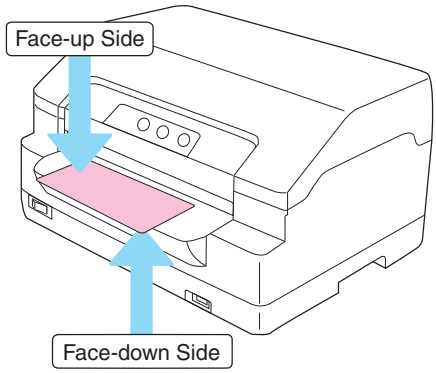
The following are errors that occur when calling a PLQ-22 API (return values of the PLQ-22 API functions). The nature of the errors varies depending on the PLQ-22 API.

Macro Definition (Constant)	Cause	Troubleshooting Action
ERR_TYPE	The parameter for nType is incorrect.	Specify the correct constant defined in the header file.
ERR_OPENED	The port is already in use by another application.	Exit the other application.
ERR_NO_PRINTER	The specified device cannot be found.	Check the device. (power, cable connection, etc)
ERR_NO_MEMORY	Insufficient memory.	Exit other applications or install more memory.
ERR_HANDLE	The handle value that specifies the device is invalid.	Check if the handle value was retrieved by BiOpenMonPrinter.
ERR_TIMEOUT	Timeout error.	If a timeout error continues to occur, check the device status.
ERR_ACCESS	Cannot read/write to the device. (The device's power is not on, bad cable connection, etc)	Check the device. (power, cable connection, etc)
ERR_PARAM	Parameter error.	Check if the value of the parameter is correct.
ERR_NOT_SUPPORT	The function of the API that was called is not provided.	Check the API that was called and the parameters used.
ERR_OFFLINE	The device is offline.	Check the status of the device.
ERR_NOT_FOUND	The specified data or module cannot be found.	• If a data cannot be found, check the parameter of the API that was executed. • If a module cannot be found, reinstall the scanner driver.
ERR_PAPERINSERT_TIMEOUT	Failed to insert the paper.	Check if the paper is correctly set in the device.
ERR_EXEC_FUNCTION	A process that was called by another API is running.	Wait for the process to end, and call the API again.
ERR_RESET	The device is being reset, and cannot be used.	Wait a while and call the API again.
ERR_MICR	Failed to read the MICR data.	Run the reading process again.
ERR_NOT_EXEC	The process specified by the API was not run.	Call the API again.
ERR_PAPER_JAM	A paper jam error has occurred.	Remove the paper.
ERR_PAPER_INSERT	Failed to insert the paper.	Set the paper into the device correctly.
ERR_UNLOCKED	The port is not locked.	Call BiLockPort to lock the port.
ERR_NET_CONNECTED	Failed to connect to the device on the network.	Check the specified IP address and host name.

Macro Definition (Constant)	Cause	Troubleshooting Action
ERR_MSRW_NODATA	The Magnetic Stripe data could not be detected.	Check to see if the insertion direction is set correctly.
ERR_MSRW_WRITE	• Failed to write the Magnetic Stripe data. • Failed to erase the Magnetic Stripe data.	• Run the write process again. • Run the erase process again.

PLQ-22CS/PLQ-22CSM and Reading Surface

In PLQ-22CS/PLQ-22CSM, the MSRW/MICR head is on the bottom side, and the print head is on the top side. Because of this, a check should be set face-down to be able to scan it.
In this manual, the reading surface on the MSRW/MICR head side is referred to as the “face-down side”, and reading surface on the print head side is referred to as “face-up side”.



TWAIN API Reference

This chapter explains the specifications of the TWAIN APIs (Triplets and Capabilities) used in PLQ-22CS/PLQ-22CSM.

TWAIN Specifications

The PLQ-22CS/PLQ-22CSM scanner driver is TWAIN Ver.2.0 compliant.
For details on TWAIN, visit the TWAIN Working Group website.

http://www.twain.org/

Supported TWAIN APIs

Triplet

NOTE

For details on triplets, refer to the TWAIN Specification from the TWAIN Working Group.

Data Group (DG)	Data Argument Type (DAT)	Massage (MSG)
DG_CONTROL	DAT_CALLBACK	MSG_REGISTER_CALLBACK
	DAT_CAPABILITY	MSG_GET
		MSG_GETCURRENT
		MSG_GETDEFAULT
		MSG_QUERYSUPPORT
		MSG_RESET
		MSG_RESETALL
		MSG_SET
	DAT_CUSTOMDSDATA	MSG_GET
		MSG_SET
	DAT_EVENT	MSG_PROCESSEVENT
	DAT_IDENTITY	MSG_CLOSEDS
		MSG_OPENDS
	DAT_PENDINGXFERS	MSG_ENDXFER
		MSG_GET
		MSG_RESET
	DAT_SETUPFILEXFER	MSG_GET
		MSG_GETDEFAULT
		MSG_RESET
		MSG_SET
	DAT_SETUPMEMXFER	MSG_GET
	DAT_STATUS	MSG_GET
	DAT_USERINTERFACE	MSG_DISABLEDS
		MSG_ENABLEDS
		MSG_ENABLEDSUIONLY
	DAT_XFERGROUP	MSG_GET

Data Group (DG)	Data Argument Type (DAT)	Massage (MSG)
DG_IMAGE	DAT_EXTIMAGEINFO	MSG_GET
	DAT_IMAGEFILEXFER	MSG_GET
	DAT_IMAGEINFO	MSG_GET
	DAT_IMAGELAYOUT	MSG_GET
		MSG_GETDEFAULT
		MSG_RESET
		MSG_SET
	DAT_IMAGEMEMXFER	MSG_GET
	DAT_IMAGENATIVEXFER	MSG_GET

Capability

The following capabilities are supported by PLQ-22CS/PLQ-22CSM:

Type	Capability	Function
Automatic Adjustments	ICAP_AUTOMATICBORDERDETECTION	Reports automatic border detection
	ICAP_AUTOMATICDESKEW	Corrects any skew in the scanned image
	ICAP_EPCS_TEXTEHNANCEMENT *	Removes the background from an image
	ICAP_EPCS_BLANKPAGESKIP *	Removes blank pages
	ICAP_EPCS_COLORENHANCEMENT *	Applies an RGB color enhancement filter
Barcode Detection	ICAP_BARCODEDETECTIONENABLED	Enables/disables barcode reading
	ICAP_BARCODESEARCHMODE	The direction in which barcodes are to be detected
	ICAP_BARCODESEARCHPRIORITIES	Specifies the barcodes to detect
	ICAP_SUPPORTEDBARCODETYPES	Retrieves the barcode types supported by the driver
Capability Negotiation Parameters	CAP_SUPPORTEDCAPS	Retrieves the capability values supported by PLQ-22CS/PLQ-22CSM
	CAP_CUSTOMDSDATA	Enables/disables support for DG_CONTROL / DAT_CUSTOMDSDATA
Device Parameters	CAP_DEVICEONLINE	Retrieves the connection status of the device
	ICAP_MINIMUMHEIGHT	Retrieves the minimum height of the capture size
	ICAP_MINIMUMWIDTH	Retrieves the minimum width of the capture size
	ICAP_PHYSICALHEIGHT	Retrieves the maximum height of the capture size
	ICAP_PHYSICALWIDTH	Retrieves the maximum width of the capture size
	ICAP_UNITS	Specifies the unit of the sizes set by the parameters

Type	Capability	Function
Endorser	CAP_ENDORSER	Specifies the starting number for printing the transaction number
	CAP_PRINTER	Specifies the endorsement printing method
	CAP_PRINTERENABLED	Executes endorsement
	CAP_PRINTERMODE	Specifies the operation mode of endorsement
	CAP_PRINTERSTRING	Specifies the string to endorse
	CAP_EPCS_PRINTERPOSITIONX *	Sets the X coordinate of the printing start position
	CAP_EPCS_PRINTERPOSITIONY *	Sets the Y coordinate of the printing start position
	CAP_EPCS_PRINTDIRECTION *	Specifies the printing direction
	CAP_EPCS_PRINTFONTSIZE *	Specifies the font size
Image Parameters for Acquire	ICAP_BRIGHTNESS	Specifies the brightness of an image
	ICAP_CONTRAST	Specifies the contrast of an image
	ICAP_GAMMA	Specifies the gamma value for gamma correction
	ICAP_ROTATION	Specifies the rotation of an image
	ICAP_THRESHOLD	Sets the threshold for simple binarization on brightness
	ICAP_EPCS_ADAPTIVETHRESHOLD *	Enables/disables adaptive binarization on brightness
Image Type	ICAP_BITDEPTH	Specifies the number of bits per one pixel
	ICAP_BITORDER	Specifies how the values are set when one byte contains multiple pixel data
	ICAP_LIGHTSOURCE	Specifies the color of the light source
	ICAP_PIXELFLAVOR	Specifies the pixel value of zero as black or white
	ICAP_PIXELTYPE	Specifies the pixel type
	ICAP_PLANARCHUNKY	Specifies the color data format
MICR	CAP_MICRENABLED	Enables/disables MICR data reading
	ICAP_EPCS_MICRFONT *	Specifies the MICR font to recognize
	ICAP_EPCS_MICRMETHOD *	Specifies the method of MICR recognition
OCR-A/B	ICAP_EPCS_OCRABENABLED *	Enables/disables OCR-A/B font scanning
	ICAP_EPCS_OCRABDIRECTION *	Specifies the direction of the OCR-A/B font to detect
	ICAP_EPCS_OCRABFONT *	Specifies the font to recognize
	ICAP_EPCS_OCRABFRAME *	Specifies the recognition target area
	ICAP_EPCS_OCRABSIDE *	Specifies the detection side
	ICAP_EPCS_OCRABSPACEHANDLING *	Specifies how the space characters are handled
Pages	ICAP_FRAMES	Specifies the scanning range
	ICAP_SUPPORTEDSIZES	Specifies fixed sizes of scanning range

Type	Capability	Function
Paper Handling	CAP_AUTOFEED	Sets continuous automatic paper feed by the auto feeder
	CAP_AUTOSCAN	Sets continuous scanning by the auto feeder
	CAP_CLEARPAGE	Ejects the paper in the paper path
	CAP_DUPLEX	Retrieves the support status of the duplex scanning feature
	CAP_DUPLEXENABLED	Enables/disables duplex scanning
	CAP_EPCS_DUPLEXORDER *	Specifies the scanning order for duplex scanning
	CAP_FEEDERENABLED	Retrieves the type of feeder
	CAP_FEEDERLOADED	Queries whether or not there is paper in the paper path
	CAP_PAPERDETECTABLE	Queries if there is support for detecting paper in the paper path
	CAP_EPCS_AUTOEJECT *	Enables/disables automatic ejection of the paper after the scan is complete
	CAP_EPCS_EJECTDIRECTION *	Specifies the direction of paper ejection initiated by CAP_CLEARPAGE
	CAP_EPCS_SCANSIDE *	Specifies which side to scan for single-side scanning
	CAP_EPCS_SCANDIRECTION *	Specifies the scanning direction
Resolution	ICAP_XNATIVERESOLUTION	Retrieves the optical resolution for the main scanning direction (X direction)
	ICAP_XRESOLUTION	Specifies the scanning resolution for the main scanning direction (X direction)
	ICAP_YNATIVERESOLUTION	Retrieves the optical resolution for the sub-scanning direction (Y direction)
	ICAP_YRESOLUTION	Specifies the scanning resolution for the sub-scanning direction (Y direction)
Transfers	CAP_XFERCOUNT	Specifies the number of image data that the application can accept
	ICAP_COMPRESSION	Specifies the compression format for memory transfer mode and file transfer mode
	ICAP_IMAGEFILEFORMAT	Specifies the file format for file transfers
	ICAP_XFERMECH	Specifies the transfer mode of image data.

Type	Capability	Function
User Interface	CAP_ENABLEDSUIONLY	Retrieves the support status of the GUI for specifying the values of the settings
	CAP_INDICATORS	Specifies whether or not to display the progress indicator when scanning UI-suppressed mode
	CAP_UICONTROLLABLE	Retrieves the support status of the UI-suppressed mode

* This capability is proprietary to PLQ-22CS/PLQ-22CSM.

Extended Capability

The following extended capabilities are supported by PLQ-22CS/PLQ-22CSM:

Extended Capability	Description
TWEI_MAGDATA	Retrieves the result of MICR recognition
TWEI_MAGTYPE	Retrieves the type of magnetic character data
TWEI_EPCS_OCRABDATA *	Retrieves the result of OCR recognition
TWEI_BARCODECOUNT	Retrieves the number of barcodes recognized
TWEI_BARCODEROTATION	Retrieves the rotation of the barcode that was recognized
TWEI_BARCODETEXTLENGTH	Retrieves the length of the string in the barcode that was recognized
TWEI_BARCODETEXT	Retrieves the barcode that was recognized
TWEI_BARCODEX	Retrieves the start position of the barcode that was recognized, in the main scanning direction (X direction)
TWEI_BARCODEY	Retrieves the start position of the barcode that was recognized, in the sub-scanning direction (Y direction)
TWEI_BARCODETYPE	Retrieves the type of barcode that was recognized

* This extended capability is proprietary to PLQ-22CS/PLQ-22CSM.

Reference for Triplets

This section explains the specifications of the triplets in PLQ-22CS/PLQ-22CSM.

DG_CONTROL / DAT_CALLBACK

Message: MSG_REGISTER_CALLBACK

Valid States: 4

Description

This triplet is sent from the application to the DSM to register the callback function in the application.

❏ For applications compatible with TWAIN 2.0:

Notifications can be received from the Data Source by linking TWAINDSM.DLL and using this triplet to register the callback function.

❏ For applications compatible with TWAIN 1.9 and earlier:

Do not use this triplet. Notifications from the Data Source can be received by sending DG_CONTROL / DAT_EVENT / MSG_PROCESSEVENT in State 5 or later.

DG_CONTROL / DAT_CAPABILITY

Message: MSG_GET
MSG_GETCURRENT
MSG_GETDEFAULT
MSG_QUERYSUPPORT
MSG_RESET
MSG_RESETALL
MSG_SET

Valid States: 4 through 7 (MSG_GET, MSG_GETCURRENT, MSG_GETDEFAULT)
4 (MSG_QUERYSUPPORT, MSG_RESET, MSG_RESETALL, MSG_SET)

Description

Retrieves and sets capability values. After opening the Data Source, each capability is set to their default values. Refer to "Reference for Capabilities" on page 50 and "Reference for Extended Capabilities" on page 99 for details.

DG_CONTROL / DAT_CUSTOMDSDATA

Message: MSG_GET
MSG_SET

Valid States: 4

Description

Allows you to retrieve and set scan conditions collectively.

- Retrieve (MSG_GET)

  Allows you to retrieve the memory area where the scan conditions are written in INI file format, and return it to pCustomData, which is a pointer to the TW_CUSTOMDSDATA structure.

- Set (MSG_SET)

  Saves the scan conditions in INI file format, and sends its contents to the Data Source.

> **NOTE** The scan conditions can be easily set by opening the GUI with DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDSUIONLY. Then after the GUI is closed, these scan conditions that were set on the GUI can be saved collectively using this triplet with MSG_GET.

DG_CONTROL / DAT_EVENT

Message: MSG_PROCESSEVENT

Valid States: 5 through 7

Description

❏ For applications compatible with TWAIN 1.9 and earlier:

After enabling the Data Source in State 5 or above, use this triplet to send all events to the Data Source.

❏ For applications compatible with TWAIN 2.0:

If the application links TWAINDSM.DLL and registers the callback function using DG_CONTROL / DAT_CALLBACK / MSG_REGISTER_CALLBACK, it is not necessary to send the events to the Data Source.

DG_CONTROL / DAT_IDENTITY

Message: MSG_CLOSEDS
MSG_OPENDS

Valid States: 4 (MSG_CLOSEDS)
3 (MSG_OPENDS)

Description

❑ MSG_OPENDS

Opens the Data Source specified by this triplet.

The Condition Codes for when "TWRC_FAILURE" is returned to the Return Code are as follows:

Condition Code	Description
TWCC_OPERATIONERROR	The PLQ-22CS/PLQ-22CSM is not connected, or the power is not on
TWCC_CHECKDEVICEONLINE	The PLQ-22CS/PLQ-22CSM is offline
TWCC_EPCS_NETWORKUNREACHABLE	When connected to a network, the PLQ-22CS/PLQ-22CSM cannot be found on the specified network.
TWCC_MAXCONNECTIONS	When connected to a network, the PLQ-22CS/PLQ-22CSM is used by another client computer.

❑ MSG_CLOSEDS

Closes the Data Source.

DG_CONTROL / DAT_PENDINGXFERS

Message: MSG_ENDXFER
MSG_GET
MSG_RESET

Valid States: 6 and 7 (MSG_ENDXFER)
4 through 7 (MSG_GET)
6 (MSG_RESET)

Description

Ends or aborts the image transfer.
Normally, DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER is sent after each image transfer is complete (i.e. each time TWRC_XFERDONE is received), and the value of pPendingXfers ->Count is checked to determine if the Data Source will continue sending more images.
If MSG_RESET is sent before all data are transferred (i.e. before TWRC_XFERDONE is returned) in memory transfer mode, the subsequent data are not transferred. In such a case, the application should destroy the previously received data, as the image data is incomplete.

DG_CONTROL / DAT_SETUPFILEXFER

Message: MSG_GET
MSG_GETDEFAULT
MSG_RESET
MSG_SET

Valid States: 4 through 6 (MSG_GET, MSG_GETDEFAULT, MSG_SET)
4 (MSG_RESET)

Description

Sets and retrieves the file information to be used for saving an image, before the data is transferred in file transfer mode.

The default file name is TWAIN.TMP, and it is saved in the current folder. If a file name including the file path is specified by the application, the file is generated and saved with the specified name.
If a file path is not specified, the file is saved in the current folder of the application.
If the specified file name already exists, the file is handled differently depending on the file format specified, as described below:

BMP, TIFF, and JPEG: The existing file is overwritten.

PDF:

- If the same file name was specified as the previous scan in a series of scans:
  The file is appended to the last page of the existing file.
- If a different file format or file name was specified, or if DG_CONTROL/DAT_IDENTITY/MSG_CLOSEDS was issued:
  The existing file is overwritten.

DG_CONTROL / DAT_SETUPMEMXFER

Message: MSG_GET

Valid States: 4 through 6

Description

This triplet retrieves the size of the memory that the Data Source needs for memory transfer.

DG_CONTROL / DAT_STATUS

Message: MSG_GET

Valid States: 2 through 6

Description

Retrieves the Condition Code of the last executed process. The Condition Code of the last executed process is cleared after this triplet or the next triplet is called.

DG_CONTROL / DAT_USERINTERFACE

Message: MSG_DISABLEDS
MSG_ENABLEDS
MSG_ENABLEDSUIONLY

Valid States: 5 (MSG_DISABLEDS)
4 (MSG_ENABLEDS, MSG_ENABLEDSUIONLY)

Description

❏ MSG_ENABLEDS

- UI Mode (TW_USERINTERFACE.ShowUI = TRUE)
  Opens the GUI. After the Scan button on the GUI is pressed, MSG_XFERREADY is sent to the application by the Data Source.
- UI-suppressed mode (TW_USERINTERFACE.ShowUI = FALSE)
  Transitions to State 5 without opening the GUI. Subsequent to this, the Data Source immediately sends MSG_XFERREADY to the application.

❏ MSG_DISABLEDS

- UI Mode (TW_USERINTERFACE.ShowUI = TRUE)
  Closes the GUI. Normally, when the Close button on the GUI is pressed, the Data Source sends MSG_CLOSEDSREQ to the application. Once the application receives MSG_CLOSEDSREQ, it calls MSG_DISABLEDS.
- UI-suppressed mode (TW_USERINTERFACE.ShowUI = FALSE)
  Transitions from State 5 to State 4.

❏ MSG_ENABLEDSUIONLY
Opens the GUI. After the scan conditions are set on the GUI and the GUI is closed, the application can save the scan conditions collectively by calling DG_CONTROL / DAT_CUSTOMDSDATA / MSG_GET. You can collectively set the saved scan conditions with DG_CONTROL / DAT_CUSTOMDSDATA / MSG_SET. Then you can scan in UI-suppressed mode, using the saved scan conditions.

DG_CONTROL / DAT_XFERGROUP

Message: MSG_GET

Valid States: 4 through 6

Description

Retrieves the data group supported by the Data Source. For PLQ-22CS/PLQ-22CSM, “DG_IMAGE” is returned to pXferGroup.

DG_IMAGE / DAT_EXTIMAGEINFO

Message: MSG_GET

Valid States: 7

Description

Retrieves extended image information. Refer to "Method of Retrieving Extended Image Information" on page 103 for details on the method of retrieval. Issue this triplet after the image transfer triplet DG_IMAGE / DAT_IMAGExxxXFER returns TWRC_DONE.

> **NOTE** If an unsupported extended information ID is specified for pExtImageInfo → Info[n].InfoID, TWRC_INFONOTSUPPORTED is returned to ImageInfo → Info[n].CondCode. In addition, if the extended information ID is supported but is not valid as a function, TWRC_DATANOTAVAILABLE is returned to pExtImageInfo → Info[n].CondCode.

DG_IMAGE / DAT_IMAGEFILEXFER

Message: MSG_GET

Valid States: 6

Description

Transfers an application image in file transfer mode. The file to store the image information should be specified in advance, using DG_CONTROL / DAT_SETUPFILEXFER / MSG_SET. If it is not specified, the image will be saved with a default file name.

About the Return Codes

- If a capture range is specified at a position that is out of range of the original document that was fed, “TWRC_CANCEL” is returned.
- The Condition Codes for when “TWRC_FAILURE” is returned is as follows:

Condition Code	Description
TWCC_PAPERJAM	Paper jam
TWCC_INTERLOCK	Cover is open
TWCC_OPERATIONERROR	Power is off, or cable is disconnected
TWCC_EPCS_PAPERTIMEOUT	Paper detection timeout

DG_IMAGE / DAT_IMAGEINFO

Message: MSG_GET

Valid States: 6 and 7

Description

Sends information about the image, such as its size, to the application before it is transferred.

> **CAUTION** When transferring multiple images, if automatic size detection or crop area was specified in the images, each image may have a different image size. For this reason, for memory transfer, be sure to secure the necessary memory on the application side, by retrieving image information using this triplet before each image is transferred.

DG_IMAGE / DAT_IMAGELAYOUT

Message: MSG_GET
MSG_GETDEFAULT
MSG_RESET
MSG_SET

Valid States: 4 through 6 (MSG_GET, MSG_GETDEFAULT)
4 (MSG_RESET, MSG_SET)

Description

Specifies the capture range. Specifying the capture range with this triplet overwrites the value of ICAP_FRAME. Issuing MSG_SET causes FrameNumber, PageNumber, and DocumentNumber of pImageLayout to be ignored. Issuing MSG_GET stores “1” to all of these values.

DG_IMAGE / DAT_IMAGEMEMXFER

Message: MSG_GET

Valid States: 6 and 7

Description

Transfers an image to the application in memory transfer mode. Since the image is transferred in blocks, if the image size is larger than one block, it is divided and transferred in parts. When the last block is transferred, "TWRC_XFERDONE" is returned to the Return Code.
If ICAP_PIXELTYPE is "TWPT_RGB", one pixel of the image is filled in R, G, B order. The data are transferred per line in 4-byte alignment.

About the Return Codes

❏ If a capture range is specified at a position that is out of range of the original document that was fed, "TWRC_CANCEL" is returned.

❏ The Condition Codes for when "TWRC_FAILURE" is returned is as follows:

Condition Code	Description
TWCC_PAPERJAM	Paper jam
TWCC_INTERLOCK	Cover is open
TWCC_OPERATIONERROR	Power is off, or cable is disconnected
TWCC_EPCS_PAPERTIMEOUT	Paper detection timeout

DG_IMAGE / DAT_IMAGENATIVEXFER

Message: MSG_GET

Valid States: 6 and 7

Description

Transfers an image to the application in Native transfer mode.
The image is stored as a Windows Device-Independent Bitmap format whose memory was secured by the Data Source, and its memory handle is transferred with pHandle. This memory needs to be released on the application side.

About the Return Codes

❏ If a capture range is specified at a position that is out of range of the original document that was fed, "TWRC_CANCEL" is returned.

❏ The Condition Codes for when "TWRC_FAILURE" is returned is as follows:

Condition Code	Description
TWCC_PAPERJAM	Paper jam
TWCC_INTERLOCK	Cover is open
TWCC_OPERATIONERROR	Power is off, or cable is disconnected
TWCC_EPCS_PAPERTIMEOUT	Paper detection timeout

Reference for Capabilities

This section explains the specifications of capabilities in PLQ-22CS/PLQ-22CSM.

ICAP_AUTOMATICBORDERDETECTION

Reports automatic border detection.

Values

Type:	TW_BOOL
Default Value:	TRUE
Allowed Values:	TRUE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

Description

The Automatic Border Detection function is always enabled for PLQ-22CS/PLQ-22CSM.

ICAP_AUTOMATICDESKEW

Detects and corrects any skew in the scanned original document.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_EPCS_TEXTEHNANCEMENT

Removes the background from the document so that the text in the image data is easier to read.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_EPCS_BLANKPAGESKIP

Removes blank pages.

> **NOTE** This capability is enabled when the transfer mode (ICAP_XFERMECH) is in file transfer mode (TWSX_FILE) and the file format (ICAP_IMAGEFILEFORMAT) is PDF (TWFF_PDF).

Values

Type:	TW_UINT16	
Default Value:	TWEPCS_BS_OFF	
Allowed Values:	TWEPCS_BS_OFF:	Blank page removal disabled
	TWEPCS_BS_LOW:	Blank page removal level - Low
	TWEPCS_BS_MEDIUM:	Blank page removal level - Medium
	TWEPCS_BS_HIGH:	Blank page removal level - High
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	

> **NOTE** Setting the level to high (TWEPCS_BS_LOW → TWEPCS_BS_HIGH) allows you to detect blank pages even if a page is dirty.

Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE
	TW_ENUMERATION

ICAP_EPCS_COLORENHANCEMENT

Applies an RGB color enhancement filter.

NOTE

- This capability is enabled when the pixel type (ICAP_PIXELTYPE) is set to Black and White (TWPT_BW) or Grayscale (TWPT_GRAY).
- If a value other than TWEPCS_CE_NONE is specified for this capability, ICAP_LIGHTSOURCE will be set to TWLS_WHITE.

Values

Type:	TW_UINT16	
Default Value:	TWEPCS_CE_NONE	
Allowed Values:	TWEPCS_CE_NONE:	Color enhancement filter disabled
	TWEPCS_CE_RED:	Applies a red color enhancement filter
	TWEPCS_CE_GREEN:	Applies a green color enhancement filter
	TWEPCS_CE_BLUE:	Applies a blue color enhancement filter
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	
Containers for MSG_GET:	TW_ENUMERATION	
Containers for MSG_SET:	TW_ONEVALUE	
	TW_ENUMERATION	

ICAP_BARCODEDETECTIONENABLED

Enables/disables barcode reading.

NOTE

The read barcode is retrieved by TWEI_BARCODETEXT (page 101), which is an extended capability.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET
	MSG_GETCURRENT
	MSG_GETDEFAULT
	MSG_SET
	MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE
	TW_ENUMERATION

ICAP_BARCODESEARCHMODE

Specifies the direction in which barcodes are to be detected.

Values

Type:	TW_UINT16
Default Value:	TWBD_HORZ
Allowed Values:	TWBD_HORZ TWBD_VERT TWBD_HORZVERT
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

Description

Limitations When Detecting Multiple Barcodes

❏ It is recommended that you set this capability to "TWBD_HORZ" and scan horizontally to detect the barcodes. Scanning the barcode vertically lowers the recognition accuracy.

❏ Up to five barcodes can be detected simultaneously.

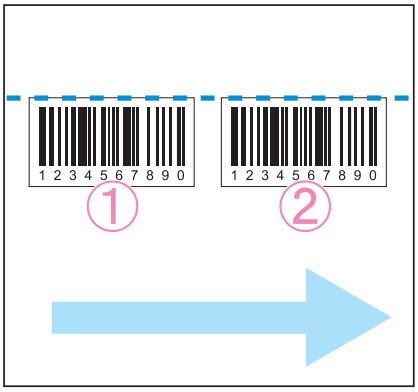
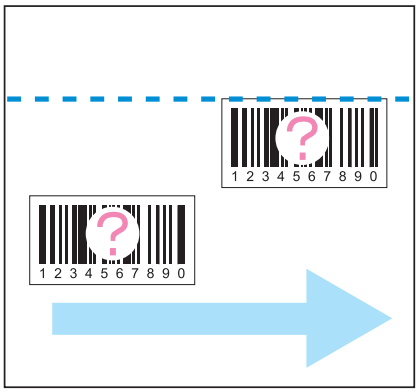
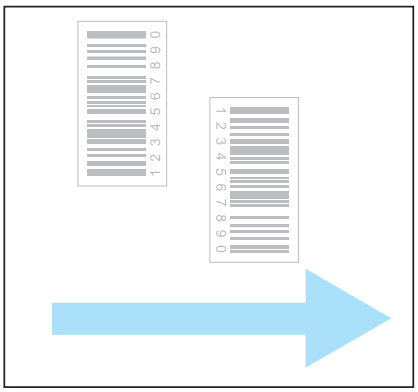
❏ A quiet zone is required according to each barcode standard.

❏ To detect multiple barcodes, the following conditions must be met within the specified area:

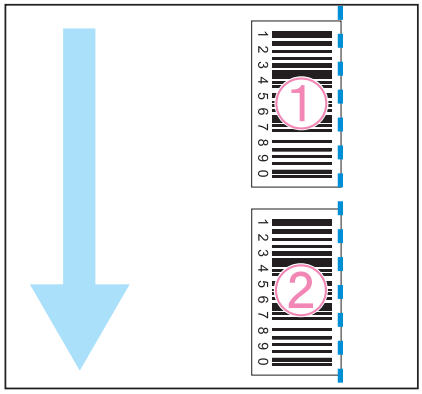
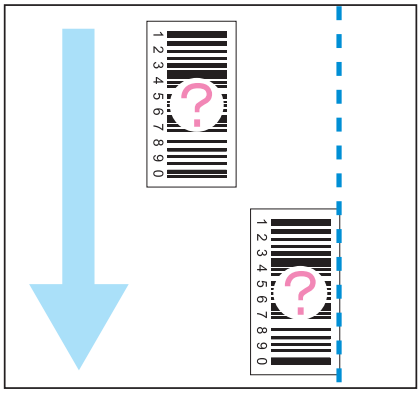
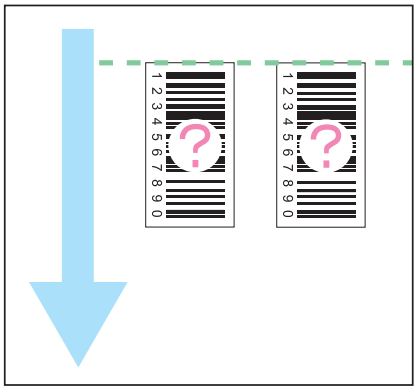
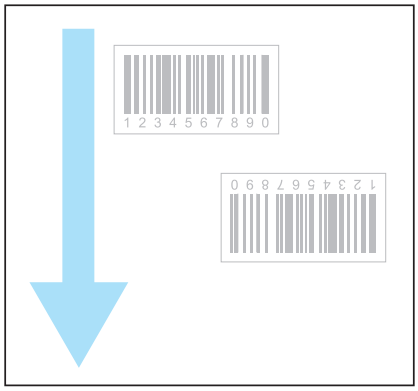
- Both barcodes are of the same barcode standard.
- The quiet zones of the barcodes do not overlap.
- Both barcodes have the same height (not including the HRI characters), and are both printed from the same position in the height-wise direction.

How Scan Direction Affects Multiple Barcode Detection:

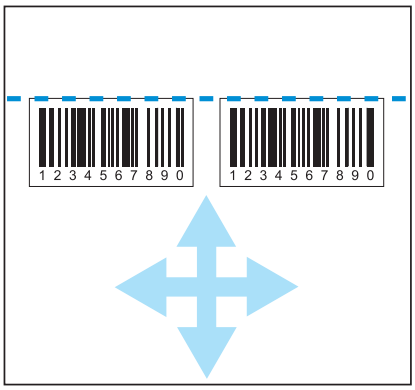
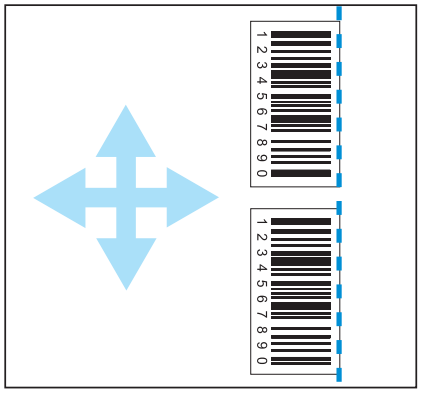
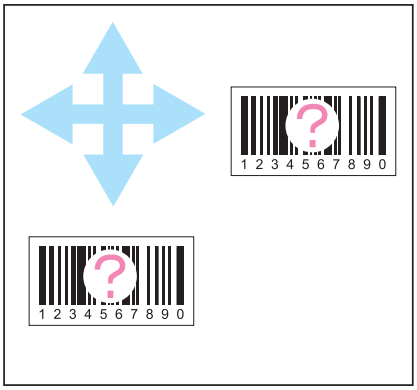
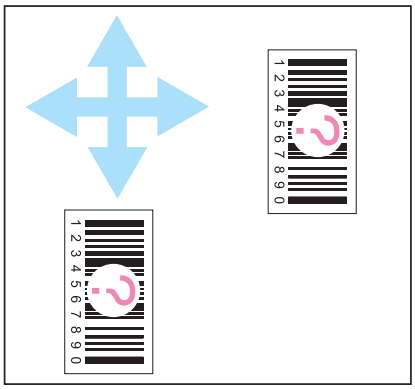
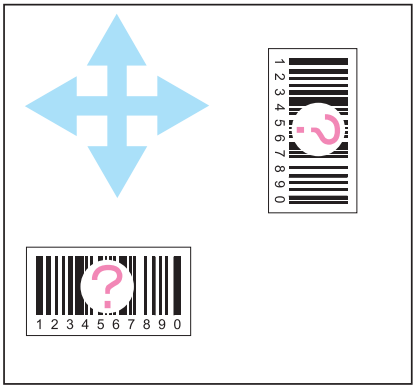
TWBD_HORZ: Detecting in horizontal direction

Case 1: Start positions placed vertically	Case 2: Start positions not placed vertically
	
The barcodes are detected in order, starting from the left edge.	Only one barcode is detected.
Case 3: Start positions placed horizontally	**Case 4: Both perpendicular to the scan direction**
	
Only one barcode is detected.	No barcode is detected.

TWBD_VERT: Detecting in perpendicular direction

Case 1: Start positions placed vertically	Case 2: Start positions not placed vertically
	
The barcodes are detected in order, starting from the top edge.	Only one barcode is detected.
Case 3: Start positions placed horizontally	**Case 4: Both perpendicular to the scan direction**
	
Only one barcode is detected.	No barcode is detected.

TWBD_HORZVERT: Detecting in horizontal and vertical directions

Case 1: Both barcodes placed horizontally with same vertical start positions	Case 2: Both barcodes placed vertically with same horizontal start positions
	
The barcodes are detected in order, starting from the left edge.	The barcodes are detected in order, starting from the top edge.
Case 3: Both barcodes placed horizontally with different vertical start positions	**Case 4: Both barcodes placed vertically with different horizontal start positions**
	
Only one barcode is detected.	Only one barcode is detected.
Case 5: A horizontal barcode and a perpendicular barcode coexist	
	
Barcodes are detected in horizontal direction, then in vertical direction.	

ICAP_BARCODESEARCHPRIORITIES

Specifies the barcodes to detect. You can speed up barcode detection by setting this capability.

> **NOTE** UPC-A, UPC-E, EAN8, and EAN13 all comply to the same barcode standard. When you select one of these barcodes, all barcodes of the same standard (UPC-A, UPC-E, EAN8/EAN13) are detected.

Values

Type:	TW_UINT16	
Default Value:	List of Allowed values	
Allowed Values:	TWBT_3OF9:	Code39
	TWBT_2OF5INTERLEAVED:	ITF
	TWBT_CODE128:	Code128
	TWBT_CODABAR:	Codabar
	TWBT_UPCA:	UPC-A
	TWBT_UPCE:	UPC-E
	TWBT_EAN8:	EAN8
	TWBT_EAN13:	EAN13
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	
Containers for MSG_GET:	TW_ARRAY	
Containers for MSG_SET:	TW_ARRAY	

ICAP_SUPPORTEDBARCODETYPES

Retrieves the barcode types supported by the driver.

Values

Type:	TW_UINT16	
Default Value:	No Default	
Allowed Values:	TWBT_3OF9:	Code39
	TWBT_2OF5INTERLEAVED:	ITF
	TWBT_CODE128:	Code128
	TWBT_CODABAR:	Codabar
	TWBT_UPCA:	UPC-A
	TWBT_UPCE:	UPC-E
	TWBT_EAN8:	EAN8
	TWBT_EAN13:	EAN13
Allowed Operations:	MSG_GET	
Containers for MSG_GET:	TW_ARRAY	
Containers for MSG_SET:	Not Allowed	

CAP_SUPPORTEDCAPS

Retrieves the capability values supported by PLQ-22CS/PLQ-22CSM.

Values

Type:	TW_UINT16
Default Value:	No Default
Allowed Values:	Any supported capabilities
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ARRAY
Containers for MSG_SET:	Not Allowed

CAP_CUSTOMDSDATA

Enables/disables support for DG_CONTROL / DAT_CUSTOMDSDATA.
Since PLQ-22CS/PLQ-22CSM supports DG_CONTROL / DAT_CUSTOMDSDATA, “TRUE” is always returned.

Values

Type:	TW_BOOL
Default Value:	TRUE
Allowed Values:	TRUE
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

CAP_DEVICEONLINE

Retrieves the connection status of the device. If the device is online, “TRUE” is returned.

Values

Type:	TW_BOOL
Default Value:	TRUE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

ICAP_MINIMUMHEIGHT

Retrieves the minimum height of the capture size.

> NOTE
>
> This capability retrieves the size in the unit specified in ICAP_UNITS (page 63).

Values

Type:	TW_FIX32
Default Value:	No Default
Allowed Values:	0 to 32767
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

ICAP_MINIMUMWIDTH

Retrieves the minimum width of the capture size.

> NOTE
>
> This capability retrieves the size in the unit specified in ICAP_UNITS (page 63).

Values

Type:	TW_FIX32
Default Value:	No Default
Allowed Values:	0 to 32767
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

ICAP_PHYSICALHEIGHT

Retrieves the maximum height of the capture size.

> **NOTE** This capability retrieves the size in the unit specified in ICAP_UNITS (page 63).

Values

Type:	TW_FIX32
Default Value:	No Default
Allowed Values:	0 to 65535
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

ICAP_PHYSICALWIDTH

Retrieves the maximum width of the capture size.

> **NOTE** This capability retrieves the size in the unit specified in ICAP_UNITS (page 63).

Values

Type:	TW_FIX32
Default Value:	No Default
Allowed Values:	0 to 65535
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

ICAP_UNITS

Specifies the unit of sizes set by the parameters.

Values

Type:	TW_UINT16	
Default Value:	TWUN_INCHES	
Allowed Values:	TWUN_INCHES:	inches
	TWUN_CENTIMETERS:	centimeters
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	
Containers for MSG_GET:	TW_ENUMERATION	
Containers for MSG_SET:	TW_ONEVALUE	
	TW_ENUMERATION	

CAP_ENDORSER

Specifies the starting number for printing the transaction number.

> **NOTE** Specifying the value in a format that uses special enclosing characters “<>” allows you to print in serial numbers. Refer to "CAP_PRINTERSTRING" on page 66 for details.

Values

Type:	TW_UINT32
Default Value:	0
Allowed Values:	Any Value
Allowed Operations:	MSG_GET
	MSG_GETCURRENT
	MSG_GETDEFAULT
	MSG_SET
	MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE

CAP_PRINTER

Specifies the endorsement printing method.

> **CAUTION** Real endorsement is not performed if other than “0 (zero)” (no rotation) has been specified in ICAP_ROTATION (page 71).

Values

Type:	TW_UINT16
Default Value:	TWPR_EPCS_ENDORSERBOTTOMELECTRIC
Allowed Values:	TWPR_ENDORSERBOTTOMBEFORE Face-up side real endorsement (scan before printing) TWPR_ENDORSERBOTTOMAFTER Face-up side real endorsement (scan after printing) TWPR_EPCS_ENDORSERTOPELECTRIC Face-down side electronic endorsement TWPR_EPCS_ENDORSERBOTTOMELECTRIC Face-up side electronic endorsement
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_PRINTERENABLED

Executes endorsement.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_PRINTERMODE

Specifies the operation mode of endorsement. For PLQ-22CS/PLQ-22CSM, only specify "TWPM_SINGLESTRING", which prints a single string.

Values

Type:	TW_UINT16
Default Value:	TWPM_SINGLESTRING
Allowed Values:	TWPM_SINGLESTRING
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_PRINTERSTRING

Specifies the string to endorse.

> **CAUTION** Specify within the number of characters that the paper can contain.

Values

Type:	TW_STR255
Default Value:	No Default
Allowed Values:	Any String
Allowed Operations:	MSG_GET MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE

Description

The characters specified in this capability will be printed for endorsement.

The following characters enclosed in "<>" allow the transaction numbers specified in CAP_ENDORSER to be printed in serial numbers.

- <0000>:
  The number of "0"s indicates the number of digits to print (1 to 9 digits). If the specified transaction number does not use all digits, the number will have leading zeros (e.g. "002").
- <xxxx>:
  The number of "x"s indicates the number of digits to print (1 to 9 digits). If the specified transaction number does not use all digits, the number will have leading space characters (e.g. " 2").
- <llll>:
  The number of "l"s (small case "L") indicates the number of digits to print (1 to 9 digits). If the specified transaction number does not use all digits, the number will be printed with no leading characters (e.g. "2").

> **NOTE** The special characters "<>" are not printed. If you want to print them, enclose them in ampersands ("&<>&").

CAP_EPCS_PRINTERPOSITIONX

Sets the X coordinate of the printing start position.

NOTE

This capability retrieves the size in the unit specified in ICAP_UNITS (page 63).

Values

Type:	TW_FIX32
Default Value:	0.119 inch (0.3 cm)
Allowed Values:	Any Value
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE

CAP_EPCS_PRINTERPOSITIONY

Sets the Y coordinate of the printing start position.

NOTE

This capability retrieves the size in the unit specified in ICAP_UNITS (page 63).

Values

Type:	TW_FIX32
Default Value:	0.079 inch (0.2 cm)
Allowed Values:	Any Value
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE

CAP_EPCS_PRINTDIRECTION

Specifies the printing direction.

> **NOTE** This capability is enabled when the endorsement printing method (CAP_PRINTER) is Face-up side electronic endorsement (TWPR_EPCS_ENDORSERTOPELECTRIC) or Face-down side electronic endorsement (TWPR_EPCS_ENDORSERBOTTOMELECTRIC).

Values

Type:	TW_UINT16	
Default Value:	TWEPCS_PD_LEFTRIGHT	
Allowed Values:	TWEPCS_PD_LEFTRIGHT:	Left to right
	TWEPCS_PD_RIGHTLEFT:	Right to left
	TWEPCS_PD_TOPBOTTOM:	Top to bottom
	TWEPCS_PD_BOTTOMTOP:	Bottom to top
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	
Containers for MSG_GET:	TW_ENUMERATION	
Containers for MSG_SET:	TW_ONEVALUE	
	TW_ENUMERATION	

CAP_EPCS_PRINTFONTSIZE

Specifies the font size for endorsement printing.

> **NOTE** This capability is enabled when the endorsement printing method (CAP_PRINTER) is Face-up side electronic endorsement (TWPR_EPCS_ENDORSERTOPELECTRIC) or Face-down side electronic endorsement (TWPR_EPCS_ENDORSERBOTTOMELECTRIC).

Values

Type:	TW_UINT32
Default Value:	10
Allowed Values:	8 to 72
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_RANGE
Containers for MSG_SET:	TW_ONEVALUE TW_RANGE

ICAP_BRIGHTNESS

Specifies the brightness of an image.

Values

Type:	TW_FIX32
Default Value:	0
Allowed Values:	-1000 to 1000
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_RANGE
Containers for MSG_SET:	TW_ONEVALUE TW_RANGE

ICAP_CONTRAST

Specifies the contrast of an image.

Values

Type:	TW_FIX32
Default Value:	0
Allowed Values:	-1000 to 1000, Step 1
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_RANGE
Containers for MSG_SET:	TW_ONEVALUE TW_RANGE

ICAP_GAMMA

Specifies the gamma correction value.

Values

Type:	TW_FIX32
Default Value:	2.2
Allowed Values:	1.0, 1.8, 2.2
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_ROTATION

Specifies the rotation of an image.

NOTE

- An image on the face-down side rotates counter-clockwise, and an image on the face-up side rotates clockwise.
- If a value outside of the range shown for Allowed Values is set, the message TWRC_FAILURE / TWCC_BADVALUE will appear.

Values

Type:	TW_FIX32	
Default Value:	0	
Allowed Values:	0:	No rotation
	90, -270:	Rotate 90 degrees
	270, -90:	Rotate 270 degrees
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	
Containers for MSG_GET:	TW_RANGE	
Containers for MSG_SET:	TW_ONEVALUE	
	TW_RANGE	

ICAP_THRESHOLD

Sets the threshold for simple binarization on brightness.

NOTE

- This capability is enabled when the pixel type (ICAP_PIXELTYPE) is set to Black and White (TWPT_BW).
- If ICAP_EPCS_ADAPTIVETHRESHOLD or ICAP_EPCS_TEXTEHNANCEMENT is set to "TRUE", the setting for this capability is ignored.

Values

Type:	TW_FIX32
Default Value:	128
Allowed Values:	0 to 255
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_RANGE
Containers for MSG_SET:	TW_ONEVALUE TW_RANGE

ICAP_EPCS_ADAPTIVETHRESHOLD

Enables/disables adaptive binarization on brightness.
Especially for binarization of a check image, it is recommended that you enable this capability (i.e. set it to "TRUE") to remove the background so that the characters are easier to see.

NOTE

- This capability is enabled when the pixel type (ICAP_PIXELTYPE) is set to Black and White (TWPT_BW).
- If ICAP_EPCS_ADAPTIVETHRESHOLD and ICAP_EPCS_TEXTEHNANCEMENT are both set to "TRUE", priority is given to ICAP_EPCS_TEXTEHNANCEMENT.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_BITDEPTH

Specifies the number of bits per one pixel.

Values

Type:	TW_UINT16
Default Value:	Varies with the setting for ICAP_PIXELTYPE.
Allowed Values:	Varies with the setting for ICAP_PIXELTYPE.
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

Description

The value of this capability is determined by the pixel type (ICAP_PIXELTYPE). See below:

Pixel type (ICAP_PIXELTYPE)	ICAP_BITDEPTH value
Black and White (TWPT_BW)	1
Grayscale (TWPT_GRAY)	8
Color (TWPT_RGB)	24

ICAP_BITORDER

Specifies how the values are set when one byte contains multiple pixel data.
For PLQ-22CS/PLQ-22CSM, only specify “TWBO_MSBFIRST”, in which the upper left pixel is filled from the Most Significant Bit.

Values

Type:	TW_UINT16
Default Value:	TWBO_MSBFIRST
Allowed Values:	TWBO_MSBFIRST
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_LIGHTSOURCE

Specifies the color of the light source.

NOTE

- This capability is enabled when the pixel type (ICAP_PIXELTYPE) is set to Black and White (TWPT_BW) or Grayscale (TWPT_GRAY).
- When a value other than white (TWLS_WHITE) is specified for this capability, the RGB color enhancement filter setting (ICAP_EPCS_COLORENHANCEMENT) becomes disabled (TWEPCS_CE_NONE).
- TWLS_RED, TWLS_GREEN, and TWLS_BLUE in this capability correspond to the [Drop out] menu in EPSON Scan Business.

CAUTION

If a value other than white (TWLS_WHITE) is selected, barcode recognition will not get executed.

Values

Type:	TW_UINT16
Default Value:	TWLS_WHITE
Allowed Values:	TWLS_WHITE: White light (composite image from RGB3 color light source)
	TWLS_RED: Red light
	TWLS_GREEN: Green light
	TWLS_BLUE: Blue light
	TWLS_IR: Infrared light
Allowed Operations:	MSG_GET
	MSG_GETCURRENT
	MSG_GETDEFAULT
	MSG_SET
	MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE
	TW_ENUMERATION

ICAP_PIXELFLAVOR

Specifies the pixel value "0" as black or white. For PLQ-22CS/PLQ-22CSM, only specify "TWPF_CHOCOLATE" (black).

Values

Type:	TW_UINT16
Default Value:	TWPF_CHOCOLATE
Allowed Values:	TWPF_CHOCOLATE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_PIXELTYPE

Specifies the pixel type.

> **CAUTION** Even if this capability is set to Black and White (TWPT_BW), if the format of the saved file in file transfer mode is JPEG, it is saved in grayscale.

Values

Type:	TW_UINT16
Default Value:	TWPT_RGB
Allowed Values:	TWPT_BW: Black and White TWPT_GRAY: Grayscale TWPT_RGB: Color
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_PLANARCHUNKY

Specifies the color data format. For PLQ-22CS/PLQ-22CSM, only specify Chunky Mode (TWPC_CHUNKY).

Values

Type:	TW_UINT16
Default Value:	TWPC_CHUNKY
Allowed Values:	TWPC_CHUNKY
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_MICRENABLED

Enables/disables MICR data reading.

NOTE

- This capability is proprietary to PLQ-22 CSM.
- The read MICR data can be retrieved with TWEI_MAGDATA, which is an extended capability.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_EPCS_MICRFONT

Specifies the MICR font to recognize.

> NOTE This capability is proprietary to PLQ-22 CSM.

Values

Type:	TW_UINT16	
Default Value:	TWEPCS_MF_E13B	
Allowed Values:	TWEPCS_MF_E13B:	E13B
	TWEPCS_MF_CMC7_ALPHANUM:	CMC7 (Alphanumeric)
	TWEPCS_MF_CMC7_NUM:	CMC7 (Numeric)
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	
Containers for MSG_GET:	TW_ENUMERATION	
Containers for MSG_SET:	TW_ONEVALUE	
	TW_ENUMERATION	

ICAP_EPCS_MICRMETHOD

Specifies the MICR recognition method.

> NOTE This capability is proprietary to PLQ-22 CSM.

Values

Type:	TW_UINT16	
Default Value:	TWEPCS_MM_MICRANDOCR	
Allowed Values:	TWEPCS_MM_MICRONLY:	Magnetic waveform
	TWEPCS_MM_MICRANDOCR:	Magnetic waveform and optical recognition
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	
Containers for MSG_GET:	TW_ENUMERATION	
Containers for MSG_SET:	TW_ONEVALUE	
	TW_ENUMERATION	

ICAP_EPCS_OCRABENABLED

Enables/disables OCR-A/B font reading.

> **NOTE** The results of OCR recognition can be retrieved with TWEI_EPCS_OCRABDATA, which is an extended capability.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_EPCS_OCRABDIRECTION

Specifies the direction of the OCR-A/B font to detect.

Values

Type:	TW_UINT16
Default Value:	TWEPCS_OD_LEFTRIGHT
Allowed Values:	TWEPCS_OD_LEFTRIGHT: Left to right TWEPCS_OD_TOPBOTTOM: Top to bottom (clockwise) TWEPCS_OD_RIGHTLEFT: Right to left TWEPCS_OD_BOTTOMTOP: Bottom to top (clockwise)
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_EPCS_OCRABFONT

Specifies the font to recognize.

Values

Type:	TW_UINT16
Default Value:	TWEPCS_OF_OCR_FONT_OCRA_NUM
Allowed Values:	TWEPCS_OF_OCR_FONT_OCRA_NUM : OCR-A font and numbers TWEPCS_OF_OCR_FONT_OCRA_ALPHA : OCR-A font and alphabets TWEPCS_OF_OCR_FONT_OCRA_ALPHANUM : OCR-A font and alpha numerals TWEPCS_OF_OCR_FONT_OCRA_ALPHANUM_WOOH : OCR-A font and alpha numerals (excluding OH) TWEPCS_OF_OCR_FONT_OCRA_ALPHANUM_WOZERO : OCR-A font and alpha numerals (excluding zero) TWEPCS_OF_OCR_FONT_OCRA_SYSNUM : OCR-A font and numbers/symbols (excluding the + sign) TWEPCS_OF_OCR_FONT_OCRB_NUM : OCR-B font and numbers TWEPCS_OF_OCR_FONT_OCRB_ALPHA : OCR-B font and alphabets TWEPCS_OF_OCR_FONT_OCRB_ALPHANUM : OCR-B font and alpha numerals TWEPCS_OF_OCR_FONT_OCRB_ALPHANUM_WOOH : OCR-B font and alpha numerals (excluding OH) TWEPCS_OF_OCR_FONT_OCRB_ALPHANUM_WOZERO : OCR-B font and alpha numerals (excluding zero) TWEPCS_OF_OCR_FONT_OCRB_SYSNUM : OCR-B font and numbers/symbols (excluding the + sign)
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_EPCS_OCRABFRAME

Specifies the detection target area.

The area to specify is different depending on the direction in which the OCR-A/B fonts are detected (ICAP_EPCS_OCRABDIRECTION).

- For TWEPCS_OD_LEFTRIGHT and TWEPCS_OD_RIGHTLEFT:
  Set the height of the area at 1 to 27 mm
- For TWEPCS_OD_TOPBOTTOM and TWEPCS_OD_BOTTOMTOP:
  Set the width of the area at 1 to 27 mm

OCR-A/B font recognition is possible for 1 to 2 lines.

NOTE

- This capability is applied to the side specified in ICAP_EPCS_OCRABSIDE (page 83).
- For the maximum value of the scan range, refer to PLQ-22CS/PLQ-22CSM User's Guide.
- This capability depends on the unit specified in ICAP_UNITS (page 63).

Values

Type:	TW_FRAME
Default Value:	Maximum value of the scan range
Default Value:	Maximum value of the scan range
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE

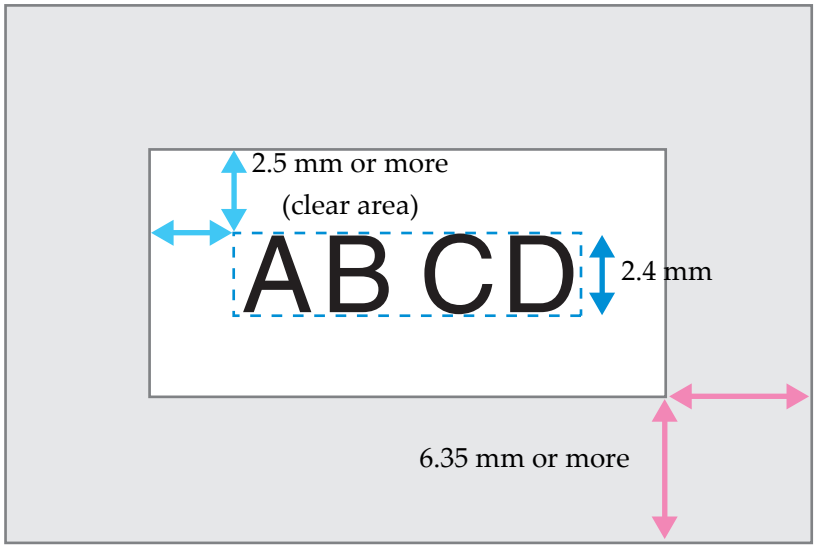
Limitations in OCR-A/B Font Detection

❏ Make sure that the area specified with this capability does not contain any characters or patterns other than the characters to be detected. Otherwise, misrecognition may occur.

❏ The only supported font height is size I (width: 1.4 mm, height: 2.4 mm).

❏ The margin of 2.5 mm or more around the OCR-A/B font is called the “clear area”, and must not contain any print. However, do not include the clear area in the specified area.

❏ There should be at least 6.35 mm space between the clear area and the edge of the paper.



ICAP_EPCS_OCRABSIDE

Specifies which side of paper to scan OCR-A/B font.

Values

Type:	TW_UINT16	
Default Value:	TWEPCS_OS_FRONT	
Allowed Values:	TWEPCS_OS_FRONT:	Face-down side
	TWEPCS_OS_BACK:	Face-up side
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	
Containers for MSG_GET:	TW_ENUMERATION	
Containers for MSG_SET:	TW_ONEVALUE	
	TW_ENUMERATION	

ICAP_EPCS_OCRABSPACEHANDLING

Specifies the handling of space characters in the OCR recognition result.

Values

Type:	TW_UINT16	
Default Value:	TWEPCS_SH_SPACEENABLE	
Allowed Values:	TWEPCS_SH_SPACEENABLE:	Include space characters
	TWEPCS_SH_SPACEDISABLE:	Do not include space characters
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	
Containers for MSG_GET:	TW_ENUMERATION	
Containers for MSG_SET:	TW_ONEVALUE	
	TW_ENUMERATION	

ICAP_FRAMES

Specifies the scanning range.

NOTE

- For the maximum value of the scan range, refer to PLQ-22CS/PLQ-22CSM User's Guide.
- This capability depends on the unit specified in ICAP_UNITS (page 63).

Values

Type:	TW_FRAME
Default Value:	Maximum value of the scan range
Default Value:	Maximum value of the scan range
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE

ICAP_SUPPORTEDSIZES

Specifies fixed sizes of scanning range.

Values

Type:	TW_UINT16
Default Value:	TWSS_NONE
Allowed Values:	TWSS_NONE TWSS_USLETTER TWSS_USLEGAL TWSS_A5 TWSS_A4
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_AUTOFEED

Sets continuous automatic paper feed by the auto feeder. For PLQ-22CS/PLQ-22CSM, only specify "FALSE".

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_AUTOSCAN

Sets continuous scanning by the auto feeder. For PLQ-22CS/PLQ-22CSM, only specify “FALSE”.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_CLEARPAGE

Ejects the paper in the paper path.

NOTE

- MSG_SET of this capability is valid in States 5 to State 6.
- MSG_GET of this capability returns “FALSE”.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_DUPLEX

Retrieves the support status of the duplex scanning feature. For PLQ-22CS/PLQ-22CSM, only "TWDX_1PASSDUPLEX" is returned.

Values

Type:	TW_UINT16
Default Value:	No Default
Allowed Values:	TWDX_1PASSDUPLEX
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

CAP_DUPLEXENABLED

Enables/disables duplex scanning.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	TRUE: Duplex scanning FALSE: Single-side scanning
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_EPCS_DUPLEXORDER

Specifies the scanning order for duplex scanning.

Values

Type:	TW_UINT16	
Default Value:	TWEPCS_DO_UPPERFIRST	
Allowed Values:	TWEPCS_DO_LOWERFIRST:	Face-down side is scanned first
	TWEPCS_DO_UPPERFIRST:	Face-up side is scanned first
Allowed Operations:	MSG_GET	
	MSG_GETCURRENT	
	MSG_GETDEFAULT	
	MSG_SET	
	MSG_RESET	
Containers for MSG_GET:	TW_ENUMERATION	
Containers for MSG_SET:	TW_ONEVALUE	
	TW_ENUMERATION	

CAP_FEEDERENABLED

Queries whether or not the type of feeder is a sheet feeder type. For PLQ-22CS/PLQ-22CSM, only "TRUE" is returned.

Values

Type:	TW_BOOL
Default Value:	TRUE
Allowed Values:	TRUE
Allowed Operations:	MSG_GET
	MSG_GETCURRENT
	MSG_GETDEFAULT
	MSG_SET
	MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE

CAP_FEEDERLOADED

Queries whether or not the paper is loaded in the paper path.

Values

Type:	TW_BOOL
Default Value:	FALSE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

CAP_PAPERDETECTABLE

Queries if there is support for detecting paper in the paper path. For PLQ-22CS/PLQ-22CSM, only "TRUE" is returned.

Values

Type:	TW_BOOL
Default Value:	No Default
Allowed Values:	TRUE
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

CAP_EPCS_AUTOEJECT

Enables/disables automatic ejection of the paper after the scan is complete.

> **NOTE**
>
> If this capability is set to “FALSE”, the paper is not ejected after the scan is complete. In this case, eject the paper with CAP_CLEARPAGE (page 86).
> As an exception, if endorsement printing is executed (CAP_PRINTERENABLED = TRUE) with the endorsement method (CAP_PRINTER) set to “Print after scanning” (TWPR_ENDORSERBOTTOMAFTER), the paper is not ejected immediately after the scan is complete, but is ejected after the device has printed on it.

Values

Type:	TW_BOOL
Default Value:	TRUE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE

CAP_EPCS_EJECTDIRECTION

Specifies the direction of ejection initiated by CAP_CLEARPAGE.

Values

Type:	TW_UINT16	
Default Value:	TWEPCS_ED_FRONT	
Allowed Values:	TWEPCS_ED_FRONT:	Eject to front
	TWEPCS_ED_REAR:	Eject to rear
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET	
Containers for MSG_GET:	TW_ENUMERATION	
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION	

CAP_EPCS_SCANSIDE

Specifies which side to scan for single-side scanning.

> **NOTE** When the setting for the scan side (CAP_DUPLEXENABLED) is set to duplex (TRUE), this capability is ignored.

Values

Type:	TW_UINT16
Default Value:	TWEPCS_SS_FRONT
Allowed Values:	TWEPCS_SS_FRONT: Scans the face-down side TWEPCS_SS_BACK: Scans the face-up side
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_EPCS_SCANDIRECTION

Specifies the scanning direction.

Values

Type:	TW_UINT16
Default Value:	TWEPCS_SD_FRONTTOREAR
Allowed Values:	TWEPCS_SD_FRONTTOREAR: Delivers the document from the front to the rear as it scans TWEPCS_SD_REARTOFRONT: Delivers the document from the rear to the front as it scans
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_XNATIVERESOLUTION

Retrieves the optical resolution (unit: dpi) for the main scanning direction (X direction)

> **NOTE** The optical resolution of PLQ-22CS/PLQ-22CSM is 600 dpi.

Values

Type:	TW_FIX32
Default Value:	No Default
Allowed Values:	>0
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

ICAP_XRESOLUTION

Specifies the scan resolution (unit: dpi) for the main scanning direction (X direction).

Values

Type:	TW_FIX32
Default Value:	300
Allowed Values:	75, 100, 120, 200, 254, 240, 300, 400, 600
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_YNATIVERESOLUTION

Retrieves the optical resolution (unit: dpi) for the sub-scanning direction (Y direction).

> **NOTE** The optical resolution of PLQ-22CS/PLQ-22CSM is 600 dpi.

Values

Type:	TW_FIX32
Default Value:	No Default
Allowed Values:	>0
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

ICAP_YRESOLUTION

Specifies the scan resolution (unit: dpi) for the sub-scanning direction (Y direction).

> **NOTE** This capability is ignored, as the resolution for the main scanning direction (page 92) is also applied to the sub-scanning direction.

Values

Type:	TW_FIX32
Default Value:	300
Allowed Values:	75, 100, 120, 200, 240, 254, 300, 400, 600
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_XFERCOUNT

Specifies the number of image data that the application can accept.

Values

Type:	TW_INT16
Default Value:	-1
Allowed Values:	-1, 1 to 2^{15}
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_COMPRESSION

Specifies the compression format for memory transfer mode and file transfer mode.

Values

Type: TW_UINT16

Default Value: Varies with transfer mode (ICAP_XFERMECH) and file format setting (ICAP_IMAGEFILEFORMAT).

Allowed Values: Varies with transfer mode (ICAP_XFERMECH) and file format setting (ICAP_IMAGEFILEFORMAT).

Allowed Operations: MSG_GET
MSG_GETCURRENT
MSG_GETDEFAULT
MSG_SET
MSG_RESET

Containers for MSG_GET: TW_ENUMERATION

Containers for MSG_SET: TW_ONEVALUE
TW_ENUMERATION

Description

The values that you can specify for this capability varies with transfer mode (ICAP_XFERMECH) and file format setting (ICAP_IMAGEFILEFORMAT). See below:

Transfer mode (ICAP_XFERMECH)	File format (ICAP_IMAGEFILEFORMAT)	ICAP_COMPRESSION	File format
Memory transfer mode (TWSX_MEMORY)	-	TWCP_NONE	-
File transfer mode (TWSX_FILE)	Windows Bitmap format (TWFF_BMP)	TWCP_NONE	-
	JFIF format (TWFF_JFIF)	TWCP_EPCS_JPEGHIGH	JPEG format - high compression (priority on size)
		TWCP_EPCS_JPEGNORMAL *1	JPEG format - normal compression
		TWCP_EPCS_JPEGLOW	JPEG format - low compression (priority on quality)
	PDF format (TWFF_PDF)	TWCP_EPCS_PDFHIGH	PDF format - high quality
		TWCP_EPCS_PDFNORMAL *1	PDF format - normal quality
	TIFF format (TWFF_TIFF)	TWCP_NONE	-
		TWCP_GROUP4 *2	TIFF format CCITT (Group4) compression
		TWCP_JPEG *3	TIFF format JPEG compression

*1: Default value

*2: Default value when the pixel type (ICAP_PIXELTYPE) is Black and White (TWPT_BW)

*3: Default value when the pixel type (ICAP_PIXELTYPE) is Grayscale (TWPT_GRAY) or Color (TWPT_RGB)

ICAP_IMAGEFILEFORMAT

Specifies the file format for file transfers.

Values

Type:	TW_UINT16
Default Value:	TWFF_BMP
Allowed Values:	TWFF_BMP: Windows Bitmap format TWFF_JFIF: JFIF format TWFF_PDF: PDF format TWFF_TIFF: TIFF format

NOTE

PDF format (TWFF_PDF) can store multiple image data in one file.

Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

ICAP_XFERMECH

Specifies the transfer mode of image data.

Values

Type:	TW_UINT16
Default Value:	TWSX_NATIVE
Allowed Values:	TWSX_NATIVE TWSX_MEMORY TWSX_FILE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ENUMERATION
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_ENABLEDSUIONLY

Retrieves the support status of the GUI for specifying the values of the settings. For PLQ-22CS/PLQ-22CSM, only "TRUE" is retrieved.

Values

Type:	TW_BOOL
Default Value:	No Default
Allowed Values:	TRUE
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

CAP_INDICATORS

Specifies whether or not to display the progress indicator when scanning in UI-suppressed mode.

> **NOTE** Pressing the "Cancel" button on the progress indicator destroys the scanned data.

Values

Type:	TW_BOOL
Default Value:	TRUE
Allowed Values:	TRUE or FALSE
Allowed Operations:	MSG_GET MSG_GETCURRENT MSG_GETDEFAULT MSG_SET MSG_RESET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	TW_ONEVALUE TW_ENUMERATION

CAP_UICONTROLLABLE

Retrieves the support status of the UI-suppress mode. For PLQ-22CS/PLQ-22CSM, only “TRUE” is retrieved.

Values

Type:	TW_BOOL
Default Value:	No Default
Allowed Values:	TRUE
Allowed Operations:	MSG_GET
Containers for MSG_GET:	TW_ONEVALUE
Containers for MSG_SET:	Not Allowed

Reference for Extended Capabilities

This section explains the specifications of extended capabilities in PLQ-22CS/PLQ-22CSM.

TWEI_MAGDATA

Retrieves the result of MICR recognition.

> **NOTE** In addition to numbers, the retrieved result of MICR recognition contains control characters. Refer to "MICR Control Characters" on page 144 for details.

Values

Type: TW_STR255

Allowed Values: Any String

TWEI_MAGTYPE

Retrieves the type of magnetic character data.

Values

Type: TW_UINT16

Allowed Values: TWMD_MICR

TWEI_EPCS_OCRABDATA

Retrieves the result of OCR recognition.

NOTE

- The maximum number of characters that can be recognized is 127bytes.
- If the recognized OCR characters are on two lines, the division between the lines is marked with 0x0a.
- Characters that cannot be recognized are shown as “?”.
- To scan (detect) OCR-A/B font, the side of paper to scan (ICAP_EPSC_OCRABSIDE) and the detection target area (ICAP_EPSC_OCRABFRAME) need to be specified in advance.

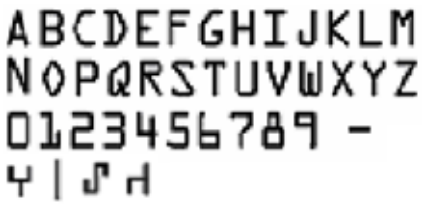
Values

Type: TW_STR255

Allowed Values: Any String

Description

The OCR-A and OCR-B fonts that can be detected by PLQ-22CS/PLQ-22CSM are shown below.

OCR-A Font	OCR-B Font
	

TWEI_BARCODECOUNT

Retrieves the number of barcodes recognized.

NOTE

If a barcode cannot be recognized, “-1” is returned.

Values

Type: TW_UINT32

Allowed Values: >=0

TWEI_BARCODEROTATION

Retrieves the rotation of the barcode that was recognized.

Values

Type:	TW_UINT32	
Allowed Values:	TWBCOR_ROT0:	Normal
	TWBCOR_ROT90:	90 degrees
	TWBCOR_ROT180:	180 degrees
	TWBCOR_ROT270:	270 degrees

TWEI_BARCODETEXTLENGTH

Retrieves the length of the string in the barcode that was recognized.

Values

Type:	TW_UINT32
Allowed Values:	>=0

TWEI_BARCODETEXT

Retrieves the barcode that was recognized.

Values

Type:	TW_STR255
Allowed Values:	Any String

TWEI_BARCODEX

Retrieves the starting position of the barcode that was recognized, in the main scanning direction (X direction).

Values

Type:	TW_UINT32
Allowed Values:	>=0

TWEI_BARCODEY

Retrieves the starting position of the barcode that was recognized, in the sub-scanning direction (Y direction).

Values

Type: TW_UINT32

Allowed Values: >=0

TWEI_BARCODETYPE

Retrieves the type of the barcode that was recognized.

Values

Type: TW_UINT32

Allowed Values:

TWBT_3OF9:	Code39
TWBT_2OF5INTERLEAVED:	ITF
TWBT_CODE128:	Code128
TWBT_CODABAR:	Codabar
TWBT_UPCA:	UPC-A
TWBT_UPCE:	UPC-E
TWBT_EAN8:	EAN8
TWBT_EAN13:	EAN13

Method of Retrieving Extended Image Information

To retrieve extended image information, issue DG_IMAGE/DAT_EXTIMAGEINFO/MSG_GET to the Data Source and retrieve the TW_EXTIMAGEINFO structure (page 104) which stores the extended image information.

The timing in which retrieval is allowed is as follows:

- To retrieve extended image information for the top side of the scan document:
  From the time the image on the top side is acquired until when DAT_PENDINGXFER is called.
- To retrieve extended image information for the bottom side of the scan document:
  From the time the image on the bottom side is acquired until when DAT_PENDINGXFER is called.

> **CAUTION** To retrieve extended image information, the image must be acquired. It cannot be retrieved before the image is transferred.

The following information can be retrieved from PLQ-22CS/PLQ-22CSM:

Type of information (InfoType)	InfoID	Data type	Remarks
MICR type	TWEI_MAGTYPE	TW_UINT16	Always returns TWMD_MICR.
MICR string	TWEI_MAGDATA	TW_STR255	Returns the read MICR string.
OCR-AB string	TWEI_EPCS_OCRABDATA	TW_STR255	Returns the read OCR-AB string.
Number of detected barcodes	TWEI_BARCODECOUNT	TW_UINT32	Returns -1 if barcode reading is disabled.
Barcode string	TWEI_BARCODETEXT	TW_STR255	Returns the read barcode string.
Barcode string length	TWEI_BARCODETEXTLENGTH	TW_UINT32	
Barcode type	TWEI_BARCODETYPE	TW_UINT32	Returns TWBT_****.
Coordinate (X axis) of the detected barcode	TWEI_BARCODEX	TW_UINT32	Returns the horizontal position in pixel unit, with the top left corner of the entire document image as the origin.
Coordinate (Y axis) of the detected barcode	TWEI_BARCODEY	TW_UINT32	Returns the vertical position in pixel unit, with the top left corner of the entire document image as the origin.
Direction of barcode detection	TWEI_BARCODEROTATION	TW_UINT32	Returns TWBCOR_ROT**.

> **NOTE**
> - The MICR type and MICR string can only be retrieved in PLQ-22CSM. In PLQ-22CS, TWRC_DATANOTAVAILABLE is set for TW_INFO.CondCode.
> - As for information related to barcodes, if multiple barcodes were detected, the results are stored in an array in the TW_INFO structure in such a way that integrity is maintained in the order of the information in the array.

TW_EXTIMAGEINFO Structure

```
typedef struct {
    TW_UINT32 NumInfos;                    IN
    TW_INFO[1] Info;                       IN
} TW_EXTIMAGEINFO, FAR * pTW_EXTIMAGEINFO;
```

TW_UINT32 NumInfos:

Stores the number of TW_INFO structures contained in TW_EXTIMAGEINFO.

TW_INFO[1] Info:

Stores a TW_INFO structure. Refer to "TW_INFO Structure" on page 104 for details.

TW_INFO Structure

```
typedef struct {
    TW_UINT16 InfoID;                      IN
    TW_UINT16 ItemType;                    OUT
    TW_UINT16 NumItems;                    OUT
    TW_UINT16 CondCode;                    OUT
    TW_UINTPTR Info;                       OUT
} TW_INFO, FAR * pTW_INFO;
```

TW_UINT16 InfoID:

Stores the InfoID for the type of extended image information.

TW_UINT16 ItemType:

Stores the data type of the information contained in the Item member.

TW_UINT16 NumItems:

Stores the number of information contained in the Item member.

TW_UINT16 CondCode:

Stores the retrieval results of the extended image information.

TW_UINTPTR Item:

Stores the extended image information.

NOTE

- The Item member is declared as TW_UINTPTR type, however, if the data size of the additional information is less than or equal to 4 bytes, the actual value is stored instead of a pointer.
- If the data size is more than 4 bytes, such as TWEI_MAGDATA, or if the data is an array, such as barcode data, a handle to the memory area that holds the data is stored.
- If the data is an array, the result is passed with a handle even if the size of each element is less than or equal to 4 bytes. The number of elements is set in the NumItems member.

CAUTION

The memory for the TW_INFO structure is secured on the Data Source side, however, releasing of the memory should be done on the application side.

Example of Implementation

The method of storing the retrieved information to the TW_INFO structure varies depending on whether there are multiple TW_INFO structures of the same information type (InfoType), or just one structure.

❏ If there is one TW_INFO structure for each InfoType (page 105)
Multiple information are stored the TW_INFO structure as an array. If multiple barcodes were recognized, the Item member of the TW_INFO structure is a memory handle to the array of size TW_STR255.

❏ If there are multiple TW_INFO structures with the same InfoType (page 107)
Additional information is stored in each TW_INFO structure. If multiple barcodes were recognized and two TW_INFO structures with InfoType=TWEI_BARCODEINFO exist in the TW_EXTIMAGEINFO structure, the 1st and 2nd recognition results are stored in each TW_INFO structure, respectively. If three or more barcodes exist, the third one and the rest are not acquired. This method allows you to retrieve only the necessary number of additional information.

Programming

The following is an example of implementation in Visual C++. Please make applicable changes for implementation in Visual Basic .NET.

If there is one TW_INFO structure for each InfoType

```
// Retrieve all of MICR string, barcode data, barcode data length, and barcode type
// fnMemAlloc, fnMemLock, fnMemUnlock, and fnMemFree are pointers to memory functions, added in TWAIN 2.0
// For versions older than TWAIN 2.0, GlobalAlloc, GlobalLock, GlobalUnlock, and GlobalFree are their counterparts
// pDSIdentity and pAppIdentity are pointers to TW_IDENTITY for the Data Source and the application, respectively

// Set TW_EXTIMAGEINFO structure
// The memory for TW_EXTIMAGEINFO and TW_INFO structures within is secured and released by the application
pTW_EXTIMAGEINFO pExtInfo = NULL;
pExtInfo = (pTW_EXTIMAGEINFO)(fnMemAlloc(sizeof(TW_EXTIMAGEINFO) + sizeof(TW_INFO) * (4 - 1)));

pExtInfo->NumInfos = 4;                                    // MICR & Barcode text & text length & type
pExtInfo->Info[0].InfoID = TWEI_MAGDATA;                   // MICR
pExtInfo->Info[1].InfoID = TWEI_BARCODETEXT;               // Barcode text
pExtInfo->Info[2].InfoID = TWEI_BARCODETEXTLENGTH;// Length of barcode text
pExtInfo->Info[3].InfoID = TWEI_BARCODETYPE;               // Type of barcode

// Retrieve TW_EXTIMAGEINFO
TW_UINT16 rc = fnDSMEntry(pAppIdentity, pDSIdentitiy, DG_IMAGE, DAT_EXTIMAGEINFO, MSG_GET,
pExtInfo);

if(rc == TWRC_SUCCESS)
{
        // MICR
        if(pExtInfo->Info[0].CondCode == TWRC_SUCCESS)
        {
                // Info member is a memory handle to TW_STR255
                char* pMicrText = (char*)(fnMemLock(pExtInfo->Info[0].Item));
```

```
                    //
                    // Process MICR data
                    //

                    // Memory released by the application
                    fnMemUnlock(pExtInfo->Info[0].Item);
                    fnMemFree(pExtInfo->Info[0].Item);
        } else {
                    // error case
        }

        // Barcode
        if(pExtInfo->Info[1].CondCode == TWRC_SUCCESS)
        {
                    // The Info[1] to [3] Items are handles to the array
                    // Because of the integrity in the order of arrays for each information, the nth element in
                    // BarcodeText and the nth element in BarcodeTextLength, and the nth element in
                    // BarcodeType all correspond to each other
                    // The number of elements in each array is stored in NumItems
                    pTW_STR255 barcodeText = (pTW_STR255)(fnMemLock(pExtInfo->Info[1].Item));
                    pTW_UINT32 barcodeTextLen = (pTW_UINT32)(fnMemLock(pExtInfo->Info[2].Item));
                    pTW_UINT16 barcodeType = (pTW_UINT16)(fnMemLock(pExtInfo->Info[3].Item));

                    for(int i=0; i< pExtInfo->NumItems; i++)
                    {
                                //
                                // Process barcode data
                                //
                    }

                    // Memory released by the application
                    fnMemUnlock(pExtInfo->Info[1].Item);
                    fnMemUnlock(pExtInfo->Info[2].Item);
                    fnMemUnlock(pExtInfo->Info[3].Item);
                    fnMemFree(pExtInfo->Info[1].Item);
                    fnMemFree(pExtInfo->Info[2].Item);
                    fnMemFree(pExtInfo->Info[3].Item);
        } else {
                    // error case
        }
}
else
{
        // error case
}

// Release TW_EXTIMAGEINFO structure
fnMemUnlock(pExtInfo);
fnMemFree(pExtInfo);
```

If there are multiple TW_INFO structures with the same Info Type

```
// Retrieve only two barcode data
// fnMemAlloc, fnMemLock, fnMemUnlock, and fnMemFree are pointers to memory functions, added in TWAIN 2.0
// For versions older than TWAIN 2.0, GlobalAlloc, GlobalLock, GlobalUnlock, and GlobalFree are their counterparts
// pDSIdentity and pAppIdentity are pointers to TW_IDENTITY for the Data Source and the application, respectively

// Set TW_EXTIMAGEINFO structure
// The memory for TW_EXTIMAGEINFO and TW_INFO structures within is secured and released by the application
pTW_EXTIMAGEINFO pExtInfo = NULL;
pExtInfo = (pTW_EXTIMAGEINFO)(fnMemAlloc(sizeof(TW_EXTIMAGEINFO) + sizeof(TW_INFO) * (2 ? 1)));

pExtInfo->NumInfos = 2;                                 // Barcode text * 2
pExtInfo->Info[0].InfoID = TWEI_BARCODETEXT;            // Barcode text 1
pExtInfo->Info[1].InfoID = TWEI_BARCODETEXT;            // Barcode text 2

// Retrieve TW_EXTIMAGEINFO
TW_UINT16 rc = fnDSMEntry(pAppIdentity, pDSIdentitiy, DG_IMAGE, DAT_EXTIMAGEINFO, MSG_GET,
pExtInfo);

if(rc == TWRC_SUCCESS)
{
        // Barcode text 1
        if(pExtInfo->Info[0].CondCode == TWRC_SUCCESS)
        {
                // Info member is a memory handle to TW_STR255
                char* pMicrText = (char*)(fnMemLock(pExtInfo->Info[0].Item));

                //
                // Process barcode text 1
                //

                // Memory released by the application
                fnMemUnlock(pExtInfo->Info[0].Item);
                fnMemFree(pExtInfo->Info[0].Item);
        } else {
                // error case
        }

        // Barcode text 2
        if(pExtInfo->Info[1].CondCode == TWRC_SUCCESS)
        {
                // Info member is a memory handle to TW_STR255
                char* pMicrText = (char*)(fnMemLock(pExtInfo->Info[1].Item));

                //
                // Process barcode text 2
                //

                // Memory released by the application
                fnMemUnlock(pExtInfo->Info[1].Item);
                fnMemFree(pExtInfo->Info[1].Item);
        }
        else if(pExtInfo->Info[1].CondCode == TWRC_DATANOTAVAILABLE)
        {
                // If the recognized barcodes do not meet the number of TW_INFO structures
                // TWRC_DATANOTAVAILABLE is stored as CondCode
```

```
            }
            else
            {
                        // error
            }
}
else
{
            // error case
}

// Release TW_EXTIMAGEINFO structure
fnMemUnlock(pExtInfo);
fnMemFree(pExtInfo);
```

PLQ-22 API Reference

This chapter explains the PLQ-22 APIs and their syntax.

Types of PLQ-22 APIs

Function	PLQ-22 API	Description
Starting/stopping status monitoring	BiOpenMonPrinter	Starts monitoring the status of the device.
	BiCloseMonPrinter	Stops monitoring the status of the device.
Exclusive control	BiLockPort	Locks the port.
	BiUnlockPort	Unlocks the port.
Retrieves the device status	BiGetStatus	Retrieves the device status when required by the application.
	BiGetMultiStatus	Retrieves detailed device status when required by the application.
	BiSetStatusBackFunction	Retrieves the device status using a callback when the status changes.
	BiSetStatusBackFunctionEx	Retrieves the device status using a callback when the status changes. Also retrieves the port that sent the notification.
	BiSetMultiStatusBackFunction	Retrieves detailed device status using a callback when the status changes.
	BiCancelStatusBack	Cancels the callback.
Retrieving device information	BiGetPrnCapability	Retrieves information such as the firmware of the device.
Executing and setting the reading process	BiSCNMICRFunction	Executes and sets MICR data reading.
Setting MICR data	BiMICRClearSpaces	Removes the space characters from the acquired MICR data.
Setting Magnetic Stripe	BiMSRWSetting	Sets the Magnetic Stripe.
Reading and retrieving the Magnetic Stripe data	BiMSRWReadStripeData	Executes the Magnetic Stripe reading process.
	BiMSRWGetStripeData	Retrieves the Magnetic Stripe data.
Writing and deleting Magnetic Stripe data	BiMSRWWriteStripeData	Writes data to the Magnetic Stripe.
	BiMSRWClearStripeData	Erases the Magnetic Stripe data.
Resetting the device	BiResetPrinter	Resets the device.
Ejecting paper	BiEjectPaper	Ejects the paper inserted in the device.
Retrieving driver/module versions	BiGetVersion	Retrieves the version of drivers and modules.

PLQ-22 API Type Definition

PLQ-22 API Types	C/C++ (Windows)
EPBS_INT32	LONG
EPBS_UINT8	BYTE
EPBS_UINT32	DWORD
EPBS_PUINT8	LPBYTE
EPBS_PCCHAR	LPCSTR
EPBS_PUINT32	LPDWORD
EPBS_PINT32	LPINT
EPBS_CHAR	CHAR
EPBS_PCHAR	LPSTR
EPBS_PVOID	LPVOID
EPBS_PUINT16	LPWORD
EPBS_UINT16	WORD
EPBS_BOOL	BOOL
EPBS_RECT	RECT

BiOpenMonPrinter

Starts monitoring the status of the specified device, and returns the handle.

> **NOTE** The handle retrieved by this API can only be used within the same application.

> **CAUTION** For Network connection, when you set a firewall, add the port number 2291 to [Exception].

Syntax

int ***BiOpenMonPrinter*** (int nType, EPBS_PCHAR pName)

Argument

nType: Specifies “TYPE_PRINTER” (constant).

pName: Specifies the name of the device whose status is to be monitored.

Example

<Local Connection>

```
int   BiOpenMonPrinter(TYPE_PRINTER, “PLQ-22U”);
```

<Network Connection>

```
int   BiOpenMonPrinter(TYPE_PRINTER, “192.168.192.168\PLQ-22U”);
int   BiOpenMonPrinter(TYPE_PRINTER, “SCANNER SERVER\PLQ-22U”);
```

Return value

If status monitoring started successfully, this API returns the handle that identifies the device. The handle is returned even if the device is offline. If it fails to open, one of the following return values is returned:

Macro Definition (Constant)	Value	Description
ERR_TYPE	-10	nType parameter error
ERR_OPENED	-20	The specified device is already open
ERR_NO_PRINTER	-30	The specified device cannot be found
ERR_NO_MEMORY	-50	Insufficient memory
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Cannot read/write to the device
ERR_PARAM	-90	Parameter error
ERR_NET_CONNECTED	-1150	Failed to connect to the device on the network
ERR_UNKNOWN	-9999	Invalid install configuration

> **NOTE** For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

Call this API before using other PLQ-22 API functions.The returned handle is used as the argument of other PLQ-22 API functions. This API returns one of the following return values, according to the status of the device when it was called:

Device Status	Behavior
Online	Returns the handle.
Offline	Returns the handle. However, the operator needs to switch the device back online.
Cable not connected/power is off	Returns "ERR_ACCESS".

BiGetStatus

Retrieves the current device status.

Syntax

int ***BiGetStatus*** (int nHandle, EPBS_PUINT32 lpStatus)

Argument

nHandle: Specifies the handle.

lpStatus: Returns the device status retained by PLQ-22 API.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

For a list of device statuses that can be retrieved with this API, refer to "Device Status" on page 147.

BiGetMultiStatus

Retrieves the current device status. A more detailed status can be retrieved than BiGetStatus.

Syntax

int ***BiGetMultiStatus*** (int nHandle, EPBS_UINT32 dwStatusLength, EPBS_PUINT32 lpdwStatuses)

Argument

nHandle: Specifies the handle.

dwStatusLength: Specifies the array length of lpdwStatuses. For PLQ-22 API, specify “4”.

lpdwStatuses: Returns the device status retained by PLQ-22 API.
The returned device status is different for each array. See below:

Array	Description
lpdwStatuses(0)	General status
lpdwStatuses(1)	Detailed recoverable error status
lpdwStatuses(2)	Detailed unrecoverable error status
lpdwStatuses(3)	Detailed auto-recovered error status

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

For a list of device statuses that can be retrieved with this API, refer to "Device Status" on page 147.

BiSetStatusBackFunction

Registers the callback function that gets called for automatic status notification.

Syntax

int ***BiSetStatusBackFunction*** (int nHandle,
int (CALLBACK EXPORT *pStatusCB)
(EPBS_UINT32 dwStatus))

Argument

nHandle: Specifies the handle. This is an INT type.

CALLBACK EXPORT *pStatusCB) (EPBS_UINT32 dwStatus):
Sets the address of the callback function that receives the device status notification.

dwStatus: Returns the device status retained by PLQ-22 API.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset

NOTE For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

When this API is called, the callback function is called with the device status set in dwStatus. When the device status changes, the callback function is called with the new information automatically set in dwStatus. BiCancelStatusBack cancels this API. For a list of device statuses that can be retrieved with this API, refer to "Device Status" on page 147.

NOTE You cannot use the PLQ-22 APIs from within the registered callback function.

BiSetStatusBackFunctionEx

Registers the callback function that gets called for automatic status notification. In addition, it can identify which port the callback is from.

Syntax

```
int BiSetStatusBackFunctionEx ( int nHandle,
                                int (CALLBACK EXPORT *pFunction)
                                ( DWORD dwStatus,
                                LPSTR lpcPortName ) )
```

Argument

nHandle: Specifies the handle.

int (CALLBACK EXPORT * pStatusCB)(EPBS_UINT32 dwStatus, EPBS_PCHAR lpcPortName):
Sets the address of the callback function that receives the device status notification.

dwStatus: Returns the device status retained by PLQ-22 API.

lpcPortName: Returns the name of the port used for the callback.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset

NOTE For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

When this API is called, the callback function is called with the device status set in dwStatus. When the device status changes, the callback function is called with the new information automatically set in dwStatus. BiCancelStatusBack cancels this API. For a list of device statuses that can be retrieved with this API, refer to "Device Status" on page 147.

NOTE You cannot use the PLQ-22 APIs from within the registered callback function.

BiSetMultiStatusBackFunction

Registers the callback function that gets called for automatic status notification. In addition, it can identify which port the callback is from. The status can be retrieved in more detail than BiSetStatusBackFunction or BiSetStatusBackFunctionEx.

Syntax

int ***BiSetMultiStatusBackFunction***
(int nHandle, int (CALLBACK EXPORT *pMultiStatusCB)
(EPBS_UINT32 dwStatusType, EPBS_UINT32 dwStatus,
EPBS_PCHAR lpcPortName))

Argument

nHandle: Specifies the handle.

int (CALLBACK EXPORT *pMultiStatusCB)
(EPBS_UINT32 dwStatusType, EPBS_UINT32 dwStatus, EPBS_PCHAR lpcPortName):
Sets the address of the callback function that receives the device status notification.

dwStatusType: Returns the type of device status retrieved. See below for different types of device status:

dwStatusType	Type
EPS_STATUS_INDEX_GENERAL	General status
EPS_STATUS_INDEX_RETURNNABLE_ERROR	Detailed recoverable error status
EPS_STATUS_INDEX_UNRETURNABLE_ERROR	Detailed unrecoverable error status
EPS_STATUS_INDEX_AUTO_CARRIAGE_RETURN_ERROR	Detailed auto-recovered error status

dwStatus: Returns the device status retained by PLQ-22 API.

lpcPortName: Returns the name of the port used for the callback.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

When this API is called, the callback function is called with the device status set in dwStatus. When the device status changes, the callback function is called with the new information automatically set in dwStatus. BiCancelStatusBack cancels this API. For a list of device statuses that can be retrieved with this API, refer to "Device Status" on page 147.

NOTE

You cannot use the PLQ-22 APIs from within the registered callback function.

BiCancelStatusBack

Cancels the automatic status notification request process that was called by BiSetStatusBackFunction, BiSetStatusBackFunctionEx, or BiSetMultiStatusBackFunction.

Syntax

int ***BiCancelStatusBack*** (int nHandle)

Argument

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value

NOTE

- For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.
- Returns “SUCCESS” even if you call this API by mistake while the automatic status notification request process has not been registered.

BiResetPrinter

Resets the device whose status is being monitored.
If an error occurs in the device, resolve the cause of the error, and then reset the device.

NOTE

- Use this API when a recoverable error occurs. Refer to "Device Status" on page 33 for details on recoverable errors.
- This is an exclusive API. Before calling it, call BiLockPort (page 135).
  After calling it, call BiUnlockPort (page 137).

Syntax

int ***BiResetPrinter*** (int nHandle)

Argument

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	Timeout error
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

To verify that this API was executed successfully, check its return value and verify that the device was reset and is back online (by checking the device status).
This API behaves as follows, according to the status of the device when the API was called:

Device Status		Return Value and Behavior of Device
Online		Returns "SUCCESS", resets the device, and brings it online.
Offline	without cause	Returns "SUCCESS", resets the device, and brings it online.
	with cause	Returns "SUCCESS" and resets the device, however, does not bring it online, since the cause of it being offline remains. The operator should check the device and remove the cause.
Cable not connected/power is off		Returns "ERR_ACCESS".
Scan or print in progress		Returns "ERR_EXEC_FUNCTION".

BiSCNMICRFunction

Executes and sets MICR data reading.

> **NOTE** This is an exclusive API. Before calling it, call BiLockPort (page 135).
> After calling it, call BiUnlockPort (page 137).

Syntax

int ***BiSCNMICRFunction*** (int nHandle, EPBS_PVOID lpvStruct,
EPBS_UINT16 wFunction)

Argument

nHandle: Specifies the handle.

lpvStruct: Sets the following parameters according to the function specified in wFunction

wFunction (Constant)	Parameter to Set for lpvStruct
MF_EXEC	"NULL"
MF_MICR_RETRANS	"NULL"
MF_SET_BASE_PARAM	Address of the MF_BASE01 structure
MF_SET_MICR_PARAM	Address of the MF_MICR structure
MF_CLEAR_BASE_PARAM	Address of the MF_BASE01 structure
MF_CLEAR_MICR_PARAM	Address of the MF_MICR structure
MF_GET_BASE_DEFAULT	Address of the MF_BASE01 structure
MF_GET_MICR_DEFAULT	Address of the MF_MICR structure

wFunction: Specifies the function of this API.

wFunction (Constant)	Description
MF_EXEC	Execute MICR data reading
MF_MICR_RETRANS	Retrieve the previously read MICR data
MF_SET_BASE_PARAM	Set the contents of the MF_BASE01 structure to the driver
MF_SET_MICR_PARAM	Set the contents of the MF_MICR structure to the driver
MF_CLEAR_BASE_PARAM	Destroy the MF_BASE01 structure retained in the driver
MF_CLEAR_MICR_PARAM	Destroy the MF_MICR structure retained in the driver
MF_GET_BASE_DEFAULT	Set the default value of the MF_BASE01 structure to its members
MF_GET_MICR_DEFAULT	Set the default value of the MF_MICR structure to its members

Return value

BiSCNMICRFunction

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Insufficient memory
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Cannot read/write to the device
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported function executed
ERR_OFFLINE	-110	Cannot use: Waiting to return from an offline state
ERR_PAPERINSERT_TIMEOUT	-300	Failed to insert paper
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset
ERR_MICR	-440	Failed to read the MICR data
ERR_PAPER_JAM	-1020	Paper jam error
ERR_PAPER_INSERT	-1100	Failed to insert paper

MF_BASE01.iRet

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Insufficient memory
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Cannot read/write to the device
ERR_PARAM	-90	Parameter error
ERR_PAPERINSERT_TIMEOUT	-300	Failed to insert paper
ERR_MICR	-440	Failed to read the MICR data
ERR_NOT_EXEC	-470	Reading process not executed
ERR_PAPER_JAM	-1020	Paper jam error
ERR_PAPER_INSERT	-1100	Failed to insert paper

MF_MICR.iRet

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Insufficient memory
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Cannot read/write to the device
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-140	Insufficient buffer error
ERR_NOT_FOUND	-220	Data not found
ERR_MICR	-440	Failed to read the MICR data
ERR_NOT_EXEC	-470	Reading process not executed

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

❏ When setting the reading process, consider the following:

- When executing MICR data reading, be sure to set each structure (MF_BASE01 and MF_MICR).
- To re-set the structure, change the members of the structure and call this API. When doing this, set the second parameter (wFunction) to MF_SET_xxxx_PARAM.
- The contents of all structures are stored by the PLQ-22 API until BiCloseMonPrinter is called.

❏ When executing the reading process, consider the following:

- When MICR data reading is executed with this API, the reading result is stored in the structure set in the driver.
- The return value of the reading result is stored in the structure. Therefore, do not destroy the structure until the reading process is complete. When destroying the structure, first call MF_CLEAR_xxxx_PARAM before destroying it.

BiGetPrnCapability

Retrieves device information.

> **NOTE** This is an exclusive API. Before calling it, call BiLockPort (page 135).
> After calling it, call BiUnlockPort (page 137).

Syntax

int ***BiGetPrnCapability*** (int nHandle, EPBS_UINT8 prnID,
EPBS_PUINT8 pBuffSize, EPBS_PUINT8 pBuff)

Argument

nHandle: Specifies the handle.

prnID: Specifies the ID of the device whose device information you want to retrieve.

pBuffSize: Specifies the size of the memory for setting the device information (1 to 80). After this API is called, the size of the actual read data is returned. If there is insufficient buffer, the number of bytes required is returned.

pBuff: Specifies the memory address to set the device information. In addition, the information of the specified PrnID is returned.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Cannot read/write to the device
ERR_PARAM	-90	Parameter error
ERR_OFFLINE	-110	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset

> **NOTE** For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

For a list of device information that can be retrieved, refer to "Device ID" on page 148.

BiCloseMonPrinter

Stops monitoring the status of the device.

Syntax

int ***BiCloseMonPrinter*** (int nHandle)

Argument

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

BiGetVersion

Retrieves the version of drivers and modules.

> NOTE
> This API can be called without starting the status monitoring (BiOpenMonPrinter).

Syntax

int ***BiGetVersion*** (int nDriverType, int nType,
LPVERSION_INFO lpVersion)

Argument

nDriverType: Specifies the driver whose version you want to retrieve.
For PLQ-22 API, specify "DRIVER_TYPE_PLQ22" (constant).

nType: Specifies the version to retrieve. Specify from the following:

nType (Constant)	Type
VERSION_TYPE_DRIVER	Scanner driver version
VERSION_TYPE_OCR	OCR recognition module version
VERSION_TYPE_IMAGE	Image processing module version
VERSION_TYPE_BARCODE	Barcode module version
VERSION_TYPE_PDF	PDF file creator module
VERSION_TYPE_DDE	Text enhancement/color enhancement module
VERSION_TYPE_PH	Communication module

lpVersion: Sets the address of the VERSION_INFO structure. The retrieved version is set in the VERSION_INFO structure. Refer to "Description" on page 126 for details.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	The specified module cannot be found

> NOTE
> For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

The details of the VERSION_INFO structure and the version information to be set are as follows:

```
typedef struct {
    EPBS_UINT16    lpszDescription[VERSION_CHAR_MAX];
    EPBS_UINT16    lpszVersion[VERSION_CHAR_MAX];
} VERSION_INFO, *LPVERSION_INFO;
```

Version information
Information to retrieve
Retrieved version information

BiMICRClearSpaces

Removes the space characters contained in the acquired MICR data.

Syntax

int ***BiMICRClearSpaces*** (int nHandle, EPBS_UINT8 bClearSpace)

Argument

nHandle: Specifies the handle.

bClearSpace: Disables/enables removal of the space characters contained in the MICR data. Specify from the following:

bClearSpace (Constant)	Description
CLEAR_SPACE_DISABLE (Default value)	Space characters not removed
CLEAR_SPACE_ENABLE	Space characters removed

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

The MICR data subject to the removal of the space characters by this API are the following members of the MF_MICR structure:

- szMicrStr: MICRstring
- stOcrReliableInfo: Reliability information of MICR character recognition

BiMSRWSetting

Sets the necessary information to the device to read, write, and clear the Magnetic Stripe data on passbooks.

> NOTE This API is proprietary to PLQ-22 CSM.

Syntax

int ***BiMSRWSetting*** (int nHandle, LPMF_MSRW pSetting)

Argument

nHandle: Specifies the handle.

pSetting: Sets the address of the MF_MSRW structure.
Refer to "MF_MSRW" on page 145 for details.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Cannot read/write to the device
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	This API is not supported
ERR_OFFLINE	-110	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset

> NOTE For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

This API takes the contents that is set in the MF_MSRW structure (page 145) and sets them to the device.

BiMSRWReadStripeData

Waits for the user to insert the passbook and reads the Magnetic Stripe.

NOTE

- The retrieval of the Magnetic Stripe data is done with BiMSRWGetStripeData.
- This is an exclusive API.Before calling it, call BiLockPort (page 135).
  After calling it, call BiUnlockPort (page 137).
- This API is proprietary to PLQ-22 CSM.

Syntax

int ***BiMSRWReadStripeData*** (int nHandle, EPBS_UINT32 dwTimeout)

Argument

nHandle: Specifies the handle.

dwTimeout: Sets the timeout value for loading the passbook, in ms.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Cannot read/write to the device
ERR_PARAM	-90	Parameter error BiMSRWSetting was not called
ERR_NOT_SUPPORT	-100	This API is not supported
ERR_OFFLINE	-110	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

The Magnetic Stripe data that was read with this API is retained until BiEjectPaper is called.

BiMSRWGetStripeData

Retrieves the Magnetic Stripe data that was read with BiMSRWReadStripeData (page 129).

> **NOTE** This API is proprietary to PLQ-22 CSM.

Syntax

int ***BiMSRWGetStripeData*** (int nHandle, EPBS_UINT32 dwBlockNumber,
EPBS_PUINT8 pbRead,
EPBS_UINT32 dwReadLength,
EPBS_PUINT32 pdwReturnLength,
EPBS_PUINT32 pdwDetail)

Argument

nHandle: Specifies the handle.

dwBlockNumber: Specifies the block number of the Magnetic Stripe data you are retrieving. The block numbers that can be specified are "1" and "2".

> **NOTE**
> - To retrieve the Magnetic Stripe data in the second block, dwBlockCount in the MF_MSRW structure must be set to "2".
> - To retrieve the Magnetic Stripe data from both the first and second blocks, retrieve the Magnetic Stripe data in the first block, and then use this API again to retrieve the Magnetic Stripe data in the second block.

pbRead: Sets the memory address. Calling this API returns the Magnetic Stripe data.

dwReadLength: Specifies the array length of pbRead.

pdwReturnLength: The data size of the Magnetic Stripe data that was retrieved is set.

> **NOTE** pdwReturnLength is set when "NULL" is specified for pbRead, and "0" is specified for dwReadLength.

pdwDetail: The details of the reading result are set.

Hex	Decimal	Description
30H	48	Success
80H	128	Passbook not loaded
81H	129	Magnetic Stripe not detected
A0H	160	Abnormal reading of unknown cause
A1H	161	Cannot read the Magnetic Stripe data
A2H	162	SOM error
A3H	163	EOM error
A4H	164	LRC error
A5H	165	VRC error
A6H	166	The number of read data exceeds the maximum length
A7H	167	Prohibited character detected

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	This API is not supported
ERR_OFFLINE	-110	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset
ERR_NOT_EXEC	-470	Reading process not executed
ERR_MSRW_NODATA	-1160	Magnetic Stripe data not detected

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

To retrieve the Magnetic Stripe data, this API needs to be called twice.

First time: Retrieves the data size of the Magnetic Stripe data.

Second time:Retrieves the Magnetic Stripe data.

(The data size retrieved from the first call is used in the fourth parameter dwReadLength in the second call.)

<Example>

```
// Retrieve the data size of the Magnetic Stripe data
nResult = BiMSRWGetStripeData(m_nHandle, 1, NULL, 0, &dwReadLength, &dwDetail);
// Retrieve the Magnetic Stripe data
nResult = BiMSRWGetStripeData(m_nHandle, 1, lpbRead, dwReadLength, &dwReadLength, &dwDetail);
```

BiMSRWWriteStripeData

Waits for the user to insert the passbook and writes the specified data to the Magnetic Stripe.

NOTE

- This is an exclusive API.Before calling it, call BiLockPort (page 135).
  After calling it, call BiUnlockPort (page 137).
- This API is proprietary to PLQ-22 CSM.

Syntax

int ***BiMSRWWriteStripeData*** (int nHandle, EPBS_UINT32 dwTimeout,
EPBS_PUINT8 lpWrite,
EPBS_UINT32 dwLength)

Argument

nHandle: Specifies the handle.

dwTimeout: Sets the timeout value for loading the passbook, in ms.

lpWrite: Specifies the first pointer (30H to 3FH) of the BYTE array to write to the Magnetic Stripe.

dwLength: Specifies the array length (0 to 105) of lpWrite.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Cannot read from or write to the device
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	This API is not supported
ERR_OFFLINE	-110	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset
ERR_MSRW_WRITE	-1180	Failed to write Magnetic Stripe data

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

BiMSRWClearStripeData

Waits for the user to insert the passbook and clears the Magnetic Stripe data.

NOTE

- This is an exclusive API. Before calling it, call BiLockPort (page 135).
  After calling it, call BiUnlockPort (page 137).
- This API is proprietary to PLQ-22 CSM.

Syntax

int ***BiMSRWClearStripeData*** (int nHandle, EPBS_UINT32 dwTimeout)

Argument

nHandle: Specifies the handle.

dwTimeout: Sets the timeout value for loading the passbook, in ms.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Cannot read from or write to the device
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	This API is not supported
ERR_OFFLINE	-110	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset
ERR_MSRW_WRITE	-1180	Failed to clear the Magnetic Stripe data

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

BiEjectPaper

Ejects the inserted paper.

> **NOTE** This is an exclusive API. Before calling it, call BiLockPort (page 135).
> After calling it, call BiUnlockPort (page 137).

Syntax

int ***BiEjectPaper*** (int nHandle, WORD wEjectType,
EPBS_UINT32 dwTimeout)

Argument

nHandle: Specifies the handle.

wEjectType: Specifies where the paper should be ejected to. The value can be specified from the following:

wEjectType (Constant)	Description
MF_EJECT_FRONT	Eject to the front
MF_EJECT_REAR	Eject to the rear

dwTimeout: Specifies the timeout value in ms (milliseconds).

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Cannot read/write to the device
ERR_PARAM	-90	Parameter error
ERR_OFFLINE	-110	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset
ERR_UNLOCKED	-1140	Port is not locked

> **NOTE** For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

BiLockPort

Locks the port and takes sole possession of the execution rights of the exclusive API (API that directly accesses the device).

CAUTION

- After locking the port with this API and executing the exclusive API, be sure to execute BiUnlockPort (page 137).
- If a TWAIN API is running, the TWAIN API has exclusive controls.
  If you want to exclusively control PLQ-22 API, you can do so only after the execution of the TWAIN API is complete.

Syntax

int ***BiLockPort*** (int nHandle, EPBS_UINT32 dwTimeout)

Argument

nHandle: Specifies the handle.

dwTimeout: Specifies the timeout value in ms (milliseconds).

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	Timeout error
ERR_EXEC_FUNCTION	-310	Cannot use: Another API is running
ERR_RESET	-400	Cannot use: Device is being reset

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

When you lock the port with this API, exclusive APIs from other processes will not be accepted. The API that cancels this process is BiUnlockPort (page 137). The exclusive APIs that require this API to be called are as follows:

- BiResetPrinter
- BiSCNMICRFunction
- BiGetPrnCapability
- BiMSRWReadStripeData
- BiMSRWWriteStripeData
- BiMSRWClearStripeData
- BiEjectPaper

CAUTION	When you lock a port with this API, the command output from other processes cannot be executed. Set short lock intervals.

NOTE	• The locking of the port is done at process level. For this reason, while the port is locked, you can execute exclusive APIs from other threads of the same process. • This API can be executed multiple times repeatedly from the process that has locked the port. This results in a multiple-locked state. To unlock the port, execute BiUnlockPort the same number of times as this API was executed.

BiUnlockPort

Unlocks the port.

Syntax

int ***BiUnlockPort*** (int nHandle)

Argument

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_RESET	-400	Cannot use: Device is being reset
ERR_NOT_EXEC	-470	Process not executed

NOTE

- For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.
- If the port was not locked, “ERR_NOT_EXEC” is returned.

Description

Unlocking the port with this API allows exclusive APIs to be executed from other processes.

Structures

MF_BASE01

The MF_BASE01 structure is used to set basic operation settings for the reading process as well as the reading process result.

```
typedef struct {
    int iSize;                                          IN
    int iVersion;                                       IN
    int iRet;                                           OUT
    EPBS_UINT32 dwNotifyType;                           IN
    EPBS_UINT32 dwTimeout;                              IN
    union {
        LPHANDLE lphNotifyEvent;                        IN
        HWND hNotifyWnd;                                IN
    } uNotifyHandle;
    HWND hProgressWnd;                                  IN
    EPBS_UINT16 wErrorEject;                            IN
    EPBS_UINT8 bBuzzerHz[MF_BUZZER_TYPE_MAX];           IN
    EPBS_UINT8 bBuzzerCount[MF_BUZZER_TYPE_MAX];        IN
    EPBS_UINT8 bUseNVMemory;                            IN
    char cPortName[256];                                OUT
    EPBS_UINT16 wSuccessEject;                          IN
} MF_BASE01, *LPMF_BASE01;
```

int iSize:

Sets the size of the structure.

int iVersion:

Sets the version of the structure. Set “MF_BASE_VERSION” (constant).

int iRet:

The return value from the reading process is set. Refer to "MF_BASE01.iRet" on page 122 for details.

EPBS_UINT32 dwNotifyType:

Specifies the behavior when the third parameter of BiSCNMICRFunction (wFunction) is set to “MF_EXEC” or “MF_MICR_RETRANS”. See below for the behaviors that can be specified.

dwNotifyType (Constant)	Description
MF_BASE_MESSAGE_NO_MESSAGE	The API does not generate a new thread, but runs the process specified by the structure. After the process is complete, the control is returned.

EPBS_UINT32 dwTimeout:

Specifies the timeout value for the paper to be inserted, in seconds (0 to 300).

union uNotifyHandle:

Set the default value. Refer to "Default Values of the MF_BASE01 Structure" on page 140 for details.

HWND hProgressWnd:

Set the default value. Refer to "Default Values of the MF_BASE01 Structure" on page 140 for details.

EPBS_UINT16 wErrorEject:

Specifies how the paper is ejected if an error occurs during the reading process.

See below for the methods you can specify:

wErrorEject (Constant)	Description
MF_EXIT_ERROR_FRONT_SIDE	Eject the paper to the front of the device
MF_EXIT_ERROR_DISCHARGE	
MF_EXIT_ERROR_REAR_SIDE	Eject the paper to the rear of the device
MF_EXIT_ERROR_RELEASE	
MF_EXIT_ERROR_NOEJECT	Do not eject paper.

EPBS_UINT8 bBuzzerHz[MF_BUZZER_TYPE_MAX]:

Set the default value. Refer to "Default Values of the MF_BASE01 Structure" on page 140 for details.

EPBS_UINT8 bBuzzerCount[MF_BUZZER_TYPE_MAX]:

Set the default value. Refer to "Default Values of the MF_BASE01 Structure" on page 140 for details.

EPBS_UINT8 bUseNVMemory:

Set the default value. Refer to "Default Values of the MF_BASE01 Structure" on page 140 for details.

char cPortName[256]:

The name of the port in which the reading process was executed is set (NULL-terminated string). If ERR_HANDLE is returned, it is zero-cleared.

EPBS_UINT16 wSuccessEject:

Specifies how the paper is ejected when the reading process is complete. See below for the methods you can specify:

wErrorEject (Constant)	Description
MF_EXIT_SUCCESS_FRONT_SIDE	Eject the paper to the front of the device
MF_EXIT_SUCCESS_DISCHARGE	
MF_EXIT_SUCCESS_REAR_SIDE	Eject the paper to the rear of the device
MF_EXIT_SUCCESS_RELEASE	
MF_EXIT_SUCCESS_NOEJECT	Do not eject paper.

Default Values of the MF_BASE01 Structure

The following are values that can be retrieved with “MF_GET_BASE_DEFAULT” of BiSCNMICRFunction (page 121).

Member of MF_BASE01 Structure	Default Value
MF_BASE01.dwNotifyType	MF_BASE_MESSAGE_NO_MESSAGE
MF_BASE01.dwTimeout	MF_BASE_TIMEOUT_DEFAULT
MF_BASE01.uNotifyHandle.hNotifyWnd	0
MF_BASE01.hProgressWnd	0
MF_BASE01.wErrorEject	MF_EXIT_ERROR_FRONT_SIDE
MF_BASE01.bBuzzerHz(0)	MF_BUZZER_HZ_4000
MF_BASE01.bBuzzerHz(1)	MF_BUZZER_HZ_4000
MF_BASE01.bBuzzerHz(2)	MF_BUZZER_HZ_4000
MF_BASE01.bBuzzerCount(0)	MF_BUZZER_DISABLE
MF_BASE01.bBuzzerCount(1)	MF_BUZZER_DISABLE
MF_BASE01.bBuzzerCount(2)	MF_BUZZER_DISABLE
MF_BASE01.bUseNVMemory	MF_BASE_NVMEMORY_NOT_USE
MF_BASE01.wSuccessEject	MF_EXIT_SUCCESS_FRONT_SIDE

MF_MICR

The MF_MICR structure is used to set the settings for the MICR data to be read, as well as the MICR data reading result.

```
typedef struct {
    int iSize;                                                          IN
    int iVersion;                                                       IN
    int iRet;                                                           OUT
    EPBS_UINT8 bFont;                                                   IN
    EPBS_UINT8 bMicOcrSelect;                                           IN
    EPBS_BOOL blParsing;                                                IN
    EPBS_UINT8 bStatus;                                                 OUT
    EPBS_UINT8 bDetail;                                                 OUT
    EPBS_CHAR szMicrStr[MF_MICR_CHAR_MAX];                              OUT
    MF_OCR_RELIABLE_INFO stOcrReliableInfo[MF_MICR_CHAR_MAX];           OUT
    EPBS_CHAR szAccountNumber[MF_MICR_CHAR_MAX];                        OUT
    EPBS_CHAR szAmount[MF_MICR_CHAR_MAX];                               OUT
    EPBS_CHAR szBankNumber[MF_MICR_CHAR_MAX];                           OUT
    EPBS_CHAR szSerialNumber[MF_MICR_CHAR_MAX];                         OUT
    EPBS_CHAR szEPC[MF_MICR_CHAR_MAX];                                  OUT
    EPBS_CHAR szTransitNumber[MF_MICR_CHAR_MAX];                        OUT
    long lCheckType;                                                    OUT
    long lCountryCode;                                                  OUT
} MF_MICR, *LPMF_MICR;
```

int iSize:

Sets the size of the structure.

int iVersion:

Sets the version of the structure. Set “MF_MICR_VERSION” (constant).

int iRet:

The return value from the result of the MICR data reading is set. Refer to "MF_MICR.iRet" on page 122 for details.

EPBS_UINT8 bFont:

Specifies the font of the MICR data to be read. See below for the fonts that can be specified:

bFont (Constant)	Description
MF_MICR_FONT_E13B	E13B font
MF_MICR_FONT_CMC7	CMC7 font (including numbers)
MF_MICR_FONT_CMC7_ALPHANUM	CMC7 font (including alphabets and numbers)
MF_MICR_FONT_CMC7_NUM	CMC7 font (including numbers)

EPBS_UINT8 bMicOcrSelect:

Specifies the reading operation of MICR data. See below for the behaviors that can be specified.

bMicOcrSelect (Constant)	Description
MF_MICR_USE_MICR	Retrieve the analysis result for magnetic waveform data only (No retries)
MF_MICR_USE_MICR_RETRY	Retrieve the analysis result for magnetic waveform data only (With retries)
MF_MICR_USE_MICOCR_RETRY1	Retrieve the analysis results of magnetic waveform data and optical character recognition (With retries. Initially reads using magnetic waveform)
MF_MICR_USE_MICOCR_RETRY2	Retrieve the analysis results of magnetic waveform data and optical character recognition font (With retries. Initially read using both magnetic waveform and optical recognition)

EPBS_BOOL blParsing:

Specifies parsing. For PLQ-22 API, specify "FALSE".

EPBS_UINT8 bStatus:

The MICR reading status is set.

Bit	Status Contents	ON / OFF	Value	Status
0	Reading font	ON	0x0001	E13B
		OFF	0x0000	CMC7
1	---	---	0x0000	Reserved (Fixed to 0)
2	---	---	0x0000	Reserved (Fixed to 0)
3	Detailed information	ON	0x0008	With addition information
		OFF	0x0000	Without additional information
4	Reread	ON	0x0010	Not possible
		OFF	0x0000	Possible
5	Reading result	ON	0x0020	Abnormal termination
		OFF	0x0000	Success
6	---	---	0x0040	Reserved (Fixed to 1)
7	---	---	0x0000	Reserved (Fixed to 0)

NOTE

The MICR data reading status is set when the MICR reading result from PLQ-22 is read successfully by the API. If an error occurs before the API can successfully read the MICR reading result from PLQ-22, the MICR reading status does not get set.

EPBS_UINT8 bDetail:

Detailed MICR reading status is set.

Value	Information
40h	No error
41h	Reading process not executed
45h	MICR character not detected
46h	Unsupported character detected during the analysis process
48h	Abnormality detected in noise measurement

EPBS_CHAR szMicrStr[MF_MICR_CHAR_MAX]:

The read MICR data is set.

Refer to "MICR Control Characters" on page 144 for details on MICR control characters.

MF_OCR_RELIABLE_INFO stOcrReliableInfo[MF_MICR_CHAR_MAX]:

The reliability of the read MICR data is set. Refer to "The Reliability of the MICR Data (MF_OCR_RELIABLE_INFO Structure)" on page 144 for details.

EPBS_CHAR szAccountNumber[MF_MICR_CHAR_MAX]:

Set the default value. Refer to "Default Values of the MF_MICR Structure" on page 143 for details.

EPBS_CHAR szAmount[MF_MICR_CHAR_MAX]:

Set the default value. Refer to "Default Values of the MF_MICR Structure" on page 143 for details.

EPBS_CHAR szBankNumber[MF_MICR_CHAR_MAX]:

Set the default value. Refer to "Default Values of the MF_MICR Structure" on page 143 for details.

EPBS_CHAR szSerialNumber[MF_MICR_CHAR_MAX]:

Set the default value. Refer to "Default Values of the MF_MICR Structure" on page 143 for details.

EPBS_CHAR szEPC[MF_MICR_CHAR_MAX]:

Set the default value. Refer to "Default Values of the MF_MICR Structure" on page 143 for details.

EPBS_CHAR szTransitNumber[MF_MICR_CHAR_MAX]:

Set the default value. Refer to "Default Values of the MF_MICR Structure" on page 143 for details.

long lCheckType:

Set the default value. Refer to "Default Values of the MF_MICR Structure" on page 143 for details.

long lCountryCode:

Set the default value. Refer to "Default Values of the MF_MICR Structure" on page 143 for details.

Default Values of the MF_MICR Structure

The following are values that can be retrieved with “MF_GET_MICR_DEFAULT” of BiSCNMICRFunction (page 121)

Member of MF_MICR Structure	Default Value
MF_MICR.bFont	MF_MICR_FONT_E13B
MF_MICR.bMicOcrSelect	MF_MICR_USE_MICR
MF_MICR.blParsing	FALSE
MF_MICR.bStatus	0
MF_MICR.bDetail	0
MF_MICR.szMicrStr	Zero clear
MF_MICR.stOcrReliableInfo	Zero clear
MF_MICR.szAccountNumber	Zero clear
MF_MICR.szAmount	Zero clear
MF_MICR.szBankNumber	Zero clear
MF_MICR.szSerialNumber	Zero clear
MF_MICR.szEPC	Zero clear
MF_MICR.szTransitNumber	Zero clear
MF_MICR.lCheckType	0
MF_MICR.lCountryCode	0

MICR Control Characters

MICR data that can be retrieved includes control characters in addition to numbers. See below:

E13B Font

MICR Character	Name	Alternate Character
	Transit	t
	Amount	a
	On-Us	o
	Dash	-

CMC7 Font

MICR Character	Alternate Character
	/
	#
	=
	>
	^

The Reliability of the MICR Data (MF_OCR_RELIABLE_INFO Structure)

The reliability of the MICR data is set in the MF_OCR_RELIABLE_INFO structure.

```
typedef struct {
    long lPosition;                          OUT
    MF_OCR_RELIABILITY stFirstSelect;        OUT
    MF_OCR_RELIABILITY stSecondSelect;       OUT
} MF_OCR_RELIABLE_INFO, *LPMF_OCR_RELIABLE_INFO;
```

Description
Position (0 is left edge)
First recognition candidate
Second recognition candidate

MF_MSRW

The MF_MSRW structure sets the Magnetic Stripe functions.

```
typedef struct {
    int iSize;                          IN
    int iVersion;                       IN
    EPBS_UINT32 dwRecordFormat;         IN
    EPBS_UINT32 dwIBMDataType;          IN
    EPBS_UINT32 dwReadRetry;            IN
    EPBS_UINT32 dwWriteAndClearRetry;   IN
    EPBS_UINT32 dwBlockCount;           IN
    EPBS_UINT32 dwVerticalAdjust;       IN
} MF_MSRW, *LPMF_MSRW;
```

int iSize:

Sets the size of the structure.

int iVersion:

Sets the version of the structure.

Set "MF_MSRW_VERSION" (constant).

EPBS_UINT32 dwRecordFormat:

Specifies the record format. See below for the record formats that can be specified:

dwRecordFormat (Constant)	Record Format
EPS_BI_MSRW_FORMAT_IBM3604	IBM3604
EPS_BI_MSRW_FORMAT_DINISO	DINISO
EPS_BI_MSRW_FORMAT_ANSI	ANSI
EPS_BI_MSRW_FORMAT_IBM4746	IBM4746
EPS_BI_MSRW_FORMAT_ISO7811	ISO7811
EPS_BI_MSRW_FORMAT_HT2751CIZ	HT2751CIZ
EPS_BI_MSRW_FORMAT_ISO8484	ISO8484

EPBS_UINT32 dwIBMDataType:

Sets the EOM of the record format (dwRecordFormat). The relationship between the EOF that can be specified and the record format is as follows:

dwIBMDataType / dwIBMDataType (Constant)	EPS_BI_MSRW_FORMAT_IBM3604	EPS_BI_MSRW_FORMAT_DINISO	EPS_BI_MSRW_FORMAT_ANSI	EPS_BI_MSRW_FORMAT_IBM4746	EPS_BI_MSRW_FORMAT_ISO7811	EPS_BI_MSRW_FORMAT_HT2751CIZ	EPS_BI_MSRW_FORMAT_ISO8484
EPS_BI_MSRW_EOM_NONE	○	○	○	○	○	○	○
EPS_BI_MSRW_EOM_3C	○	-	-	○	-	-	-
EPS_BI_MSRW_EOM_3F	○	-	-	○	-	-	-

EPBS_UINT32 dwReadRetry:

Sets the number of retries for Magnetic Stripe reading (0 to 3).

EPBS_UINT32 dwWriteAndClearRetry:

Sets the number of retries for writing and clearing the Magnetic Stripe (0 to 3).

EPBS_UINT32 dwBlockCount:

Specifies the block to read and write the Magnetic Stripe data. See below for details:

dwBlockCount (Value)	PLQ-22 API	Description
1	BiMSRWReadStripeData	Read data in the first block
	BiMSRWWriteStripeData	Write data to the first block
2	BiMSRWReadStripeData	Read data in the second block
	BiMSRWWriteStripeData	Write data to the second block

EPBS_UINT32 dwVerticalAdjust:

Sets the correction distance in vertical direction (0 to 5588) in units of 0.1 mm.

Device Information

Device Status

BiGetMultiStatus/ BiSetMultiStatusBackFunction

General Statuses

Macro Definition (Constant)	ON / OFF	Value	Status
ESCMC_ASB_NO_RESPONSE	ON	0x00000001	Device not responding
	OFF	0x00000000	Device responding
ESCMC_ASB_OFFLINE	ON	0x00000002	Offline
	OFF	0x00000000	Online
ESCMC_ASB_PAPER_FEED	ON	0x00000004	
	OFF	0x00000000	
ESCMC_ASB_PAPER_SET	ON	0x00000008	Paper is set
	OFF	0x00000000	Paper is not set
ESCMC_ASB_COVER_STATUS	ON	0x00000010	Cover is open
	OFF	0x00000000	Cover is closed

Detailed Recoverable Error Statuses

Macro Definition (Constant)	ON / OFF	Value	Status
ESCMC_ASB_NO_PAPER	ON	0x00000002	Out-of-paper error occurred
	OFF	0x00000000	No out-of-paper error
ESCMC_ASB_PAPER_EJECT	ON	0x00000010	Ejection error occurred
	OFF	0x00000000	No ejection error
ESCMC_ASB_COVER_OPEN	ON	0x00000800	Cover open error occurred
	OFF	0x00000000	No cover open error

Detailed Unrecoverable Error Statuses

Macro Definition (Constant)	ON / OFF	Value	Status
ESCMC_ASB_FATAL_ERROR	ON	0x00000001	Fatal error occurred
	OFF	0x00000000	No fatal error

Detailed Auto-recovered Error Status

Macro Definition (Constant)	ON / OFF	Value	Status
ESCMC_ASB_HEAD_HOT_ERROR	ON	0x00000001	Head hot error occurred
	OFF	0x00000000	No head hot error

BiGetStatus/ BiSetStatusBackFunction/ BiSetStatusBackFunctionEx

Macro Definition (Constant)	ON / OFF	Value	Status
ASB_NO_RESPONSE	ON	0x00000001	Device not responding
	OFF	0x00000000	Device responding
ASB_OFF_LINE	ON	0x00000008	Offline
	OFF	0x00000000	Online
ASB_COVER_OPEN	ON	0x00000020	Cover is open
	OFF	0x00000000	Cover is closed
ASB_UNREOVER_ERR	ON	0x00002000	Unrecoverable error occurred
	OFF	0x00000000	No unrecoverable error

Device ID

Device ID (Set in prnID)	Content of Device ID	Device Information
64	Manufacturer name	String data. "EPSON"
65	Model name	String data. "PLQ-22"
66	Firmware version	String data.
67 *	Device serial number	String data (NULL terminated). Ex. "ABCD012345"
80	Implementation of the Magnetic Stripe	31h: Implemented
		30h: Not implemented
81	Passbook support	31h: Supported
		30h: Not supported
82	CSF implementation	31h: Implemented
		30h: Not implemented

* Compatible with PLQ-22KCS/PLQ-22KCSM only

PLQ-22 .NET API Reference

This chapter explains the PLQ-22 API used in a .NET environment.

MFDevice Class

Method	Description
MFDevice Constructor	Constructor
ResetLastError	Resets the error code value returned after the API was called
OpenMonPrinter	Starts monitoring the status of the device
CloseMonPrinter	Stops monitoring the status of the device
ResetPrinter	Resets the device
GetPrnCapability	Retrieves the device information
SCNMICRFunction	Reads one check
SetStatusBack	Registers status callback
SetMultiStatusBack	Registers status callback
CancelStatusBack	Cancels the registration of status callback
LockPort	Locks the port.
UnlockPort	Unlocks the port.
GetVersion	Retrieves the driver and module versions
MICRClearSpaces	Removes the space characters in the MICR characters
MSRWSetting	Sets Magnetic Stripe control
MSRWReadStripeData	Reads the Magnetic Stripe
MSRWGetStripeData	Retrieves the read Magnetic Stripe data
MSRWWriteStripeData	Writes data to the Magnetic Stripe
MSRWClearStripeData	Clears the Magnetic Stripe data
EjectPaper	Ejects the inserted paper

Properties

Property	Description
IsValid	Checks if Open in the extended constructor succeeded
LastError	Retains the error returned after the API was called
Status	Retrieves the latest status of the device
MultiStatus	Retrieves the latest status of the device

Events

Event	Description
StatusCallback	Registers so that the status is notified when it changes
StatusCallbackEx	Registers so that the status and the port name are notified when the status changes
MultiStatusCallbackEx	Registers so that the status type, the status, and the port name are notified when the status changes

MFDevice

A constructor of MFDevice class.
Generates a class which defines the APIs that control this driver.

Syntax

MFDevice( )

ResetLastError

Sets the return value of a called method, retained by the .NET API, to SUCCESS.

Syntax

void ***ResetLastError***( )

Description

Sets the value retrieved from the property LastError to “SUCCESS”, until another method gets called and returns a return value.

OpenMonPrinter

Starts monitoring the status of the specified device. Refer to "BiOpenMonPrinter" on page 111 for details.

Syntax

ErrorCode ***OpenMonPrinter*** (OpenType type, String name)

Argument

OpenType type: Specifies “TYPE_PRINTER” (constant)

String name: Specifies the name of the device whose status you want to monitor.

Example

<Local Connection>

ErrorCode OpenMonPrinter(“PLQ-22U”);

<Network Connection>

ErrorCode OpenMonPrinter(“192.168.192.168\PLQ-22U”);

ErrorCode OpenMonPrinter(“SCANNER SERVER\PLQ-22U”);

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_TYPE	Type parameter error
ERR_OPENED	The specified device is already open
ERR_NO_PRINTER	The specified device cannot be found
ERR_NO_MEMORY	Insufficient memory
ERR_TIMEOUT	Timeout error
ERR_PARAM	Parameter error
ERR_NET_CONNECTED	Failed to connect to the device on the network

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

CloseMonPrinter

Stops monitoring the status of the device.Refer to "BiCloseMonPrinter" on page 125 for details.

Syntax

ErrorCode ***CloseMonPrinter*** ( )

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

ResetPrinter

Resets the device whose status is being monitored. Refer to "BiResetPrinter" on page 120 for details.

NOTE

- Use this API when a recoverable error occurs. Refer to "Device Status" on page 33 for details on recoverable errors.
- This is an exclusive API.Before calling it, call LockPort (page 160).
  After calling it, call UnlockPort (page 161).

Syntax

ErrorCode ***ResetPrinter*** ( )

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

GetPrnCapability

Retrieves the device information. Refer to "BiGetPrnCapability" on page 124 for details.

> **NOTE** This is an exclusive API. Before calling it, call LockPort (page 160).
> After calling it, call UnlockPort (page 161).

Syntax

ErrorCode ***GetPrnCapability*** (byte printerID, out byte() data)
ErrorCode ***GetPrnCapability*** (byte printerID, out String data)

Argument

byte printerID: Specifies the device ID of the device information you want to retrieve.

byte[] data: Specifies the array length for setting the retrieved information (1 to 80).

String data: Specifies the string for the setting the retrieved information.

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_TIMEOUT	Timeout error
ERR_ACCESS	Cannot read/write to the device
ERR_PARAM	Parameter error
ERR_OFFLINE	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset

> **NOTE** For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

For a list of device information that can be retrieved, refer to "Device ID" on page 148.

SCNMICRFunction

Executes/sets MICR data reading. Refer to "BiSCNMICRFunction" on page 121 for details.

> NOTE
> This is an exclusive API. Before calling it, call LockPort (page 160).
> After calling it, call UnlockPort (page 161).

Syntax

ErrorCode ***SCNMICRFunction*** (FunctionType functionType)

Argument

FunctionType functionType:

Specifies the function to execute with this API.

functionType (Constant)	Description
MF_EXEC	Execute MICR data reading
MF_MICR_RETRANS	Retrieve the previously read MICR data

ErrorCode ***SCNMICRFunction*** (MF mf, FunctionType functionType)

Argument

MF mf: Sets an instance of MFBase and MFMicr classes.

FunctionType functionType:

Specifies the function to execute with this API.

wFunction (Constant)	Description
MF_SET_BASE_PARAM	Set MFBase class to the driver
MF_SET_MICR_PARAM	Set MFMicr class to the driver
MF_CLEAR_BASE_PARAM	Clear MFBase class that is set
MF_CLEAR_MICR_PARAM	Clear MFMicr class that is set
MF_GET_BASE_DEFAULT	Retrieve the default values of MFBase class.
MF_GET_MICR_DEFAULT	Retrieve the default values of MFMicr class.

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_NO_MEMORY	Insufficient memory
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_TIMEOUT	Timeout error
ERR_ACCESS	Cannot read/write to the device
ERR_PARAM	Parameter error
ERR_OFFLINE	Cannot use: Waiting to return from an offline state
ERR_PAPERINSERT_TIMEOUT	Failed to insert paper
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_MICR	Failed to read the MICR data
ERR_PAPER_JAM	Paper jam error
ERR_PAPER_INSERT	Failed to insert paper

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

SetStatusBack

Starts the status notification of StatusCallback or StatusCallbackEx (page 173) event. Refer to "BiSetStatusBackFunctionEx" on page 116 for details.

Syntax

ErrorCode ***SetStatusBack*** ( )

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

For a list of device statuses that can be retrieved with this API, refer to "Device Status" on page 147.

SetMultiStatusBack

Starts the status notification of MultiStatusCallback (page 173) event.
Refer to "BiSetMultiStatusBackFunction" on page 117 for details.

Syntax

ErrorCode ***SetMultiStatusBack*** ( )

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

For a list of device statuses that can be retrieved with this API, refer to "Device Status" on page 147.

CancelStatusBack

Cancels the automatic status notification request process that was called by SetStatusBack or SetMultiStatusBack.
Refer to "BiCancelStatusBack" on page 119 for details.

Syntax

ErrorCode ***CancelStatusBack*** ( )

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.

NOTE

- For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.
- Returns “SUCCESS” even if you call this API by mistake while the automatic status notification request process has not been registered.

LockPort

Locks the port and takes sole possession of the execution right for the exclusive API (API that directly accesses the device).
Refer to "BiLockPort" on page 135 for details.

> **CAUTION** After locking the port with this API and executing the exclusive API, be sure to execute UnlockPort (page 161).

Syntax

ErrorCode ***LockPort*** (uint timeout)

Argument

uint timeout: Specifies the timeout value in ms (milliseconds).

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_TIMEOUT	Timeout error
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset

> **NOTE** For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

UnlockPort

Unlocks the port. Refer to "BiUnlockPort" on page 137 for details.

Syntax

ErrorCode ***UnlockPort*** ( )

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_RESET	Cannot use: Device is being reset
ERR_NOT_EXEC	Process not executed

NOTE

- For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.
- If the port was not locked, “ERR_NOT_EXEC” is returned.

GetVersion

Retrieves the version of drivers and modules.

Refer to "BiGetVersion" on page 126 for details.

> **NOTE** This API can be called without starting the status monitoring (OpenMonPrinter).

Syntax

ErrorCode ***GetVersion*** (DriverType eDriverType, VersionType eType, MFVersion ptVersion)

Argument

DriverType eDriverType:

Specifies the driver whose version you want to retrieve.

For PLQ-22 .NET API, specify “DRIVER_TYPE_PLQ22” (constant).

VersionType eType:

Specifies the version to retrieve.Specify from the following:

eType (Constant)	Type
VERSION_TYPE_DRIVER	Scanner driver version
VERSION_TYPE_OCR	OCR recognition module version
VERSION_TYPE_IMAGE	Image processing module version
VERSION_TYPE_BARCODE	Barcode module version
VERSION_TYPE_PDF	PDF file creator module
VERSION_TYPE_DDE1	Retrieves the DDE module version
VERSION_TYPE_DDE2	Retrieves the DDE module version
VERSION_TYPE_PH	Communication module

MFVersion ptVersion:

Specifies the MFVersion class object. The retrieved version is set. Refer to "MFVERSION Class - Properties" on page 182 for details.

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_PARAM	Parameter error
ERR_NOT_FOUND	The specified module cannot be found

> **NOTE** For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

MICRClearSpaces

Removes the space characters contained in the acquired MICR data.
Refer to "BiMICRClearSpaces" on page 127 for details.

Syntax

ErrorCode ***MICRClearSpaces*** (RemoveSpace removeSpace)

Argument

RemoveSpace removeSpace:

Disables/enables the removal of the space characters contained in the MICR data.
Specify from the following:

removeSpace (Constant)	Description
CLEAR_SPACE_DISABLE (Default value)	Space characters not removed
CLEAR_SPACE_ENABLE	Space characters removed

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Cannot use: Another API is running

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

The MICR data subject to the removal of the space characters by this API are the following properties of MFMICR class:

- MicrStr: MICR string
- OcrReliableInfo:Reliability information of MICR character recognition

MSRWSetting

Sets the necessary information to the device to read, write, and clear the Magnetic Stripe data on passbooks. Refer to "BiMSRWSetting" on page 128 for details.

> **NOTE** This API is proprietary to PLQ-22 CSM.

Syntax

ErrorCode ***MSRWSetting*** (MFMsrw mfSetting)

Argument

pSetting: Specifies MFMsrw class. Refer to "MFMSRW Class - Properties" on page 183 for details.

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	This API is not supported
ERR_OFFLINE	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset

> **NOTE** For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

MSRWReadStripeData

Waits for the user to insert the passbook and reads the Magnetic Stripe. Refer to "BiMSRWReadStripeData" on page 129 for details.

NOTE

- This is an exclusive API. Before calling it, call LockPort (page 160). After calling it, call UnlockPort (page 161).
- The retrieval of the Magnetic Stripe data is done with MSRWGetStripeData.
- This API is proprietary to PLQ-22 CSM.

Syntax

ErrorCode ***MSRWReadStripeData*** (uint timeout)

Argument

uint timeout: Sets the timeout value for loading the passbook, in ms.

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_TIMEOUT	Timeout error
ERR_ACCESS	Cannot read/write to the device
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	This API is not supported
ERR_OFFLINE	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

MSRWGetStripeData

Retrieves the Magnetic Stripe data that was read with MSRWReadStripeData (page 165). Refer to "BiMSRWGetStripeData" on page 130 for details.

NOTE

- This is an exclusive API. Before calling it, call LockPort (page 160). After calling it, call UnlockPort (page 161).
- This API is proprietary to PLQ-22 CSM.

Syntax

```
ErrorCode MSRWGetStripeData ( uint blocknumber,
                              ref byte() read,
                              out uint returnlength,
                              out uint detail )
```

Argument

uint blocknumber: Specifies the block number of the Magnetic Stripe data you are retrieving. The block numbers that can be specified are “1” and “2”.

NOTE

- To retrieve the Magnetic Stripe data in the second block, BlockCount of MFMsrw class must be set to “2”.
- To retrieve the Magnetic Stripe data from both the first and second blocks, retrieve the Magnetic Stripe data in the first block, and then use this API again to retrieve the Magnetic Stripe data in the second block.

byte[] read: Specifies the buffer for retrieving data.

out uint returnlength:

Sets the data size of the Magnetic Stripe data.

NOTE

returnlength is set when “NULL” is specified for read.

out uint detail: The details of the reading result are set.

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	This API is not supported
ERR_OFFLINE	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset
ERR_NOT_EXEC	Reading process not executed
ERR_MSRW_NODATA	Magnetic Stripe data not detected.

NOTE For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Description

To retrieve the Magnetic Stripe data, this API needs to be called twice.

First time: Retrieves the data size of the Magnetic Stripe data.

Second time: Retrieves the Magnetic Stripe data.
(The data size retrieved from the first call is used in the read array in the second parameter of the second call.)

<Example>

```
' Retrieve the data size of the Magnetic Stripe data
eErrorCode = m_objMfDevice.MSRWGetStripeData(1, Nothing, iReadLength, iDetail)

' Retrieve the Magnetic Stripe data
If iReadLength <> 0 Then
    ReDim bRead(CInt(iReadLength - 1))
End If
eErrorCode = m_objMfDevice.MSRWGetStripeData(1, bRead, iReadLength, iDetail)
```

MSRWWriteStripeData

Waits for the user to insert the passbook and writes the specified data to the Magnetic Stripe. Refer to "BiMSRWWriteStripeData" on page 132 for details.

NOTE

- This is an exclusive API. Before calling it, call LockPort (page 160). After calling it, call UnlockPort (page 161).
- This API is proprietary to PLQ-22 CSM.

Syntax

ErrorCode ***MSRWWriteStripeData*** (uint timeout, byte[] write)

Argument

uint timeout: Sets the timeout value for loading the passbook, in ms.

byte[] write: Specifies the data to write (30H to 3FH).

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_TIMEOUT	Timeout error
ERR_ACCESS	Cannot read/write to the device
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	This API is not supported
ERR_OFFLINE	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset
ERR_MSRW_WRITE	Failed to write Magnetic Stripe data

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

MSRWClearStripeData

Waits for the user to insert the passbook and clears the Magnetic Stripe data. Refer to "BiMSRWClearStripeData" on page 133 for details.

NOTE

- This is an exclusive API. Before calling it, call LockPort (page 160). After calling it, call UnlockPort (page 161).
- This API is proprietary to PLQ-22 CSM.

Syntax

ErrorCode ***MSRWClearStripeData*** (uint timeout)

Argument

uint timeout: Sets the timeout value for loading the passbook, in ms.

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_TIMEOUT	Timeout error
ERR_ACCESS	Cannot read/write to the device
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	This API is not supported
ERR_OFFLINE	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset
ERR_MSRW_WRITE	Failed to clear the Magnetic Stripe data

NOTE

For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

EjectPaper

> **NOTE** This is an exclusive API. Before calling it, call LockPort (page 160).
> After calling it, call UnlockPort (page 161).

Ejects the inserted paper. Refer to "BiEjectPaper" on page 134 for details.

Syntax

ErrorCode ***EjectPaper*** (MfEjectType type, uint timeout)

Argument

MFEjectType type:

Specifies where the paper should be ejected to. The value can be specified from the following:

type (Constant)	Description
MF_EJECT_FRONT	Eject to the front
MF_EJECT_REAR	Eject to the rear

uint timeout: Specifies the timeout value in ms (milliseconds).

Return value

Macro Definition (Constant)	Description
SUCCESS	Success
ERR_HANDLE	The monitoring of the device status has not been started.
ERR_TIMEOUT	Timeout error
ERR_ACCESS	Cannot read/write to the device
ERR_PARAM	Parameter error
ERR_OFFLINE	Cannot use: Waiting to return from an offline state
ERR_EXEC_FUNCTION	Cannot use: Another API is running
ERR_RESET	Cannot use: Device is being reset
ERR_UNLOCKED	The port is not locked

> **NOTE** For a list of return values and troubleshooting actions, refer to "Return Values" on page 34.

Properties

IsValid

Retrieves the result of executing OpenMonPrinter.

Access

Read only

Data type

bool type

Value

true: OpenMonPrinter succeeded

false: OpenMonPrinter failed

LastError

Retrieves the return value after executing a PLQ-22 .NET API.

Access

Read only

Data type

Enum ErrorCode type

Value

Refer to "Return Values" on page 34 for details.

Status

Retrieves the current device status.

Access

Read only

Data type

Enum ASB type

Value

Refer to "BiGetStatus" on page 113 for details.

MultiStatus

Retrieves the current device status.

Access

Read only

Data type

Enum EscMcASB[4] type

Value

Refer to "BiGetMultiStatus" on page 114 for details.

Events

StatusCallback/ StatusCallbackEx/MultiStatusCallback

Notifies the application of the device status when it changes.

Event

event StatusCallbackHandler StatusCallback;
event StatusCallbackHandlerEx StatusCallbackEx;
event MultiStatusCallbackHandler MultiStatusCallback;

Delegate

delegate void StatusCallbackHandler(ASB asb);
delegate void StatusCallbackHandlerEx(ASB asb, String portName);
delegate void MultiStatusCallbackHandler(EscMcASBType type,
EscMcASB mcasb,
String portName);

Parameters

ASB asb:
The device status is set.
Refer to "Device Status" on page 147 for details.

EscMcASBType type:
The type of device status retrieved is set.
Refer to "BiSetMultiStatusBackFunction" on page 117 for details.

EscMcASB mcasb:
The device status is set.
Refer to "Device Status" on page 147 for details.

String portName:
The name of the port that was called back is set.

MFBASE Class - Properties

Version

Retrieves the version of MFBase class.

Access

Read only

Data type

int type

Value

The version of MFBase class

Ret

Access

Read only

Data type

Enum ErrorCode type

Value

ErrorCode (Constant)	Description
SUCCESS	Success
ERR_NO_MEMORY	Insufficient memory
ERR_TIMEOUT	Timeout error
ERR_ACCESS	Cannot read/write to the device
ERR_PARAM	Parameter error
ERR_PAPERINSERT_TIMEOUT	Failed to insert paper
ERR_MICR	Failed to read the MICR data
ERR_NOT_EXEC	Process not executed

Timeout

Sets the wait time for paper insertion in seconds.

Access

Read/write

Data type

int type

Value

0 to 300

ErrorEject

Specifies where the paper is ejected when an error occurs during the reading process.

Access

Read/write

Data type

Enum MfEjectType type

Value

MfEjectType (Constant)	Description
MF_EJECT_DISCHARGE	Eject to the front.
MF_EJECT_RELEASE	Eject to the rear.

SuccessEject

Specifies where the paper is ejected when the reading process completes successfully.

Access

Read/write

Data type

Enum MfEjectType type

Value

MfEjectType (Constant)	Description
MF_EJECT_DISCHARGE	Eject to the front.
MF_EJECT_RELEASE	Eject to the rear.

PortName

Retrieves the port name assigned by the driver.

Access

Read only

Data type

String type

Value

Port name assigned by the driver

MFMICR Class - Properties

Version

Retrieves the version of MFMicr class.

Access

Read only

Data type

int type

Value

Version of MFMicr class

Ret

Access

Read only

Data type

Enum ErrorCode type

Value

ErrorCode (Constant)	Description
SUCCESS	Success
ERR_NO_MEMORY	Insufficient memory
ERR_TIMEOUT	Timeout error
ERR_ACCESS	Cannot read/write to the device
ERR_PARAM	Parameter error
ERR_NOT_FOUND	MICR data not detected
ERR_MICR	Failed to read the MICR data
ERR_NOT_EXEC	Process not executed

Font

Sets the font of the MICR data to read.

Access

Read/write

Data type

Enum MfMicrFont type

Value

MfMicrFont (Constant)	Description
MF_MICR_FONT_E13B	E13B font
MF_MICR_FONT_CMC7	CMC7 font (including numbers)
MF_MICR_FONT_CMC7_ALPHANUM	CMC7 font (including alphabets and numbers)
MF_MICR_FONT_CMC7_NUM	CMC7 font (including numbers)

MicrOcrSelect

Specifies the reading operation of the MICR data.

Access

Read/write

Data type

Enum MfMicrType type

Value

MfMicrType (Constant)	Description
MF_MICR_USE_MICR	Retrieve the analysis result for magnetic waveform data only (No retries)
MF_MICR_USE_MICR_RETRY	Retrieve the analysis result for magnetic waveform data only (With retries)
MF_MICR_USE_MICOCR_RETRY1	Retrieve the analysis results of magnetic waveform data and optical character recognition (With retries. Initially reads using magnetic waveform)
MF_MICR_USE_MICOCR_RETRY2	Retrieve the analysis results of magnetic waveform data and optical character recognition font (With retries. Initially read using both magnetic waveform and optical recognition)

Status

Retrieves the MICR reading status.

Access

Read only

Data type

byte type

Value

Refer to "MF_MICR" on page 141 for details.

Detail

Retrieves the detailed MICR reading status.

Access

Read only

Data type

byte type

Value

Refer to "MF_MICR" on page 141 for details.

MicrStr

Retrieves the read MICR data.

Access

Read only

Data type

String type

Value

MICR data that was read

OcrReliableInfo.FirstSelectString

Retrieves the OCR recognized string that is likely to be the closest to the actual string.

Access

Read only

Data type

String type

Value

First recognition candidate data

OcrReliableInfo.SecondSelectString

Retrieves the OCR recognized string that is likely to be the second closest to the actual string.

Access

Read only

Data type

String type

Value

Second recognition candidate data

OcrReliableInfo.FirstSelectPercentage

Retrieves the reliability in percentage of the OCR recognized characters that are likely to be the closest to the actual string.

Access

Read only

Data type

int[] type

Value

Reliability of the first recognition candidate data (%)

OcrReliableInfo.SecondSelectPercentage

Retrieves the reliability in percentage of the OCR recognized characters that are likely to be the second closest to the actual string.

Access

Read only

Data type

int[] type

Value

Reliability of the second recognition candidate data (%)

MFVERSION Class - Properties

Description

Retrieves the information retrieved by GetVersion.

Access

Read only

Data type

String type

Value

Information retrieved by GetVersion

Version

Retrieves version information.

Access

Read only

Data type

String type

Value

Version information

MFMSRW Class - Properties

Version

Retrieves the version of MFMsrw class.

Access

Read only

Data type

int type

Value

Version of MFMsrw class

RecordFormat

Sets the record format.

Access

Read/write

Data type

Enum MsrwFormat type

Value

MsrwFormat (Constant)	Record Format
EPS_BI_MSRW_FORMAT_IBM3604	IBM3604
EPS_BI_MSRW_FORMAT_DINISO	DINISO
EPS_BI_MSRW_FORMAT_ANSI	ANSI
EPS_BI_MSRW_FORMAT_IBM4746	IBM4746
EPS_BI_MSRW_FORMAT_ISO7811	ISO7811
EPS_BI_MSRW_FORMAT_HT2751CIZ	HT2751CIZ
EPS_BI_MSRW_FORMAT_ISO8484	ISO8484

IBMDataType

Sets the EOM of the record format (dwRecordFormat).

Access

Read/write

Data type

Enum MsrwIBMDataType type

Value

MsrwIBMDataType (Constant) \ MsrwFormat	EPS_BI_MSRW_FORMAT_IBM3604	EPS_BI_MSRW_FORMAT_DINISO	EPS_BI_MSRW_FORMAT_ANSI	EPS_BI_MSRW_FORMAT_IBM4746	EPS_BI_MSRW_FORMAT_ISO7811	EPS_BI_MSRW_FORMAT_HT2751CIZ	EPS_BI_MSRW_FORMAT_ISO8484
EPS_BI_MSRW_EOM_NONE	○	○	○	○	○	○	○
EPS_BI_MSRW_EOM_3C	○	-	-	○	-	-	-
EPS_BI_MSRW_EOM_3F	○	-	-	○	-	-	-

ReadRetry

Sets the number of retries for reading the Magnetic Stripe.

Access

Read/write

Data type

uint type

Value

0 to 3

WriteAndClearRetry

Sets the number of retries for writing and clearing the Magnetic Stripe.

Access

Read/write

Data type

uint type

Value

0 to 3

BlockCount

Specifies the block to read/write the Magnetic Stripe data.

Access

Read/write

Data type

uint type

Value

BlockCount (Value)	PLQ-22 .NET API	Description
1	MSRWReadStripeData	Read data in the first block
	MSRWWriteStripeData	Write data to the first block
2	MSRWReadStripeData	Read data in the second block
	MSRWWriteStripeData	Write data to the second block

VerticalAdjust

Sets the correction distance in vertical direction in units of 0.1 mm.

Access

Read/write

Data type

uint type

Value

0 to 5588

Sample Programs

This chapter explains the sample programs necessary for developing the application.

TWAIN API

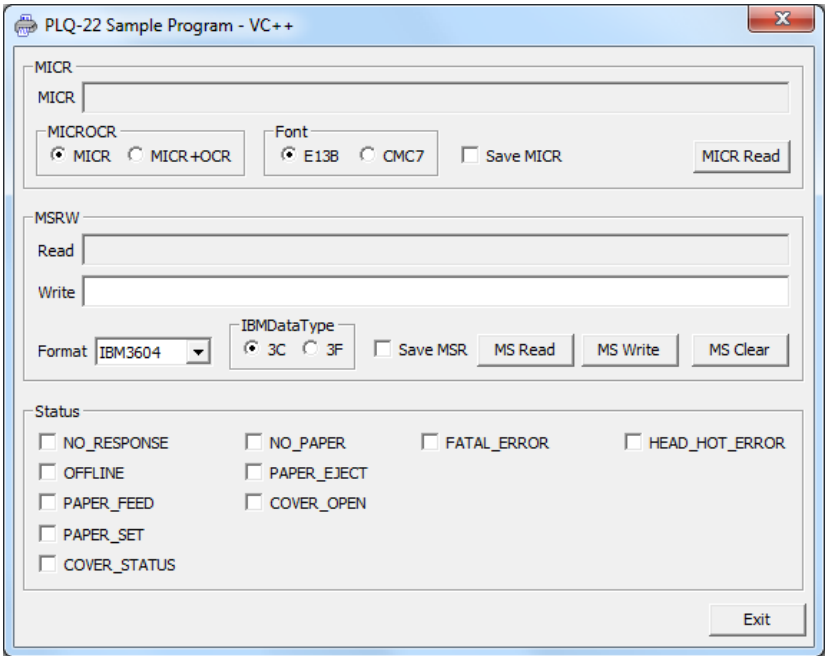
For TWAIN API, use the sample programs provided by the TWAIN Working Group. Visit the TWAIN Working Group website:

http://www.twain.org/

PLQ-22 API

The sample program that uses the PLQ-22 API has the following functions implemented:

- Reading and saving MICR to file
- Reading, writing, and clearing MS (Magnetic Stripe data)
- Retrieving device status
- Exclusive control



CAUTION For the sample programs, while TWAIN API is activated, the functions other than the Status function (those for MSRW and MICR) are not available.

Folder Structure

The folder structure of the sample programs is as follows:

```
PLQ22Sample.zip
└PLQ22Sample
  ├VB.NET     //Sample program for Visual Basic .NET
  │ ├bin
  │ │└Release
  │ │  └  SampleProgram.exe        //EXE file
  │ ├My Project
  │ │└  AssemblyInfo.vb            //Assembly attribute file (automatically generated by Visual Basic .NET)
  │ ├APIControl.vb                 //Implementation file for PLQ-22 API control
  │ ├MainForm.Designer.vb          //Form designer file
  │ ├MainForm.resx                 //.NET managed resources file
  │ ├MainForm.vb                   //Implementation file for form controls
  │ ├SampleProgram.ico             //Icon of the EXE file
  │ ├SampleProgram.sln             //Microsoft Visual Studio Solution
  │ └SampleProgram.vbproj          //Visual Basic project file
  └VC++        //Sample program for Visual C++
    ├ include
    │ ├PLQStatusApiDef.h                 //Header file for PLQ-22 API
    │ └PLQStatusApiPartialInterface.h    //Header file for PLQ-22 API
    └src
      ├Release
      │ └SampleProgram.exe         //EXE file
      ├APIControl.cpp              //Implementation file for PLQ-22 API control
      ├APIControl.h                //Header file for PLQ-22 API control
      ├ReadMe.txt                  //Standard file (automatically generated by Visual C++)
      ├resource.h                  //Standard file (automatically generated by Visual C++)
      ├SampleProgram.cpp           //Definition file for the class behaviors
      ├SampleProgram.h             //Header file for the class behaviors
      ├SampleProgram.ico           //Icon of the EXE file
      ├SampleProgram.rc            //Resource script
      ├SampleProgram.sln           //Microsoft Visual Studio Solution
      ├SampleProgram.vcproj        //Visual C++ project file
      ├SampleProgramDlg.cpp        //Implementation file for dialog controls
      ├SampleProgramDlg.h          //Header file for dialog controls
      ├stdafx.cpp                  //Standard file (automatically generated by Visual C++)
      └stdafx.h                    //Standard file (automatically generated by Visual C++)
```